RSX–11

Utilities Manual

Order No. AA-H268A-TC

RSX-11M Version 3.2
RSX-11M-PLUS Version 1.0

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, June 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.

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:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

APPENDIXES

CONTENTS

FIGURES

Page

TABLES (Cont.)

SUMMARY OF TECHNICAL CHANGES

Utilities can now be used from the DIGITAL Command Language (DCL) environment (on RSX-11M-PLUS) as well as from the Monitor Console Routine (MCR) environment (on RSX-11M and RSX-11M-PLUS).

The RSX-11M Print Spooler Task (PRT) is now called the Serial Despooler Task. The new name distinguishes it from another task named PRT that is part of the Queue Manager (see below). The choice of Serial Despooler or Queue Manager is made at system generation. All switches or commands that despool output operate the same way with either task.

The following lists the technical changes (such as new functionality, new and revised switches, and new utilities) by utility, in the order in which the utility appears in this manual.

Peripheral Interchange Program (PIP)

/CD - Creation Date (new) - allows the output file to take the creation date of the input file rather than the date of transfer.

/EOF - End-of-file (new) - specifies the end-of-file pointer for a file.

/FR - Free blocks (revised) - displays the amount of free space on the specified volume and the largest contiguous free space on that volume.

/LD - List Deleted files (new) - lists the files that have been deleted.

/NM - No Message (new) - suppresses certain PIP error messages.

/TR - Truncate (new) - truncates file(s) to logical end-of-file.

/SD - Selective Delete (new) - deletes files selectively by prompting for user response before deleting them.

/SR - Shared Reading (new) - allows shared reading of a file that has already been opened for writing by another user or task.

/SU - Supersede (revised) - allows you to copy one or more input files to a file whose file name, file type, and version number already exist in a User File Directory.

File Transfer Program (FLX)

FLX now supports RK06/RK07, RL01, and RX02 disks, and the TU58 data cartridge. All devices are RT-11 compatible only.

/DNS:n - Density (new) - specifies the density of the magnetic tape; n is 800 or 1600 bpi.

/RW - Rewind (new) - rewinds the magnetic tape before beginning the file transfer.

The Queue Manager

The Queue Manager is a collection of programs that provides for the orderly processing of queued files. The Queue Manager allows you to specify how, when, and where a file will be despooled. You can also display information about the queue. On RSX-11M-PLUS only, the Queue Manager supports batch processing.

This manual describes the Queue Manager for RSX-11M V03.2 only. For information on the Queue Manager for RSX-11M-PLUS, see the RSX-11M-PLUS Batch and Queue Operations Manual.

Disk Volume Formatter (FMT)

FMT is a new utility that formats disk volumes (cartridges, packs, and flexible disks) and, optionally, spawns the Bad Block Locator Utility (BAD) if your system supports spawned tasks. FMT allows you to format an entire volume or, where permitted, individual tracks or sectors of a volume.

Bad Block Locator Utility (BAD)

/MAN - Manual (new) - prompts you for bad block information and enters the blocks you specify in the bad block descriptor file.

/UP - Update (new) - allows you to update the bad block descriptor file by entering additional bad block information in response to prompts.

Back-Up and Restore Utility (BRU)

BRU is a new fast back-up and restore utility that features:

- Disk-to-tape back-up
- Tape-to-disk restore
- Disk-to-disk copying
- Incremental back-up or restore
- Copying from an unmounted disk
- Copying to a disk that is either unmounted (BRU initializes the disk) or mounted (using the file system)

Disk Save and Compress Program (DSC)

DSC now supports TS04 magnetic tape drives.

Specifying a label name for an output volume is now permitted.

/BAD - Bad Block Locator Utility (revised) - agrees with the syntax for the current version of the BAD utility.

/BL - Blocking Factor (new) - sets the blocking factor at which DSC accesses blocks of data by enabling the user to change the number of 512(10)-byte blocks in each of DSC's internal buffers.

Librarian Utility Program (LBR)

LBR can now create and maintain universal libraries. Any file can be installed as a module into a universal library.

/MH - Modify Header (new) - allows you to add information to a universal library header.

File Dump Utility (DMP)

/FI - File Identification (new) - allows you to enter a file number as a file identifier instead of a file name.

/HF - Header Format (new) - formats blocks that have Files-11 header structures. Other blocks are output as an unformatted octal dump.

/RW - Rewind (new) - issues a rewind command before it refers to a specific tape.

Source Language Input Program (SLP)

SLP and its audit trail switch (/AU) have been modified to allow a maximum line-length of 132(10) characters.

/CS - Checksum (new) - calculates the checksum value for the edit commands.

/TR - Truncate (new) - reports lines that have been truncated by the audit trail.

PREFACE

MANUAL OBJECTIVES

The RSX-11 Utilities Manual is a reference manual describing the use of the 17 utilities supported by DIGITAL on the RSX-11M and RSX-11M-PLUS operating systems.

INTENDED AUDIENCE

This manual is for all users of the RSX-11M and RSX-11M-PLUS operating systems.

STRUCTURE OF THIS DOCUMENT

Chapter 1 describes briefly each of the utilities, and explains how to enter command lines and how to invoke and use the utilities.

Chapter 2 describes the DEC Standard Editor (EDT).

Chapter 3 describes the Line Text Editor (EDI).

Chapter 4 describes the Peripheral Interchange Program (PIP).

Chapter 5 describes the File Transfer Program (FLX).

Chapter 6 describes the Print and Queue Utility (PRI and QUE).

Chapter 7 describes the Queue Manager.

Chapter 8 describes the Disk Volume Formatter (FMT).

Chapter 9 describes the Bad Block Locator Utility (BAD).

Chapter 10 describes the Back-Up and Restore Utility (BRU).

Chapter 11 describes the Disk Save and Compress Program (DSC).

Chapter 12 describes the Preservation Utility (PRESRV).

Chapter 13 describes the File Structure Verification Utility (VFY).

Chapter 14 describes the Librarian Utility Program (LBR).

Chapter 15 describes the File Dump Utility (DMP).

Chapter 16 describes the File Compare Program (CMP).

Chapter 17 describes the Source Language Input Program (SLP).

Chapter 18 describes the Object Module Patch Utility (PAT).

Chapter 19 describes the Task/File Patch Program (ZAP).

Appendix A is a summary of the commands and switches for the utilities.

Appendix B contains examples of the LBR, EDI, and DMP utilities.

Appendix C describes the RSX-11M Serial Despooler Task (see the Summary of Technical Changes).

Appendix D describes the Cross-Reference Processor (CRF).

ASSOCIATED DOCUMENTS

The RSX-11M/RSX-11S Documentation Directory and the RSX-11M-PLUS Documentation Directory briefly describe the manuals in the documentation set for each system. With them, you can find out where to obtain more information.

The RSX-11M/M-PLUS MCR Operations Manual describes the Monitor Console Routine (MCR) environment and its commands. The utilities can be invoked from the MCR environment. This manual provides background information about MCR.

The RSX-11M-PLUS Command Language Manual describes the DIGITAL Command Language routine (DCL) and its commands. The utilities can be invoked from the DCL environment. This manual provides background information about DCL.

CONVENTIONS USED IN THIS DOCUMENT

Use of Second Color

User (operator) input appears in red.

Use of Uppercase Characters

Uppercase characters in a command line indicate characters that must be entered as they are shown. For example, utility switches must always be entered as they are shown in format specifications. An exception is the <CR> symbol, which denotes a carriage return.

Use of Lowercase Characters

Lowercase letters, words, or symbols in command line format specifications represent variables for which the user substitutes values. For example:

```
filename.filetype;version
```

The line represents the values that comprise a file specification; values are substituted for each of these variables as appropriate.

Command Abbreviations

Where short forms of commands are allowed, the shortest form acceptable is represented by uppercase characters. The following example shows the minimum abbreviation allowed for the EDI WRITE command:

```
Write
```

This notation means that W, WR, WRI, WRIT, and WRITE are all valid specifications for the WRITE command.

Use of Brackets ([])

Brackets denote optional entries in a specification. Brackets also are a part of the User File Directory portion of file specifications, that is, [group,member]. When this portion of a file specification is entered explicitly, brackets are required syntax elements; that is, they do not indicate optional entries. Note that when an option is entered, the brackets are not included in the command line.

Use of commas (,)

Commas are used as separators for command line parameters and indicate positional entries on a command line.

Use of At Sign (@)

The at sign (@) invokes an indirect command file. The at sign immediately precedes the file specification for the indirect command file:

```
@filename.filetype;version
```

Use of Periods (.)

Periods in the file specification separate the file name and file type. When only the file name is used as the file specification, the period need not be specified.

Use of Semicolons (;)

Semicolons in the file specification separate the file type from the file version. If the version is not specified, the semicolon may be omitted from the file specification.

Use of Slashes (/)

Slashes precede switches in the file specification. When shown in the command line format, they must be specified as shown.

Carriage Return

Command lines are terminated by typing the RETURN key (carriage return) unless otherwise indicated in the text. Two forms used to denote the RETURN key are <CR> and (RET).

PART 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

The RSX-11M and RSX-11M-PLUS operating systems provide several kinds of utilities for your use. Utilities are programs that allow you to work with different kinds of files and the contents of those files, and also with different kinds of media (such as disks, magnetic tapes, and cassettes). The RSX-11 utility programs are listed and described briefly in Section 1.1; reference information for each utility is presented in a separate chapter of this manual. Four appendixes are also included to provide you with information related to the utilities and to describe the Cross-Reference Processor (CRF), which is used with the MACRO-11 assembler and the Task Builder.

In addition to summarizing the RSX-11 utilities, this introduction:

- Describes how to enter RSX-11 command lines and file specifications (Sections 1.2 and 1.3)
- Describes how to invoke utilities (Section 1.4)

1.1 RSX-11 UTILITY PROGRAMS

This manual provides reference information for the following RSX-11 utilities:

Editing Utilities
- DEC Editor (EDT)
- Line Text Editor (EDI)

File Manipulation Utilities
- Peripheral Interchange Program (PIP)
- File Transfer Program (FLX)

File Spooling Utilities
- Print and Queue Utility (PRI and QUE)
- The Queue Manager

Volume Maintenance Utilities
- Disk Volume Formatter (FMT)
- Bad Block Locator Utility (BAD)
- Back-Up and Restore Utility (BRU)
- Disk Save and Compress Program (DSC)
- Preservation Utility (PRESRV)
- File Structure Verification Utility (VFY)

Programming Utilities
- Librarian Utility Program (LBR)
- File Dump Utility (DMP)

Program Maintenance Utilities
- File Compare Utility (CMP)
- Source Language Input Program (SLP)
- Object Module Patch Program (PAT)
- Task/File Patch Program (ZAP)

The following sections briefly describe each utility.

1.1.1 Editing Utilities

DIGITAL supports two editing programs for creating and maintaining text and source files.

1.1.1.1 **DEC Editor (EDT)** - EDT is an interactive text editor that is particularly useful for creating and maintaining text files. EDT has two features that distinguish it from EDI:

- Provides unlimited access to an entire file at one time, making it unnecessary to work with most files in smaller sections. (However, a file can be too large for EDT.)
- Provides character-mode editing for users with video terminals. Character mode allows you to edit at the character and word levels as well as at line level.

1.1.1.2 **Line Text Editor (EDI)** - EDI is a line-oriented, interactive editor used to create and maintain text and source files.

1.1.2 File Manipulation Utilities

DIGITAL provides two file manipulation utilities. With these utilities, you can, among other jobs, copy and spool files and transfer files between volumes.

1.1.2.1 **Peripheral Interchange Program (PIP)** - PIP copies files and performs several file control functions, such as concatenating, renaming, spooling, listing, deleting, and unlocking.

1.1.2.2 **File Transfer Program (FLX)** - FLX is a file transfer and format conversion program that transfers files between DOS-11, RT-11, and Files-11 volumes, with some restrictions.

1.1.3 File Spooling Utilities

File spooling is handled differently on RSX-11M and RSX-11M-PLUS. For RSX-11M-PLUS, see the RSX-11M-PLUS Batch and Queue Operations Manual. RSX-11M systems can include either the Queue Manager or the Print Spooler task, ...PRT. PRT is described in Appendix C.

1.1.3.1 **Print Command and the Queue Manager (PRI and QUE)** - This chapter describes how the user interfaces with the Queue Manager. With the Queue Manager, files can be spooled with the Print command. Files spooled by tasks will also be queued automatically. You can set many attributes of the job with the Print command or with the QUE /MODIFY commands. You can also display the queues and you can alter, hold, or release a job after it has been placed in the queue.

1.1.3.2 **The Queue Manager** - This chapter describes the privileged commands for setting up and running the Queue Manager.

1.1.4 Volume Maintenance Utilities

Volume maintenance includes backing up files onto the volumes, locating bad blocks on the volumes, and verifying the contents of the volumes. DIGITAL provides six volume maintenance programs.

1.1.4.1 **Disk Volume Formatter (FMT)** - FMT formats and verifies RP02/RP03, RP04, RP05, RP06, RM02, and RM03 pack disks, RK05, RK06, and RK07 cartridge disks, and RX02 flexible disks. FMT can:

- Write complete headers for each sector of a disk
- Verify the headers it writes
- Set the density for RX02 flexible disks
- Set the maximum error limit for a disk pack and terminate processing when the limit is reached
- Allow spawning of the Bad Block Locator Utility (if your system allows spawned tasks)

1.1.4.2 **Bad Block Locator Utility (BAD)** - BAD determines the number and location of bad blocks on a volume (including magnetic tape). The information gathered from running BAD on a volume can be used in different ways when that volume is initialized.

1.1.4.3 **Back-Up and Restore Utility (BRU)** - BRU transfers files from a Files-11 volume to one or more back-up volumes (including non-Files-11 volumes) and retrieves files from the back-up volume(s). BRU is faster than DSC or PRESRV (described below) in most areas. Also, BRU compresses data, the volumes do not have to be initialized, and incremental back-ups are possible.

1.1.4.4 **Disk Save and Compress Program (DSC)** - DSC copies Files-11 disk files to disk or tape and from DSC-created tape back to disk. While copying the files, DSC also consolidates the data storage area and writes files in contiguous blocks unless it encounters a bad block. DSC can be run either online or stand-alone.

1.1.4.5 **Preservation Utility (PRESRV)** - PRESRV is a stand-alone program that allows you to create copies of volumes. Unlike DSC, PRESRV does not flag bad blocks or compress volumes.

1.1.4.6 **File Structure Verification Utility (VFY)** - VFY is a disk verification program that verifies the consistency and validity of the file structure on a Files-11 volume.

1.1.5 Programming Utilities

DIGITAL supports two programming utilities. The utilities allow you to work with library files and to examine file contents.

1.1.5.1 **Librarian Utility Program (LBR)** - LBR is a library maintenance program that creates and modifies library files. LBR can process macro, object, and universal libraries.

1.1.5.2 **File Dump Utility (DMP)** - DMP is a file listing program that allows you to examine file contents. DMP also provides options that control the format of the contents.

1.1.6 Program Maintenance Utilities

Program maintenance includes modifying, patching, and comparing files. DIGITAL provides four program maintenance utilities.

1.1.6.1 **File Compare Utility (CMP)** - CMP compares two text files, record by record, and lists the differences between the two files.

1.1.6.2 **Source Language Input Program (SLP)** - SLP is a noninteractive editing program that is used to maintain and audit source files.

1.1.6.3 **Object Module Patch Program (PAT)** - PAT is an object module patch utility that updates, or patches, a relocatable binary object module.

1.1.6.4 **Task/File Patch Program (ZAP)** - ZAP is a patch utility that examines and directly modifies locations in a task image or data file.

1.2 ENTERING RSX-11 COMMAND LINES

The general format for entering command lines to RSX-11 utilities is:

```
outfile,...outfile=infile,...infile<CR>
```

where outfile and infile are file specifications for the output and input files to be operated on by the utility.

This general format varies from utility to utility. Some use the entire command line and others use abbreviated forms of the command line. For some other utilities (such as BRU), the format is different. The syntax for each utility is described in the chapter that describes that utility. The utilities also accept indirect command files containing command lines, as described in Section 1.4.4.

1.3 ENTERING FILE SPECIFICATIONS

In the command line format described in Section 1.2, outfile and infile represent file specifications. The number of file specifications you can enter depends on the utility. The maximum terminal line length depends on the size of the output buffer for your terminal.

The format for entering file specifications is:

```
dev:[group,member]filename.filetype;version/sw.../subsw...<CR>
```

dev:

The physical device containing the desired volume. The name consists of two or three ASCII characters followed by an optional 1-, 2-, or 3-digit octal unit number and a colon, for example, DK0:, TT100:, or DBA0:. For RSX-11M, device names are limited to two ASCII characters.

The default is the system device, SY0:.

[group,member]

The group number and member number associated with the User File Directory (UFD) containing the desired file. Both numbers are octal.

The default is the current UIC.

filename

The name of the file. RSX-11 file names can be null or consist of up to nine alphanumeric characters.

There is no default.

filetype

The file type of the file. The file type provides a convenient means for distinguishing different forms of the same file. For example, a FORTRAN source program file might be named COMP.FTN and the object file for the same program might be named COMP.OBJ. File type and file name are separated by a period. The file type can be zero to three alphanumeric characters. See the RSX-11M/M-PLUS MCR Operations Manual for a list of standard file types.

There is no default.

version

An octal number that specifies different versions of the same file. For example, when a file is created, it is assigned a version number of 1 by default. Thereafter, each time the file is opened and unless you specify otherwise, the file system creates a new file with the same file name and file type, but with a version number incremented by 1. Version numbers can range from 0 through 77777(8). Version number and file type are separated by a semicolon.

The default is the latest version.

/sw

An ASCII name specifying a switch associated with a function to be executed by the utility. Most utility functions are implemented by means of switches and subswitches. Switches can take one of three forms:

```
/sw     invokes the switch function
/-sw    negates the switch function
/NOsw   negates the switch function
```

Switches can take values in the form of ASCII strings and numeric strings.

Most numeric values are octal by default. To specify a decimal number, terminate the number with a decimal point. Values preceded by a pound sign (#) are octal; this optional notation provides explicit documentation of octal values. Any number can be preceded by either a plus (+) or minus (-) sign; plus is the default. Where explicit octal notation (#) is used, the sign, if specified, must precede the pound sign.

The following are valid switch specifications:

```
/SW:27.:MAP:29.
/-SW
/NOSW:-#50:SWITCH
```

/subsw

An ASCII name specifying a subswitch associated with a switch. Subswitches provide a subset of functions related to the main switch function. The following is an example of a subswitch specification:

```
PIP>[200,200]*.*;*/PR/FO<CR>
```

In this example, /FO is a subswitch applied to the PR switch.

Syntactically, subswitches are identical to switches. The rules for entering switches also apply for entering subswitches.

1.4 INVOKING RSX-11 UTILITIES

You can invoke a utility from the Monitor Console Routine (MCR) or DIGITAL Command Language (DCL) environment. MCR is on both the RSX-11M and RSX-11M-PLUS systems. DCL is on RSX-11M-PLUS only. Both MCR and DCL monitor your terminal activity; that is, they accept or reject commands you enter and they display messages. (For more

information on MCR, see the RSX-11M/M-PLUS MCR Operations Manual. For more information on DCL, see the RSX-11M-PLUS Command Language Manual.)

To determine whether you are using MCR or DCL, type CTRL/C, which returns the explicit monitor prompt: either MCR> or DCL>.

You invoke a utility and then work with it directly or by means of indirect command files. For systems in which all utilities are installed, you can use any of three methods to invoke a utility. Sections 1.4.1 and 1.4.2 describe these methods. For systems in which not all utilities are installed, you can use two methods for invoking a utility. Section 1.4.3 describes these methods.

Section 1.4.4 describes how to invoke a utility that can then accept commands from an indirect command file.

You invoke a utility when the RSX-11M/M-PLUS MCR or RSX-11M-PLUS DCL routine prompts you.

For RSX-11M/M-PLUS, the MCR prompts are:

```
>  or (if you type CTRL/C first)  MCR>
```

For RSX-11M-PLUS, the DCL prompts are:

```
>  or (if you type CTRL/C first)  DCL>
```

DCL has commands that access utilities transparently to the user, that is, you do not have to explicitly specify the utility in order to use it. For example, the DCL command DIFFERENCES invokes the File Compare Utility (CMP); and the DCL commands COPY, DELETE, and PURGE invoke the Peripheral Interchange Program (PIP). This transparent access to utilities covers most common instances of utility needs for DCL users. If you use these DCL commands, the general format for specifying files is:

```
>command[/qualifiers] infile outfile
```

DCL users can also use any MCR command forms by using the DCL MCR command. However, if you are using DCL and you want to have full access to the utilities as detailed in this manual, you must invoke the utility using the RUN command. Sections 1.4.2.3 and 1.4.3 describe this procedure.

1.4.1 Installing Utilities on Your System

RSX-11 systems provided in distribution kits require the use of the MCR RUN or DCL RUN command to invoke a utility. Utilities must be installed on your system before you can invoke a utility by its name. To install utilities on your system, use the MCR command INSTALL or the DCL command INSTALL.

1.4.2 Invoking Installed Utilities

You can use three primary methods for invoking installed utilities. Sections 1.4.2.1, 1.4.2.2, and 1.4.2.3 describe the methods.

1.4.2.1 **Invoking a Utility and Returning Control to MCR** - Use one of the following forms of command lines to invoke a utility to execute a function and then return control directly to MCR:

```
>utilityname commandstring<CR>

or

MCR>utilityname commandstring<CR>
```

Using this method to invoke the utility allows you to enter a single command for execution. The utility is loaded, the command is executed, and control returns to MCR. (The method described in Section 1.4.2.3 allows you to enter more than one command line because control returns to the utility rather than to MCR.)

Two exceptions to this command format are the SLP and ZAP utilities. You must first invoke these utilities and then enter the command lines as described in Section 1.4.2.3. (However, you can specify SLP @indirectcommandfile; see Section 1.4.4.)

1.4.2.2 **Invoking a Utility and Returning Control to DCL** - Use the following form of command line to invoke a utility to execute a function and return control directly to DCL:

```
>command<CR>
```

Using this method to invoke the utility allows you to enter a single command for execution. The DCL command transparently accesses the utility (see Section 1.4), the utility is loaded, the command is executed, and control returns to DCL.

Two exceptions to this command format are the SLP and ZAP utilities. You must first invoke these utilities and then enter the command lines as described in Section 1.4.2.3. (However, you can specify SLP @indirectcommandfile; see Section 1.4.4.)

1.4.2.3 **Invoking and Passing Control to a Utility** - Use one of the following forms of command lines to invoke a utility and pass control to it:

- For MCR:

  ```
  >utilityname<CR>
  ```

- For DCL:

  ```
  >RUN $utilityname<CR>
  ```

  or

  ```
  >MCR utilityname<CR>
  ```

These commands do not execute a function; rather, they make a utility available for execution of more than one function without returning control to MCR or DCL. When invoked using one of these forms, the utility responds with the prompt:

```
utilityname>
```

You may then enter the command string that specifies the function you want executed. For example, if you are executing a PIP function, PIP displays the prompt:

```
PIP>
```

To terminate the utility and return to MCR or DCL, type CTRL/Z.

1.4.3 Invoking Uninstalled Utilities

There are two methods for invoking uninstalled utilities. These methods are useful for smaller systems in which not all utilities are installed. Both methods use either the MCR command RUN or the DCL command RUN (depending on which monitor you are using) to invoke the utility.

The first method invokes the utility by means of the following command:

```
>RUN $utilityname<CR>
```

RUN is the MCR command RUN or the DCL command RUN; the dollar sign ($) directs MCR or DCL to search the system directory for the utility and to bring it into storage. On RSX-11M-PLUS, if the utility is not in the system directory, MCR or DCL then searches in the library directory and invokes the utility from there.

When the utility gains control, it displays the prompt:

```
utilityname>
```

Then it waits for you to enter a command line. The utility continues to prompt you after each command is executed. To terminate the utility, enter CTRL/Z.

The second method for invoking an uninstalled utility is the same as the first except that it allows the utility to run under a UIC other than the current UIC:

- For MCR:

  ```
  >RUN $utilityname/UIC=[group,member]<CR>
  ```

- For DCL:

  ```
  >RUN/UIC:[group,member] $utilityname<CR>
  ```

When the utility gains control, it prompts for functions to execute until you enter CTRL/Z.

1.4.4 Using Indirect Command Files

An indirect command file contains a sequence of command lines that can be interpreted by a single task (usually a system-supplied task such as a utility, the MACRO-11 assembler, or the Task Builder). These command lines appear in the indirect command file exactly as you would enter them from your terminal.

The commands contained in the indirect command file are executed when the indirect command file is invoked. For example, an indirect command file might contain a series of PIP command lines. To invoke such an indirect command file, enter one of the following sets of commands:

- For MCR:

```
>PIP @PIPCMDS.CMD<CR>
```

- For DCL:

```
>RUN $PIP<CR>
PIP>@PIPCMDS<CR>
```

or

```
>MCR PIP @PIPCMDS<CR>
```

or

```
>MCR PIP<CR>
PIP>@PIPCMDS<CR>
```

In this example, PIP is invoked and accesses the file PIPCMDS.CMD, which contains the sequence of PIP commands. PIP executes the commands and returns control to MCR, DCL, or PIP, depending on which commands you use.

RSX-11M and RSX-11M-PLUS also allow you to use indirect command files that contain MCR commands. The indirect command file contains both the commands for invoking the utility and the commands that you want the utility to execute. An indirect command file can contain command lines for more than one utility.

You invoke the indirect command file by entering only the file specification preceded by the at sign (@) in response to the MCR prompt:

```
>@indirectcommandfile<CR>
```

The default values for indirect command file specifications are:

- Device -- SY0:
- [group,member] -- the current UIC
- File name -- no default; must be specified
- File type -- .CMD
- Version -- the latest version of the file

For complete information on how to use MCR indirect command files, refer to the RSX-11M/M-PLUS MCR Operations Manual.

PART 2
EDITING UTILITIES

CHAPTER 2

DEC EDITOR (EDT)

The DEC Editor (EDT) provides a means of creating and editing text files. While editing a file using EDT, you always have access to the entire file. It is not necessary to deal with the file in pages or buffers of fixed size.

Within EDT, there are two basic modes of operation: Command Mode and Character Mode.

- Command Mode allows you to use EDT as a line-oriented editor. In Command Mode, each line has a line number assigned by EDT. By issuing commands that refer to these numbers or to character strings within a line, you can manipulate lines or groups of lines. You can also replace character strings within a line or a group of lines. While in Command Mode, EDT maintains a line pointer that points to one line, called the current line.

- Character Mode can only be used on video terminals. It allows you to maneuver the cursor from character to character within a line, or from line to line within a file. Using the cursor as a pointer, you can manipulate individual characters, words, lines, and groups of lines.

You gain access to Character Mode through the Change command, which you issue at Command Mode. It is not possible to go directly from the monitor to Character Mode without passing through Command Mode. In this respect, Character Mode can be viewed as one function of Command Mode. However, you should treat Character Mode as a separate editor from Command Mode. Character Mode and Command Mode complement each other to provide a complete editing system.

EDT offers several other features. Among them:

- **Main and alternate text buffers.** A buffer is a text storage area. The editing commands that you issue affect text in buffers; they do not affect files directly. By default, EDT maintains a single text buffer, called the main buffer. However, you can create alternate buffers to contain text which, for some reason, you wish to hold apart from the main body of text. EDT commands allow you to move lines and groups of lines back and forth between buffers.

- **Editing session backup commands.** You can save the current state of your editing session, including alternate buffer contents and line pointer location, in a specially formatted file. At a later time, you can restore the state of your editing session and take up exactly where you left off. This feature allows you to pause during a complicated editing session and also to protect yourself against the accidental loss of text generated during the session.

- **File and buffer I/O commands.** While working in Command Mode, you can create a file from a text buffer or part of a text buffer. You can also copy the contents of an existing file into a text buffer. This feature allows you to incorporate material from several different files into one buffer during a single editing session.

This chapter is organized into six sections:

- Section 2.1 describes the procedures used to invoke EDT, and discusses EDT input and output files.
- Section 2.2 describes the elements of an EDT Command Mode command string, provides a functionally organized list of EDT commands, and explains range and option specification in EDT.
- Section 2.3 lists Command Mode commands alphabetically, and provides a detailed description of each.
- Section 2.4 contains a complete guide to EDT's Character Mode, including the special Character Mode subcommands.
- Section 2.5 contains a summary, in tabular form, of the EDT Command Mode commands.
- Finally, Section 2.6 lists EDT error messages, and provides explanations and corrective actions for them.

2.1 INVOKING EDT

You can use most of the methods described in Chapter 1 to invoke EDT. However, there is one major difference, which occurs if you invoke EDT using the form

```
>utilityname commandstring
```

In this situation, EDT replies with its Command Mode prompt (*). Most utilities, when invoked in this manner, perform the function specified in the command string and return to the monitor.

To invoke EDT, follow one of these two sequences:

```
>EDT [outfile=][infile]<RET>
*
```

or

```
>EDT<RET>
EDT>[outfile=][infile]<RET>
*
```

outfile

Specifies the output file that EDT creates at the end of the editing session. When you exit EDT at the end of the session, EDT creates the output file and fills it with the contents of the main text buffer.

If you do not specify the output file, the output file name and type default to the input file name and type. The output file version number defaults to the version number of the input file, plus one.

infile

Specifies the file that EDT uses as an input file; or, if infile names a file that does not exist, sets the default file name and type for the output file that EDT creates when you exit at the end of the editing session. If you specify an existing input file, EDT loads its main text buffer from that file. If you do not specify an input file, or specify an input file that does not exist, EDT does not load the main text buffer; this means that you begin the editing session with an empty buffer.

*

EDT's Command Mode prompt, indicating that EDT is in Command Mode and ready to accept editing commands.

Note that you do not have to specify either an input file or an output file. If you specify neither one, you begin the EDT session with an empty main text buffer; furthermore, you must specify an output file when exiting EDT if you wish to save the text generated during the session. If you specify an output file only, you begin the session with an empty buffer, but you will not need to specify the output file when you exit. For example:

```
>EDT BILBO.TXT=
*
```

or

```
>EDT
EDT>BILBO.TXT=
*
```

both cause EDT to create the output file BILBO.TXT upon exit. (This example illustrates two principal methods of invoking EDT; the two methods are equivalent.) By default, the version number of the output file is 1, if the file did not exist previously; or, if the file already exists, n+1, where n is the highest current version number. (See Table 2-1 for a complete list of EDT file specification defaults.)

If you specify only the input file, there are two possibilities: the input file that you specify exists, or it does not exist. If the file exists, EDT loads its main text buffer from the contents of the file and also sets the default output file to match the input file, with a version number one higher than that of the input file. If the input file does not exist, EDT does not load the text buffer; it simply sets the default output file to match the input file. For example:

```
>EDT FRODO.TXT
*
```

is equivalent to

```
>EDT FRODO.TXT=FRODO.TXT
```

if the file FRODO.TXT currently exists, and equivalent to

```
>EDT FRODO.TXT=
```

if the file FRODO.TXT does not currently exist.

Table 2-1
EDT File Specification Defaults

Element	Default Value for Input File	Default Value for Output File[1]
dev:	SY0:	Same as input device
[ufd]	UFD under which EDT is currently running	Same as input [ufd]
filename	No default--must be specified	Same as input file name
.filetype	Null	Same as input file type
;version	Latest version	Latest version+1

[1] Output file defaults take effect only when no output file is specified. If you specify an output file name and type, the output file device and UFD default to your current SY0: and UFD, even if the device and/or UFD of the input file are different.

In the normal situation of editing an existing file with the intention of creating a new version of the file, you need only specify an input file. If you are creating a new file, you also need only specify an input file in order to set EDT's default output file. Note, however, that EDT does not explicitly distinguish between these two situations; you do not receive a message that tells you whether your input file existed or not. It is possible to mistype the name of an existing file and then begin your editing session in the belief that EDT has loaded the text buffer. You usually find out that the buffer is empty as soon as you issue an editing command. In these cases, issue the Quit command to leave EDT without creating an output file, and start over again.

2.2 THE EDT COMMAND STRING

EDT makes changes to the text in response to commands that you enter at your terminal. You can issue commands only when you receive the asterisk prompt that indicates Command Mode.

When you enter a command, you have to specify the action you want EDT to take, and you have to specify the buffer and buffer portion you want to affect. The buffer and buffer portion, taken together, are called the range. Finally, you can modify the action of the command by specifying an option or options to the command. Thus, the entire command string can consist of:

- Command name
- Range specification
- Options
- Carriage return

The command name tells EDT which command to execute. The range specification tells EDT which buffer or buffer portion is to be affected by the command. The options let you control the command execution or specify actions for EDT to perform during execution.

In one command or another, all of the above command string components are optional except the carriage return. Most commands provide a default range specification if you do not specify a range explicitly. If you do not specify a command explicitly, EDT defaults to the Type command. However, EDT takes no action on any command string until you terminate it with a carriage return.

2.2.1 EDT Commands

EDT commands are English words that tell EDT what action to take. In each case, the full command name suggests the intended action. However, you need not type the full command name; you can type an abbreviation for the command. Most of these command abbreviations are one or two letters long.

Table 2-2 lists the commands by category; a brief description of each category follows. Section 2.3 contains complete descriptions of each command, presented in alphabetical order.

Table 2-2
EDT Commands

Command	Function	Section Reference
Commands to Locate and Display Lines		
Find	Repositions the line pointer to the specified line	2.3.5
Type	Displays lines of the text buffer on the terminal	2.3.18
Commands to Input and Modify Text		
Copy	Copies one or more lines from one buffer location to another	2.3.2
Move	Moves one or more lines from one buffer location to another, and deletes the lines from the original location	2.3.8
Delete	Deletes one or more lines from the text buffer	2.3.3
Insert	Inserts terminal input into the text buffer	2.3.7
Replace	Deletes one or more lines, and replaces the deleted lines with terminal input	2.3.11
Substitute	Changes characters within lines of the text buffer	2.3.17

(continued on next page)

Table 2-2 (Cont.)
EDT Commands

Command	Function	Section Reference
	Commands to Leave EDT	
Exit	Outputs the main text buffer contents to a file, and then returns to the monitor	2.3.4
Quit	Returns to the monitor without creating an output file	2.3.10
	Commands to Back Up and Restore Editing Sessions	
Save	Creates a file containing the contents of all text buffers currently in use	2.3.14
Restore	Recreates all text buffer contents from a temporary file created by the Save command	2.3.13
	Commands to Output To and Input From Files	
Write	Creates a file from text buffer contents	2.3.19
Print	Creates a file from text buffer contents, including the line numbers that EDT assigns	2.3.9
Include	Copies a file into a text buffer	2.3.6
	Commands to Establish and Display Parameters	
Set	Specifies display and match criteria to be used by EDT for character searches	2.3.15
Show	Displays current match and display criteria and buffer status	2.3.16
	Miscellaneous Commands	
Change	Invokes Character Mode	2.3.1
Resequence	Renumbers the lines in the text buffer	2.3.12
XEQ (Execute)	Executes a group of previously entered EDT commands	2.3.20

Commands to Locate and Display Lines

The two commands in this group both cause EDT's line pointer to point to a new line, but they do not modify any text. The Find command positions the line pointer at the line that you specify. You can specify the line in any of a variety of ways, as explained in Section 2.2.2. The Type command also repositions the line pointer, but Type additionally prints the line or lines you specify on your terminal.

Commands to Input and Modify Text

Use the commands in this group to modify existing text or create new text. All of these commands, with the exception of Substitute, work on a line basis; that is, the smallest unit of text they can handle is a complete line.

The Copy and Move commands are similar in that they direct EDT to move a line or group of lines from one location to another. However, the Copy command does not delete the lines from the original location, while the Move command does. Both the Copy and Move commands can affect the position of the line pointer.

The Copy and Move operations can take place within a buffer or between buffers; in fact, it is possible to Copy or Move an entire buffer to a specified location within another buffer. This is useful in cases where you wish to take text from a file, modify it, and insert it into your main text buffer. Typically, you might first use the Include command to place the file into an alternate text buffer; then issue various editing commands to modify the text in the alternate buffer; and finally Copy the text to the desired location in the main buffer.

The Delete command allows you to delete a single line of text; a block of lines in a text buffer; all lines in a text buffer that contain a specified character string; or an entire buffer.

The Insert and Replace commands are similar in that both allow you to insert one or more lines of text in a text buffer. However, Replace deletes the line or lines that you specify before you insert the text; in other words, Replace replaces specified lines with text that you supply. Insert and Replace are unusual in that they do not return directly to the Command Mode asterisk prompt when you enter them. Instead, there is no prompt; EDT waits for you to supply input. You then insert one or more lines of text until you type a <RET> at the end of a line, followed by a <CTRL/Z>. At this point, EDT recognizes that the insert is finished and returns the asterisk prompt; you can then issue more Command Mode commands.

The Substitute command is the only Command Mode command that can affect text within a line, instead of affecting the entire line. It allows you to replace a specified character string with another character string. The strings need not match in length. You can replace a string in a single line; in each of a group of lines; or in all lines in a buffer. If you wish, you can substitute selectively: you specify a substitution to be made in all lines, and EDT prompts you at each substitution opportunity, allowing you to accept or reject each substitution as appropriate.

All the commands in this group can affect the position of the line pointer as part of their function. See the individual command descriptions in Section 2.3 for detailed information.

Commands to Leave EDT

The Exit and Quit commands both direct EDT to terminate the current editing session and return to the monitor. The Quit command does this without creating an output file; in other words, EDT does not output the contents of its main text buffer into a file when you issue the Quit command. Note, however, that you can direct EDT to create a file during the editing session by issuing a Write command.

The Exit command terminates the editing session and does direct EDT to output the contents of its main text buffer into an output file. There are two ways to specify the name of this file:

- When you first invoke EDT, you can specify the output file as described in Section 2.1, that is, explicitly, or implicitly, by specifying only the input file.

- When you issue the Exit command to leave EDT, you can use the /Rename option to specify the output file. The output file specification that you provide with the /Rename option overrides the output file specification that you provide when you invoke EDT. If you specify neither an input file nor an output file when you invoke EDT, you must use the /Rename option with the Exit command; otherwise, EDT cannot create an output file because it has no output file specification available.

Note that the Exit command automatically outputs the entire contents of the main text buffer to the output file. You cannot specify that an alternate text buffer be output instead of the main text buffer, and you cannot specify that only a portion of the main buffer be output. If you wish to create a file from a buffer other than the main buffer, you can use the Write command. The Write command provides greater flexibility than the Exit command, because it allows you to specify the buffer, and lines within the buffer, to be output.

Commands to Back Up and Restore Sessions

The Save and Restore commands work as a pair to provide a means of backing up the work that you do during an editing session. You can issue the Save command at any time during an editing session. This directs EDT to create a file containing the following information:

- Contents of main and alternate text buffers
- Current line pointer location
- Input and/or output file specifications
- EDT parameters

After EDT creates the file, you can continue your editing session as if you had never paused to issue the command. If you want, you can create an output file by issuing the Exit command. However, at a later time, you might want to recreate the editing session as it was when you issued the Save command. To do this, invoke EDT without specifying either an input or an output file, and issue the Restore command, specifying the same file that you specified when you issued the Save command. EDT will reconstruct its buffers, its line pointer, its file specifications, and its parameters from the contents of the file that you specify. It will be as if you had never left the Saved session.

There are two principal uses for the Save and Restore commands:

- In the middle of a long, complicated editing project involving several text buffers, you can use the Save command to save all text buffers. This allows you to spread your work over several sessions.

- If you fear the accidental loss of text that you have generated during a session, you can use the Save command to create a backup file at any time during the session. Later, after you have issued the Exit command and successfully created the output file, you can delete these backup files.

The file that EDT creates as a result of a Save command is in a special format and is useful only as input to EDT's Restore command. For this reason, you should be sure to provide a file specification with the Save command that is different from your output file specification. Typically, you might want to reserve a special file type to be used only with the Save command.

Commands to Output To and Input From Files

The Write and Print commands allow you to create an output file while your editing session is in progress, in contrast to the Exit command, which creates the output file and then returns control to the monitor. Additionally, Write and Print offer you more flexibility in creating an output file than the Exit command, since Write and Print allow you to specify the text buffer and the lines within the buffer that you want to output.

The Write and Print commands differ in that Print creates a file that contains the line numbers assigned by EDT, while Write creates a file without the line numbers. The format of a file created by the Write command is identical to that of a file created by the Exit command.

The Include command allows you to bring the contents of a file into a text buffer during an editing session. You can insert the file's contents into a buffer that already contains text, or you can create a new buffer to hold the file's contents. The Include command is particularly useful in situations where you need to prepare a file using parts of several existing files.

Commands to Establish and Display Parameters

The Set and Show commands work as a pair to provide a means of examining and altering several EDT parameters. These parameters include:

- **String match criteria.** When searching for strings, EDT can require that both the characters and the cases of the characters match, or that only the characters match. The strings ABC and abc are a match in the second instance, but not in the first.

- **Case display criteria.** EDT can flag either upper- or lower-case characters with an apostrophe while displaying text, or can display text "as is." If you are working at a terminal that displays only upper-case characters, you may want EDT to flag those characters that are upper case in the text buffer, in order to distinguish them from lower-case characters.

- **Terminal type.** When EDT is in Character Mode, it requires information about what kind of terminal is in use. Before using Character Mode, you should be sure that the terminal type is set correctly.

You can use the Set and Show commands to establish and examine each of these parameters. In addition, you can use Show to display information about the current state of EDT's text buffers and the version of EDT in use.

Miscellaneous Commands

The Change command invokes EDT's Character Mode. You should only issue it from a video terminal. Although many Character Mode operations are analogous to the Command Mode operations discussed in this section, you should treat Character Mode as a separate editor. Section 2.4 contains a complete description of Character Mode.

The Resequence command directs EDT to renumber the lines in a specified text buffer. EDT initially numbers the lines in a text buffer by tens; that is, the first line is line 10, the second line is line 20, and so on. Following any one of several text modification commands, there may not be enough line numbers available to number all the lines in an area of the buffer. For example, if you insert 15 lines between lines 10 and 20, there will be six lines without line numbers. By issuing a Resequence command, you can restore orderly line numbering to the buffer.

The Resequence operation is available to several Command Mode commands as an option. See the individual command descriptions in Section 2.3 for details.

The Execute command directs EDT to execute a series of commands that you have previously placed in a text buffer. The Execute command is particularly useful when you need to repeat a series of operations several times.

2.2.2 Range Specification

When you issue an EDT editing command, you must also specify what portion of the text buffer you wish that command to affect. The portion of the buffer is called the range; the information that you give EDT to allow it to find the range is called the range specification.

A range can be as small as a single line, or as large as an entire buffer. It can consist of a contiguous group of lines within a buffer, or all lines in a buffer that contain a certain string.

Most commands allow you to omit an explicit range specification. These commands provide default range specifications when you do not specify the range explicitly. The individual command descriptions in Section 2.3 provide details.

A complete range specification contains two broad elements:

1. **A buffer specification.** The buffer specification tells EDT which buffer contains the range. The buffer can be the main text buffer or one of the alternate text buffers.

2. **A line specification.** The line specification tells EDT which lines within the buffer make up the range. There are four general types of line specification:

   - **Single line specifications.** Single line specifications allow you to specify one line of text.
   - **Variable line specifications.** Variable line specifications allow you to specify an indeterminate number of contiguous lines. Examples are: all lines in the current text buffer, all lines between the current line in the text buffer and the end of the buffer.
   - **Compound line specifications.** Compound line specifications combine single line specifications, operators, and integers to allow you to strictly define groups of lines within the buffer. Examples are: all the lines between two specified lines, a group consisting of a specified number of lines following a specified line.
   - **Inclusive line specifications.** Inclusive line specifications allow you to specify a string that each line in the range must contain. The range consists of all lines containing the specified string.

   You can combine these different types of line specifications to produce the desired effect. For example, by combining compound and inclusive line specifications, you can specify a range consisting of all lines between lines 70 and 180 that contain the string ABC.

The remainder of this section discusses each of the elements that make up a range specification. Also presented are three concepts common to all forms of line specification:

- The line pointer
- Line numbers and line sequencing
- String searches

2.2.2.1 **Buffer Specification** - When you invoke EDT, EDT creates a text buffer, or storage area, called MAIN. If you specify the name of an existing file as an input file when you invoke EDT, EDT copies the contents of that file into the main text buffer. However, if you specify the name of a new file, EDT creates MAIN with no data stored in it. EDT performs the editing commands you issue on the contents of the text buffer. Commands can add to or take away from the contents of a buffer, move text from one buffer to another, or simply change EDT's position within a buffer.

If you are working your way through an EDT text buffer and you reach the end of the buffer, EDT prints the following message:

```
[EOB]
```

In effect, the [EOB] designation is a line that follows the final text line of the buffer and tells you that you have passed that final line. Since EDT inserts lines in front of a specified line, you must specify the [EOB] line if you want to insert text at the end of the buffer. (See Section 2.2.2.5 for a discussion of single line range specifications, and Section 2.3.7 for a description of the Insert command.)

EDT can create text buffers other than MAIN. For example, you can access other files using the Include command to copy their contents into an alternate text buffer. EDT allows you to name the alternate text buffers you use, and to make edits to the buffer contents in the same way you edit the main text buffer.

There is no limit to the size of text buffers other than the limits of the system that you are using; however, EDT does not assign line numbers above 65535.

EDT creates a text buffer when you give it a name and then use the buffer name as part of a command. Buffer names can be up to six alphanumeric characters long. Each time you refer to a buffer, you must precede the buffer name with either an equal sign (=) or a %BUF notation. Thus, the following are all legal buffer references:

```
%BUF MAIN
=MAIN
%BUF ALT1
=ALT1
%BUF lOSNE1
=lOSNE1
```

The first two examples above both refer to the main text buffer.

The number of text buffers available during an editing session is determined when EDT is installed in your system. You will always be able to use the main buffer and at least two alternate buffers, but you should check with your system manager to find out the maximum number of buffers available on your system.

EDT does not save the contents of any alternate buffers when you terminate the editing session. You can save the contents of alternate text buffers either by using the Write command to directly create a file from the buffer, or by using the Move or Copy command to place the alternate buffer's contents in the main text buffer.

EDT maintains an internal record of which buffer is currently in use. This buffer is called the current buffer. A buffer becomes current when an EDT command moves the line pointer into the buffer and leaves it there. (Section 2.2.2.2 describes the line pointer; the command descriptions in Section 2.3 provide information on each command's effect on the line pointer.) You can use the Show command (Section 2.3.16) to find out which buffer is the current buffer.

The current buffer can serve as an implied range specifier. Several commands take the current buffer as their range argument if you do not specify range explicitly.

2.2.2.2 **The Line Pointer** - The line pointer is an internal EDT mechanism that keeps track of EDT's position within text buffers. As you issue commands, EDT moves from line to line and from buffer to buffer, using the line pointer to keep track of its position. The line pointer does not point at any part of the line; rather it indicates an entire line. The current position of the line pointer (the current line) can be displayed by typing a period followed by a carriage return in response to the Command Level asterisk prompt. A carriage return immediately following the asterisk prompt advances the line pointer to the next line and displays that line.

The position of the line pointer changes as EDT executes commands. Each command's effect on the line pointer is documented in the detailed command descriptions in Section 2.3.

The line pointer can serve as an implied range specifier. Several commands take the current line as the range argument if you do not specify a range explicitly.

2.2.2.3 **Line Numbers and Line Number Sequencing** - EDT automatically assigns a line number to each line in a text buffer to help you locate and reference the lines in the buffer. Line numbers are integers in ascending order from 1 to 65535. They appear at the left margin of the terminal display, separated from the text by a tab. Line numbers can be changed or completely removed from the buffer, but are not part of the text.

By default, EDT assigns 10 as the number of the first line and increments by 10 for each additional line. Thus, the default system line numbers are 10, 20, 30, 40, and so on. However, you can use the Resequence command (Section 2.3.12) to override this default and specify the line numbers you want assigned.

Many commands that transpose, insert, or delete lines have an option, /Sequence, that allows you to specify numbers for the transferred or inserted text. Often the line numbers that result after one of these commands have varying increments between them, and some lines may not have numbers at all. If this occurs, you can use the Resequence command to reassign uniform line number increments to all the lines in your buffer.

The Move, Replace, Include, and Insert commands, among others, put lines into text buffers. The following example shows how EDT assigns line numbers to new lines in buffers. If the lines in the buffer are numbered 10, 20, 30, and so on up to 80, and if 12 lines are inserted between lines 50 and 60, EDT assigns line numbers to the group as shown in Figure 2-1.

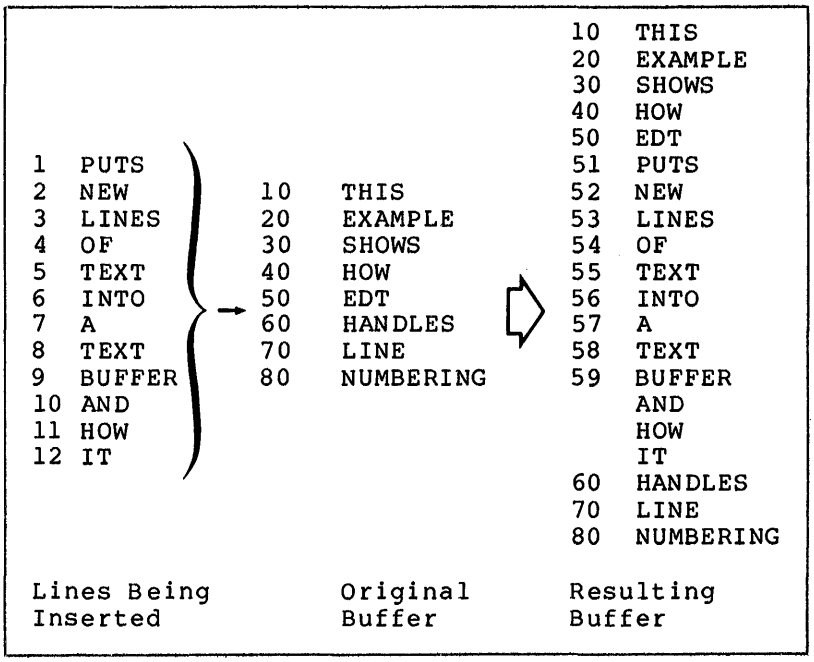


Figure 2-1 Input Line Number Sequencing

As Figure 2-1 shows, EDT numbers as many of the new lines as possible without reassigning or changing any line numbers already in use. EDT then puts the remaining lines in their proper position without assigning them any numbers.

The Move, Replace, and Delete commands take lines out of text buffers. When you use these commands, EDT removes the lines you specify, as well as their line numbers, from the buffer. The numbers of the lines that remain in the text buffer do not change. If a line number is no longer assigned to a line, it is available for either EDT or you to assign to a new line or to reassign to an existing line.

2.2.2.4 **String Searches** - Line numbers are not the only way for you to specify a line in a buffer. You can direct EDT to search the buffer contents for a string, which consists of a specific word or combination of letters. When you specify such a character string, you give EDT an object string to search for, and EDT examines the text buffer looking for an equivalent or match string.

In its search, EDT normally examines the current line first to see if it contains a match of the object string. If no match is found in the current line, EDT then examines the next line, and the next, and so on until it either finds a match or reaches the end of the buffer.

An exception to this arises if EDT located the current line through a previous string search or through a Substitute command (see Section 2.3.17). In these cases, the search proceeds from a point in the line immediately following the string that was searched or substituted for.

For example, you search for the string ILLUSTRATES, and EDT locates the line:

```
120     THIS LINE ILLUSTRATES STRING SEARCHES
```

You can now successfully search forward for STRING or SEARCHES, but not for THIS, LINE, or ILLUSTRATES. You could, however, successfully search backward for THIS, LINE, or ILLUSTRATES.

This mechanism allows you to search through a file for repeated occurrences of a string without having to move the line pointer forward each time you search.

Once you determine the object string and the action you want to perform, you must:

- Specify the string
- Dictate the direction of the search
- Determine what constitutes a match

EDT treats characters enclosed in apostrophes or quotation marks as object strings. If you enclose a character or group of characters in two apostrophes (') or two quotation marks ("), you direct EDT to search for the first occurrence of an identical string of characters (without the delimiters) contained in the text buffer. Apostrophes and quotation marks are the only character string delimiters that EDT accepts as part of a range specification.

The way you specify the object string determines the direction of EDT's search. If you enter "ABC", EDT searches from the current line toward the bottom of the buffer looking for the first occurrence of

ABC. If you precede the object string with a minus sign (-), that is, -"ABC", EDT searches from the current line toward the top of the buffer looking for the first occurrence of ABC.

EDT lets you specify whether or not a match occurs when the case of the characters in the match string differs from the case of the characters in the object string. For example, if you are searching your text buffer for an occurrence of BASIC, do you want EDT to return a match when it encounters the string basic?

The Set command (see Section 2.3.15) allows you to specify whether match strings must correspond to the cases specified in the object string. If you issue the Set Exact Case command, EDT returns as a match only those strings that are identical in both case and content to the object string. If you issue the Set Exact None command, EDT returns any occurrence of the characters in the object string as a match, regardless of case. Exact None is the default setting.

2.2.2.5 **Single Line Specifications** - Single line specifications identify one line of a text buffer. Single line specifications and the lines they specify are listed in Table 2-3. Square brackets enclose optional portions of the specifications.

All of the single line specifications, with the exception of %L, can be preceded by a buffer specification (see Section 2.2.2.1). A space must separate the buffer specification from the line specification. If you do not provide a buffer specification, EDT locates the single line in the current text buffer.

Table 2-3
Single Line Specifications

Line Specification	Meaning
nn	Line number nn
. (period)	The current line
"object string" or 'object string'	Starting with the current line, the first line located that contains an acceptable match of the object string
-"object string" or -'object string'	The first preceding line that contains acceptable match of the object string
%BE[GIN]	The first line of the text buffer
%E[ND]	The [EOB] designation that follows the last line of the text buffer
%L[AST]	The line in a previous text buffer at which the line pointer was positioned when the command to enter the current buffer was issued

Examples

```
*40
```

Locates and displays line 40 in the current text buffer. If line 40 currently does not exist, locates and displays the first line with a number higher than 40.

```
*=X 130
```

Locates and displays line 130 in buffer X.

```
*'THIS HOUSE'
```

Locates and displays the first line of the current buffer that contains the string THIS HOUSE.

```
*=NEWBUF "FOR SALE"
```

Locates and displays the first line in buffer NEWBUF containing the string FOR SALE.

```
*"FIND THIS STRING'
Missing string quote.
```

In this example, the string delimiters were mismatched; EDT returned an error message.

```
*%BE
```

Locates and displays the first line in the current text buffer.

```
*=MORDOR %E
```

Locates and displays the [EOB] designation that follows the last line in buffer MORDOR.

```
*=MAIN %L
Buffer Specification Conflict.
```

In this example, %L specifies a line in a previous text buffer. Since %L already provides a text buffer specification, you cannot explicitly specify a buffer with %L. EDT returned an error message indicating a conflict in buffer specifications.

2.2.2.6 **Variable Line Specifications** - Variable line specifications identify an indeterminate number of lines in the text buffer. The specifications and the lines they identify are listed in Table 2-4. Square brackets enclose optional portions of the specifications.

As indicated in Table 2-4, a buffer specification used by itself is a variable line specification, because it implies all lines in the buffer. The remaining variable line specifications (%BEF, %R, and

%WH) imply the current text buffer; therefore, you cannot use them in combination with an explicit buffer specification. If you attempt to do so, EDT returns the error message:

Buffer Specification Conflict.

Table 2-4
Variable Line Specifications

Line Specification	Meaning
=bufname or %BUF[FER] bufname	All lines contained in the text buffer bufname
%BEF[ORE]	All lines in the current text buffer from the first line in the buffer through the current line
%R[EST]	All lines in the current text buffer from the current line through the last line in the buffer
%WH[OLE]	All lines in the current text buffer

2.2.2.7 **Compound Line Specifications** - Compound line specifications identify a specific number of lines in the buffer and consist of single line specifications, operators, and integers. Table 2-5 lists the forms that compound line specifications may take. In the table, sl stands for any of the single line specifications from Table 2-3.

Table 2-5
Compound Line Specifications

Line Specification	Meaning
sl1:sl2 or sl1 %THRU sl2	The lines between and including the first single line (sl1) and the second single line (sl2) where sl1 must precede sl2
sl;i or sl %FOR i	The total number of lines specified by the positive integer i that begins with the line specified by the single line sl
sl1,sl2,sl3,... or sl1 %AND sl2 %AND sl3	The individual lines identified by the single line specifications
sl+i	The single line that is i lines after the single line specified by sl
sl-i	The single line that is i lines before the single line specified by sl

Any of the compound line specifications can be preceded by a buffer specification (see Section 2.2.2.1). A space must separate the buffer specification from the line specification. If you do not provide a buffer specification, EDT locates the lines in the current buffer.

Examples

```
*40:80
```

Locates and displays lines 40 through 80 of the current text buffer.

```
*=ALT1 'ABC' %THRU %E
```

Locates and displays the block of lines beginning with the first line in buffer ALT1 that contains the string ABC, and ending with the last line in the buffer.

```
*150;20
```

Locates and displays a block of lines in the current text buffer that begins with line 150 and extends for 20 lines.

```
*=MAIN "STARTLINE" %FOR 35
```

Locates and displays a block of lines in the main text buffer that begins with the first line that contains the string STARTLINE and extends for 35 lines.

```
*%BUF GHORT 10,30,'KLATAAU',%E
```

Locates and displays the following lines from buffer GHORT: line 10, line 30, the first line following line 30 that contains the string KLATAAU, and the [EOB] designation following the last line of the buffer.

```
*"WORLD"+6
```

Locates and displays the sixth line following the line that contains the string WORLD in the current buffer.

2.2.2.8 **Inclusive Line Specifications** - The inclusive line specifications identify lines of a buffer that contain a specific string. You can find all lines in the buffer that contain the string, or you can limit the lines that are checked for the string by qualifying the inclusive line specification with another line specification. Table 2-6 illustrates inclusive line specifications.

You can precede an inclusive line specification with a buffer specification (see Section 2.2.2.1). A space must separate the buffer specification from the line specification. If you do not include a buffer specification, EDT locates the lines in the current buffer.

Table 2-6
Inclusive Line Specifications

Line Specification	Meaning
%ALL 'string' or %ALL "string"	All lines in the text buffer that contain acceptable matches of the specified object string
ls %ALL 'string' or ls %ALL "string"	All lines that contain acceptable matches of the object string within the line specification ls

Examples

```
*%ALL 'VAN GOGH'
```

Locates and displays all lines containing the string VAN GOGH in the current text buffer.

```
*%R %ALL 'MONET'
```

Locates and displays all lines in the current text buffer between the current line and the end of the buffer that contain the string MONET.

```
*=PAINT 50:300 %ALL 'PICASSO'
```

Locates and displays all lines between lines 50 and 300 of buffer PAINT that contain the string PICASSO.

2.2.3 Options

Command line options either allow you to control the command execution or specify actions that EDT is to take when the execution has been completed. Each detailed command description in Section 2.3 includes a description of each option that is allowed with the command. Table 2-7 shows options with their functions and associated commands.

Each option has an abbreviation that you can use instead of the full option name. Table 2-7 shows the abbreviation outside of brackets; the remainder of the option name is inside the brackets. When you specify an option, you can use either the abbreviation or the full name.

Table 2-7
Command Options

Option	Meaning	Commands
/BR[IEF]	Displays first ten characters of lines in range	Type, Substitute
/FI[LE]	Specifies a file that is either accessed or created	Write, Save, Include, Print, Restore
/NL	Allows deletion of line terminators	Change
/Q[UERY]	Prompts you to control execution of command	Substitute, Move, Copy, Delete
/RE[NAME]	Specifies an output file	Exit
/SEQ[UENCE]	Specifies sequence numbers	Include, Insert, Copy, Move, Replace, Resequence
/-T[YPE]	Inhibits display of lines affected by the command	Substitute
/UN[SEQUENCED]	Inhibits renumbering of lines	Include, Insert, Copy Move, Replace, Resequence

2.3 DETAILED COMMAND DESCRIPTIONS

This section lists the EDT Command Mode commands alphabetically. Each command description includes the format of the command, a description of each of the options available with the command, examples of the command in typical editing situations, and usage notes.

In the command format descriptions, the command abbreviation is shown outside of brackets, while the remainder of the command name is shown in brackets. When issuing a command, you can use either the abbreviation or the full command name. Thus,

```
REST[ORE]
```

in the format description for the Restore command indicates that you can type either REST or RESTORE when you issue the command.

2.3.1 CHANGE

Use the Change command to invoke EDT's Character Mode. (Section 2.4 contains a complete description of Character Mode.)

Format

C[HANGE] [range] [/NL]

CHANGE

Specifies the Change command.

range

Specifies the lines that you will be able to access while in Character Mode. If you do not specify a range, the entire contents of the current text buffer can be accessed.

/NL

Specifies that, while in Character Mode, you can delete line terminators such as form feeds, line feeds, and carriage returns.

Examples

```
C
```

This command invokes Character Mode and gives you access to all of the current text buffer.

```
C =X 50:200/NL
```

This command invokes Character Mode, giving you access to lines 50 through 200 of the text buffer named X. While in Character Mode, you will be able to delete line terminators.

Notes

- You can use the Change command only if you are using a video display terminal. You must use the Set Terminal command to correctly establish your terminal type before you can edit in Character Mode. Section 2.3.15 contains a description of the Set command.
- While in Character Mode, you only can edit the contents of the range you specified in the Change command. If you want to edit lines that are outside the range, you first must issue an EX (Exit) subcommand and then re-enter the Change command specifying the new range.
- When you invoke Character Mode and specify the /NL option, EDT treats end-of-line characters as single characters that may be inserted or deleted. EDT displays these characters as follows:

  <FF> indicates an ASCII form feed
  <LF> indicates an ASCII line feed
  <VT> indicates an ASCII vertical tab character

Character Mode does not give any special representation to a carriage return, but allows you to delete it as well. When you delete an end-of-line character, EDT concatenates the following line to the line whose end-of-line character was deleted.

If you attempt to delete a line terminator without having specified the /NL option, EDT takes no action in response to the subcommand you enter.

- Two Character Mode subcommands, EX (Exit) and Quit, terminate Character Mode. Typing EX returns EDT to Command Level from Character Mode. Typing EX does not terminate EDT if issued from Character Mode. Typing QUIT terminates both Character Mode and EDT. The Quit subcommand does not create or update any files.

2.3.2 COPY

Use the Copy command to transfer lines from one location to another without deleting them from their original location.

Format

```
CO[PY] range-1 %TO range-2  [/Q[UERY]]
                            [/SEQ[UENCE]:initial-number:increment]
                            [/UN[SEQUENCED]]
```

COPY ... %TO

Specifies the Copy command.

range-1

Specifies the lines that are to be copied. If you specify more than one line in range-1, all the lines you specify are copied.

range-2

Specifies the line ahead of which the lines in range-1 are copied. If you specify more than one line in range-2, the lines in range-1 are copied ahead of the first line in range-2.

/QUERY

Allows you to specify how each line in range-1 is to be treated. Before EDT transfers each line, it prints the line and waits until you type one of the following responses:

Response	Result
Y or YES	Copies the line
N or NO	Does not copy the line
Q or QUIT	Stops copying lines and displays the asterisk prompt
A or ALL	Copies the remaining lines in the range without printing them first

/SEQUENCE:initial-number:increment

Assigns specific line numbers to the copied lines. The initial-number argument specifies the number of the first line that was copied, and the increment argument specifies the increment between numbers. For example, /SEQ:100:100 creates the copied lines with line numbers of 100, 200, 300, 400, and so on.

/UNSEQUENCED

Causes EDT to copy the lines without assigning them line numbers.

Examples

```
CO.:".MAC"%TO=X 40
```

This command copies all lines, beginning with the current line and continuing through the first line containing .MAC, to a position preceding line 40 in buffer X.

```
CO.;20 %TO %BE
```

This command copies 20 lines, starting with the current line, to the beginning of the current text buffer.

Notes

- After a line or group of lines has been copied, EDT positions the line pointer at the first line of the copied lines in their new position preceding the first line in range-2.
- You cannot overlap ranges: range-1 and range-2 cannot contain any of the same lines.
- If you do not use either the /Sequence or /Unsequenced option to specify line numbers, EDT numbers the new lines in the following manner:
  - If EDT can number the new lines by incrementing the existing line numbers in steps of 10, it does so as long as the resulting line numbers do not duplicate existing line numbers or violate the ascending order of line numbers.
  - If incrementing by 10 would result in an illegal line number, EDT begins incrementing subsequent new lines in steps of 1.
  - If incrementing the new lines in steps of 1 would result in illegal or duplicate line numbers, EDT places all further lines in the correct order, but does not assign them any line numbers.

2.3.3 DELETE

Use the Delete command to delete lines from a text buffer.

Format

D[ELETE] [range] [/Q[UERY]]

DELETE

Specifies the Delete command.

range

Specifies a range of lines to be deleted. If you do not specify a range, the current line is deleted.

/QUERY

Specifies that you want to control the Delete operation using EDT prompts and responses. EDT prompts you by displaying the line to be deleted; you can then enter any one of the following responses:

Response	Result
Y or YES	Deletes the prompt line
N or NO	Does not delete the prompt line
Q or QUIT	Stops deleting lines and returns to Command Mode
A or ALL	Deletes all remaining lines in the range; does not display the deleted lines

Examples

```
D 20:40
```

This command deletes lines 20 through 40, inclusive, from the current text buffer.

```
D %ALL 'ABC'/Q
```

This command finds all the lines in the current text buffer containing the string ABC, displays each line, and then waits for you to enter a Y, N, Q, or A response.

```
D =BUF2 %ALL '438'
```

This command deletes all lines containing the string 438 from text buffer BUF2.

Notes

- After a line or a group of lines has been deleted, EDT positions the line pointer at the line immediately following the last line deleted.
- The /Query option allows you to delete specific lines at random from the specified range of lines. After you delete lines using the /Query option, EDT positions the line pointer at the undeleted line nearest the beginning of the range.
- If you enter nonexistent line numbers in a range specification, EDT rejects the specification and displays an error message.

2.3.4 EXIT

Use the Exit command to terminate EDT, return control to the system monitor, and save the contents of the main text buffer.

Format

EX[IT] [/RE[NAME]:filespec]

EXIT

Specifies the Exit command.

/RENAME:filespec

Specifies the output file to which the contents of the main text buffer are written. The /Rename option overrides the output file specified when EDT was invoked. Section 1.3 contains a complete description of file specifications.

Example

```
EX/RE:SYS.MAC
```

This command causes EDT to write the contents of the main text buffer to the file SYS.MAC, then terminate.

Notes

- The Exit command only saves the contents of the text buffer named MAIN. The contents of all other text buffers are lost if they are not incorporated into MAIN or written to permanent files through the Write command. (See Section 2.3.19.)
- The /Rename option allows you to specify an output file whether or not an output file was originally specified in the EDT invocation command.

2.3.5 FIND

Use the Find command to move the line pointer.

Format

F[IND] range

FIND

Specifies the Find command.

range

Specifies the line to which the line pointer is positioned. If you specify a range of more than one line, EDT positions the line pointer at the first line in the range.

Examples

```
F = X 20
```

This command causes EDT to position the line pointer at line number 20 in text buffer X.

```
F-'ABC'
```

This command causes EDT to search upward and position the line pointer at the first line encountered that contains ABC.

```
F 20:40
```

This command causes EDT to position the line pointer at line 20, the first line of the range.

Notes

- The Find command does not generate any terminal output.

2.3.6 INCLUDE

Use the Include command to locate a file and to copy it into a text buffer.

Format

```
                        /FI:filespec
INC[LUDE] [range]       [/SEQ[UENCE]:initial-number:increment]
                        [/UN[SEQUENCED]]
```

INCLUDE

Specifies the Include command.

range

Specifies the line ahead of which the contents of the file are to be inserted. If you specify a range of more than one line, EDT inserts the file ahead of the first line in the range. If you do not specify a range, EDT inserts the file ahead of the current line.

/FI:filespec

Specifies the file that is to be copied. (See Section 1.3 for a complete description of file specifications.)

/SEQUENCE:initial-number:increment

Assigns specific line numbers to the included lines. The initial-number argument speciifes the number assigned to the first line, and the increment argument specifies the increment between numbers. For example, /SEQ:100:100 creates the new lines with line numbers of 100, 200, 300, 400, and so on.

/UNSEQUENCED

Causes EDT to include the lines without assigning line numbers.

Examples

```
INC =ADD /FI:SYS.MAC
```

This command causes EDT to copy the contents of file SYS.MAC into text buffer ADD. If ADD contains some data, EDT inserts the new lines at the beginning of the buffer.

```
INC 'ABC' /SEQ:100:2/FI:EDT.FTN
```

In this example, EDT first copies the contents of file EDT.FTN, then inserts them before the next line in the current buffer that contains the character string ABC, and finally numbers the new lines as 100, 102, 104, 106, and so on.

Notes

- After a file has been included into a text buffer, EDT positions the line pointer at the first line following the newly included text.

- The Include command copies a file into your current text buffer or to an alternate text buffer, allowing access to any portion of that file during your editing session. Once the file is copied, you can use other EDT commands to manipulate its contents.

  For example, to access a portion of a file other than the one you are currently editing, perform the following steps:

  1. Issue the Include command to copy the file into an alternate text buffer.

  2. Issue the Move command to transfer the desired portion of the file from the alternate text buffer to the main buffer.

- If more lines exist in the file being included than there are line numbers available, EDT correctly inserts the extra lines into the text buffer but does not assign them any line numbers.

  If you use the /Sequence option to specify line numbers that conflict with other line numbers in the file, the new lines are included without line numbers.

  If you do not use either the /Sequence or /Unsequenced option to specify line numbers, EDT numbers the new lines in the following manner:

  - If EDT can number the new lines by incrementing the existing line numbers in steps of 10, it does so as long as the resulting line numbers do not duplicate existing line numbers or violate the ascending order of line numbers.

  - If incrementing by 10 would result in an illegal line number, EDT begins incrementing subsequent new lines in steps of 1.

- If incrementing the new lines in steps of 1 would result in illegal or duplicate line numbers, EDT places all further lines in the correct order, but does not assign them any line numbers.

When you use the Include command to insert a file into a text buffer that already contains some data, you may want to use the Resequence command to establish uniform line number increments throughout that buffer.

2.3.7 INSERT

Use the Insert command to place text you type at your terminal into a text buffer.

Format

```
I[NSERT] [range]   [/SEQ[UENCE]:initial-number:increment]
                   [/UN[SEQUENCED]]

text

<CTRL/Z>
```

INSERT

Specifies the Insert command.

range

Specifies the line ahead of which the text you type is to be inserted. If you specify more than one line in the range, the text is inserted ahead of the first line in the range. If you do not specify a range, the text is inserted ahead of the current line.

/SEQUENCE:initial-number:increment

Assigns specific line numbers to the newly inserted lines. The initial-number argument specifies the number assigned to the first new line and the increment argument specifies the increment between numbers. For example, /SEQ:100:100 creates new lines with line numbers of 100, 200, 300, 400, and so on.

/UNSEQUENCED

Causes EDT to insert the new lines without assigning them line numbers.

text

Represents one or more complete lines of text. The last line must end with a carriage return.

<CTRL/Z>

Causes EDT to stop accepting insert text and display the asterisk prompt.

Examples

```
*I <RET>
LINE 1 <RET>
LINE 2 <RET>
LINE 3 <RET>
<CTRL/Z>
*
```

This example illustrates the entire Insert command sequence. The command, I, which is issued while EDT is displaying the asterisk prompt, directs EDT to insert the text that follows above the current line. EDT then accepts three lines of input from the terminal, inserting them above the current line. Finally, <CTRL/Z> causes EDT to terminate the insert and display the asterisk prompt.

```
I =B 40
```

This command causes EDT to insert all the lines you enter ahead of line 40 in buffer B.

```
I ".OR"
```

This command causes EDT to insert the lines you enter ahead of the next line encountered that contains the string .OR.

```
I %E
```

This command causes EDT to insert the lines you enter at the end of the buffer (that is, between the last line of the buffer and the [EOB] designation).

Notes

- After a line or group of lines has been inserted into a text buffer, EDT positions the line pointer at the line following the last inserted line.
- When you issue an Insert command, EDT first positions the line pointer ahead of the first line in the range. Without prompting you, EDT begins to accept lines of text you enter at the terminal. When all the lines have been entered from the terminal, terminate the insert by typing <CTRL/Z>. EDT inserts the lines you typed ahead of the first line in the specified range.

  It is important to remember that EDT does not prompt you for input following an Insert command. Type the command line and a carriage return, and then the lines you want to insert.
- If you specify line numbers with the /Sequence option that conflict with other line numbers in the file, EDT inserts the new lines without line numbers.

If you do not use the /Sequence or /Unsequenced option to specify line numbers, EDT numbers the new lines in the following manner:

- If EDT can number the new lines by incrementing the existing line numbers in steps of 10, it does so as long as the resulting line numbers do not duplicate existing line numbers or violate the ascending order of line numbers.
- If incrementing by 10 would result in an illegal line number, EDT begins incrementing subsequent new lines in steps of 1.
- If incrementing the new lines in steps of 1 would result in illegal or duplicate line numbers, EDT places all further lines in the correct order, but does not assign them any line numbers.

If you insert a large number of lines, you may want to enter a Resequence command to uniformly renumber the lines in the buffer. (See Section 2.3.12.)

- After you type all the lines to be inserted, type a line that contains only <CTRL/Z>. This <CTRL/Z> causes EDT to terminate the insert and display the asterisk prompt, allowing you to enter further commands.

2.3.8 **MOVE**

Use the Move command to transfer lines from one location to another and to delete them from their original location.

Format

```
M[OVE] range-1 %TO range-2    [/Q[UERY]]
                              [/SEQ[UENCE]:initial-number:increment]
                              [/UN[SEQUENCED]]
```

MOVE ... %TO

Specifies the Move command.

range-1

Specifies the lines that are to be moved. If you specify more than one line in range-1, all the lines you specify are moved.

range-2

Specifies the line ahead of which the lines in range-1 are moved. If you specify more than one line in range-2, the lines in range-1 are moved ahead of the first line in range-2.

/QUERY

Allows you to specify how each line in range-1 is to be treated. Before EDT transfers each line, it prints the line and waits until you type any one of the following responses:

Response	Result
Y or YES	Moves the line
N or NO	Does not move the line
Q or QUIT	Stops moving lines and displays the asterisk prompt
A or ALL	Moves the remaining lines in the range without printing them first

/SEQUENCE:initial-number:increment

Assigns specific line numbers to the transferred lines. The initial-number argument specifies the number of the first line that was moved, and the increment argument specifies the increment between numbers. For example, /SEQ:100:100 creates the transferred lines with line numbers of 100, 200, 300, 400, and so on.

/UNSEQUENCED

Causes EDT to transfer the lines without assigning them line numbers.

Examples

```
M.:"MAC" %TO =X 40
```

This command moves all lines, beginning with the current line and continuing through the first line containing the string MAC, to a position preceding line 40 in buffer X.

```
M.;20 %TO %BE
```

This command moves 20 lines, starting with the current line, to the beginning of the current text buffer.

```
M =X %TO =MAIN %E
```

This example appends the contents of text buffer X to the end of buffer MAIN.

Notes

- After a line or group of lines has been moved, EDT positions the line pointer at the first line of the transferred lines in their new position preceding the first line in range-2.
- You cannot overlap ranges: range-1 and range-2 cannot contain any of the same lines.
- If you use the Move command to transfer a large number of lines, you can issue a Resequence command to establish uniform line number increments. (See Section 2.3.12.)

If you do not use either the /Sequence or /Unsequenced option to specify line numbers, EDT attempts to number the new lines in the following manner:

- If EDT can number the new lines by incrementing the existing line numbers in steps of 10, it does so as long as the resulting line numbers do not duplicate existing line numbers or violate the ascending order of line numbers.
- If incrementing by 10 would result in an illegal line number, EDT begins incrementing subsequent new lines in steps of 1.
- If incrementing the new lines in steps of 1 would result in illegal or duplicate line numbers, EDT places all further lines in the correct order, but does not assign them any line numbers.

2.3.9 PRINT

Use the Print command to create a file from the contents of a text buffer. The file that you create in this way contains as part of the text the EDT line numbers assigned to the lines in the range.

Format

PR[INT] [range] /FI:filespec

PRINT

Specifies the Print command.

range

Specifies the buffer contents that you want placed in the permanent storage file you create. If you do not specify a range, EDT uses the entire current text buffer as the range.

/FI:filespec

Specifies the output file that you create. You must specify the filespec using the /FI option. Section 1.3 contains a complete description of file specifications.

Examples

```
PR =B 10:70/FI:INT.FOR
```

This command creates a file called INT.FOR that contains lines 10 through 70, inclusive, of buffer B.

```
PR =MAIN/FI:SY.MAC
```

This command creates a file called SY.MAC from the contents of the main text buffer. The file SY.MAC has as part of the text of the file the line numbers that existed in the MAIN buffer at the time the Print command was issued.

Notes

- EDT does not reposition the line pointer after the Print command generates on output file.

2.3.10 QUIT

Use the Quit command to terminate EDT and to return control to the system monitor without saving the contents of any text buffers.

Format

QUIT

QUIT

Specifies the Quit command.

Example

```
QUIT
```

This command erases all text buffer contents and returns control to the system monitor.

Notes

- The Quit command does not modify any buffers. It does not write out the contents of the main buffer or any other buffer, nor does it generate any files.
- If you invoke EDT with only an input file, you must either issue a Quit command to return to the system monitor or issue an Exit command with the /Rename option to specify an output file. You can also use the Write command to generate a file from the contents of the main text buffer or an alternate text buffer. See Section 2.3.19 for a description of the Write command.

  If you use the Write command to generate a file before you issue the Quit command, that file is not affected by Quit or any other subsequent EDT command.
- The Quit command is especially useful if you have made an error that results in the loss of all or part of a buffer you are editing. If you make a mistake that results in a loss of your buffer's contents, issuing a Quit command negates all changes made since EDT was invoked.

2.3.11 REPLACE

Use the Replace command to delete one or more lines of text in a text buffer and replace them with new lines.

Format

```
R[EPLACE] [range]     [/SEQ[UENCE]:initial-number:increment]
                      [/UN[SEQUENCED]]

text

<CTRL/Z>
```

REPLACE

Specifies the Replace command.

range

Specifies those lines that are to be deleted. The replacement lines begin with the first line in the range. If you do not specify a range, the Replace command deletes the current line and inserts the replacement lines in its place.

/SEQUENCE:initial-number:increment

Assigns specific line numbers to the replacement lines. The initial-number argument specifies the number of the first line, and the increment argument specifies the increment between numbers. For example, /SEQ:100:100 creates replacement lines with line numbers of 100, 200, 300, 400, and so on.

/UNSEQUENCED

Causes EDT to create the replacement lines without assigning them line numbers.

text

Represents one or more complete lines of text. The last line must end with a carriage return.

<CTRL/Z>

Causes EDT to stop accepting replacement text and display the asterisk prompt.

Examples

```
R <RET>
GOTO 40 <RET>
<CTRL/Z>
```

This example deletes the current line, replaces it with GOTO 40, and terminates the replacement text with <CTRL/Z>.

```
R 'ABC'+20 <RET>
IF XYZ THEN DEF <RET>
<CTRL/Z>
```

This example deletes the twentieth line after the first occurrence of ABC, replaces it with the "IF...THEN..." statement, and terminates the replacement text with <CTRL/Z>.

Notes

- After a line or group of lines has been replaced, EDT positions the line pointer at the line following the last replacement line.
- If you replace a single line with several lines, you can issue a Resequence command to assign uniform line number increments to all lines in the text buffer. (See Section 2.3.12).
- If you specify line numbers with the /Sequence option that conflict with other line numbers in the file, EDT inserts the replacement lines without line numbers.

  If you do not use the /Sequence or /Unsequenced option to specify line numbers, EDT numbers the new lines in the following manner:

  - If EDT can number the new lines by incrementing the existing line numbers in steps of 10, it does so as long as the resulting line numbers do not duplicate existing line numbers or violate the ascending order of line numbers.
  - If incrementing by 10 would result in an illegal line number, EDT begins incrementing subsequent new lines in steps of 1.
  - If incrementing the new lines in steps of 1 would result in illegal or duplicate line numbers, EDT places all further lines in the correct order, but does not assign them any line numbers.
- When you issue the Replace command, EDT does not prompt you to begin entering the replacement lines of text. You should type the Replace command line, a carriage return, and then type the lines you want to insert.

  After you type the replacement lines, type a line that contains only <CTRL/Z>. <CTRL/Z> causes EDT to terminate the replacement and display the asterisk prompt, allowing you to issue further commands.

2.3.12 RESEQUENCE

Use the Resequence command to assign new line numbers to the lines in text buffers.

Format

```
RES[EQUENCE] [range] [/SEQ[UENCE]:initial-number:increment]
                    [/UN[SEQUENCED]]
```

RESEQUENCE

Specifies the Resequence command.

range

Specifies the lines and text buffer to be renumbered. If you do not specify a range, EDT resequences all lines in the current text buffer.

/SEQUENCE:initial-number:increment

Assigns specific line numbers to the lines in the range. The initial-number argument specifies the number of the first line in the range, and the increment argument specifies the increment between numbers. For example, /SEQ:100:100 renumbers the range with line numbers of 100, 200, 300, 400, and so on.

/UNSEQUENCED

Causes EDT to remove line numbers from the range.

Examples

```
RES
```

This command causes EDT to renumber the current text buffer's lines at 10, 20, 30, and so on.

```
RES =A /SEQ:1:1
```

This command causes EDT to renumber the lines of buffer A as 1, 2, 3, 4, and so on.

Notes

- After a buffer or a group of lines has been resequenced, EDT positions the line pointer at the first line that was resequenced.
- If you do not specify the /Sequence or /Unsequenced option, EDT assigns to the lines in the text buffer the default line numbers 10, 20, 30, and so on.
- As a general practice, resequence the entire text buffer instead of a portion of it. This ensures that uniform line number increments exist for your convenience.

  When you renumber only part of the text buffer, none of the line numbers you assign can match those in the part of the buffer not being renumbered. If you specify a line number for the resequenced range that is less than the line numbers before it, all lines in the range are written without numbers. The highest sequence number that you can use is 65535; the lowest is 1.
- After you insert or delete a large number of lines, the line numbers of the remaining lines may be difficult to work with. Use the Resequence command to restore uniform line number increments to the text buffer.
- Use the Type command (Section 2.3.18) to display on your terminal the line numbers that are currently assigned to the contents of the buffer. Use the Print command (Section 2.3.9) to create a file that contains the new line numbers as part of the text.

2.3.13 RESTORE

Use the Restore command first to locate a file created by a Save command and then to use the file to recreate the status of all files and the contents of all text buffers as they were preserved in the file.

Format

REST[ORE] /FI:filespec

RESTORE

Specifies the Restore command.

/FI:filespec

Specifies the file that was created by previously issuing a Save command. That file, when accessed by the Restore command, recreates the contents of all text buffers in use at the time the Save command was entered.

EDT does not provide defaults for either the file name or the file type in the Restore and Save commands. See Section 1.3 for a complete description of file specifications.

Example

```
>EDT <RET>
EDT> <RET>
*REST /FI:TXT.SAV
*
```

This example illustrates the sequence you must follow if you want to use the Restore command. Restore must be the first command you issue after invoking EDT without specifying either an input or an output file. EDT then locates the file TXT.SAV, and uses that file to recreate the contents of all text buffers in use at the time the Save command created TXT.SAV.

Notes

- In order to issue a Restore command, Restore must be the first command that you issue after you invoke EDT. Therefore you must:

    1. Invoke EDT without specifying either an input or output file.
    2. Have all text buffers empty.
    3. Issue the Restore command, and specify the file created by a Save command in a previous editing session.

  After your text buffer contents have been restored, EDT positions the line pointer at the line that was the current line when the Save command created the input file.

2.3.14 SAVE

Use the Save command to preserve in a file that you specify the contents of the text buffers and the status of all files you use during your editing session. You can later use the Restore command to recreate your editing session as it was when you issued the Save command.

Format

SA[VE] /FI:filespec

SAVE

Specifies the Save command.

/FI:filespec

Specifies the file that EDT creates. This file contains the current status of all the buffers in use. You must use the /FI option to specify the name of the file.

EDT does not provide defaults for either the file name or the file type in the Restore and Save commands. See Section 1.3 for a complete description of file specifications.

Example

```
SA /FI:TXT.SAV
```

This command saves all text buffer contents by creating the file TXT.SAV. The file TXT.SAV contains the contents of all the text buffers in use at the time the Save command was issued. You can use the TXT.SAV file only by issuing a Restore command to recreate the contents of the text buffers.

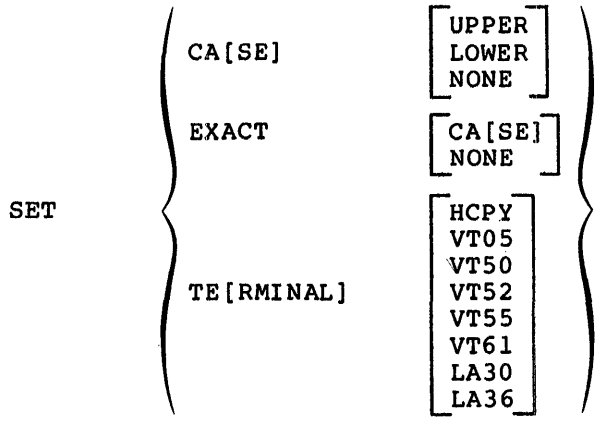
Notes

- EDT does not reposition the line pointer after the Save command generates the file, nor does the Save command alter the contents of text buffers. After a Save command, you can continue to edit as if you had never issued the command.
- The file created by a Save command is in a special format and is useful only as input to the Restore command (see Section 2.3.13). Do not try to use the file created by a Save command as a normal EDT input file.

2.3.15 SET

Use the Set command to establish criteria that are used by other EDT commands to flag upper-case or lower-case characters, to determine if two strings match, and to establish the correct parameters for the terminal you use.

Format



SET

Specifies the Set command.

CASE

Specifies that characters of the case you specify, either Upper, Lower, or None, are to be preceded by an apostrophe when they are displayed on the terminal. If you specify Case Upper, all upper-case characters are flagged. If you specify Case Lower, all lower-case characters are flagged. If you specify Case None (the default setting), no characters are flagged, and any previously entered Case specifications are terminated. EDT does not flag characters while in Character Mode.

EXACT

Specifies whether object string matches must conform to the cases specified in the object string. If Exact Case is specified, match strings must conform to the cases of characters specified in the object string. If Exact None (the default setting) is specified, only the characters must match; case matches are not required.

TERMINAL

Specifies the terminal type that you are using. The valid terminal types are listed above in the command format. Enter the terminal type that corresponds to the terminal you are using. The default setting for the Terminal parameter is established when EDT is installed.

Examples

```
*SET CASE LOWER <RET>
*40 <RET>
40 IF ABC 'L'T 'X'Y'Z GOTO 'X'X'X
```

Text of line 40 actually exists in the buffer as:

```
40 IF ABC lt xyz GOTO xxx
```

This example illustrates that Set Case Lower displays all lower-case characters with a leading apostrophe.

```
*SET EXACT CASE <RET>
*'ABC' <RET>
No such line found.
*SET EXACT NONE <RET>
*'ABC'<RET>
130 IF abc, GOTO x
```

This example illustrates that, when Set Exact Case has been specified, EDT accepts only exact case matches.

Notes

Using the Set Case Command

- The Set Case Upper and Set Case Lower commands do not alter the characters in the text buffer; they only change the way the characters are displayed on the terminal.

  You should generally use the Set Case Lower command when you have an upper-case-only terminal. All characters that subsequently appear with a preceding apostrophe are lower-case characters. Lower-case characters cannot be changed or inserted from an upper-case-only terminal, but they can be deleted.

- EDT does not flag characters while in Character Mode. You therefore cannot distinguish between upper-case and lower-case while in Character Mode if you have an upper-case-only terminal.

Using the Set Exact Command

- The Set Exact commands do not cause any changes in the text buffer. These commands only control the criteria that EDT uses to evaluate a string match.

Using the Set Terminal Command

- Setting the correct terminal type is mandatory only when you edit in Character Mode. Generally, however, you should always have the correct terminal type set.

  The default setting for the Terminal parameter is established when EDT is installed in the system. Use the Show Terminal command (Section 2.3.16) to find out what default terminal type is in effect, then use the Set Terminal command to alter the Terminal parameter if necessary.

2.3.16 SHOW

Use the Show command to display buffer and software version information, and EDT parameters established by the Set command.

Format

```
SH[OW]   [ BU[FFERS]  ]
         [ CA[SE]     ]
         [ EXACT      ]
         [ TE[RMINAL] ]
         [ VE[RSION]  ]
```

SHOW

Specifies the Show command.

BUFFERS

Specifies that EDT display the name and summarized contents of each buffer you have used since invoking EDT, and indicate the current buffer with a leading equal sign (=).

CASE

Specifies that EDT display the Case parameter currently in effect as established by the Set command.

EXACT

Specifies that EDT display the Exact parameter currently in effect as established by the Set command.

TERMINAL

Specifies that EDT display the Terminal parameter currently in effect as established by the Set command.

VERSION

Specifies that EDT display the version number of the current EDT program.

Examples

```
*SH CASE
UPPER
*
```

This example shows that the Case Upper parameter is in effect.

```
*SH BU
MAIN     14 LINES
A        0 LINES
BUF2     143 LINES
=BUF     390 LINES
*
```

This example displays the buffers referenced since EDT was invoked, the order in which they were created, and a summary of their contents. The current buffer is shown with a leading equal sign (=).

Notes

- The Show command, except for the Buffers and Version options, displays only those options that have been specified, or that can be specified, through the Set command (see Section 2.3.15).

2.3.17 SUBSTITUTE

Use the Substitute command to change characters within lines of the text buffer.

Format

```
                                                          [/BR[IEF]][:n]
S[UBSTITUTE]/object-string/replacement-string/[range]     [/Q[UERY]]
                                                          [/-T[YPE]]

     or

S[UBSTITUTE] N[EXT]
```

SUBSTITUTE

Specifies the Substitute command.

/

Represents a string delimiter. String delimiters can be slashes or any other printing, nonalphanumeric character. The delimiters in a given Substitute command must match each other.

object-string

Specifies the characters in the text buffer that you want to locate and to change. See Section 2.2.2.4 for more information about object strings.

replacement-string

Specifies the characters with which you want to replace the object string characters.

range

Specifies the range of lines within which all occurrences of the object string are changed to the replacement string. If you do not specify a range, EDT changes only the first occurrence of the object string that it locates.

/BRIEF

Specifies that EDT display the first n characters of each line in which it makes a substitution. If you omit the n argument, EDT displays the first 10 characters.

/QUERY

Specifies that you want to control the operation by using EDT prompts and responses. EDT prompts you by displaying the substitution about to be made and waits for you to enter any one of the following responses:

Response	Result
Y or YES	Makes the substitution
N or NO	Does not make the substitution
Q or QUIT	Stops substituting and displays the asterisk prompt
A or ALL	Makes substitutions in the rest of the appropriate lines in the range without printing them first

/-TYPE

Inhibits EDT's automatic display of each line after a substitution is made in that line.

NEXT

Repeats the operation of the Substitute command that it must immediately follow.

Examples

```
S/ABC/XYZ
```

This command causes EDT to change the next occurrence of ABC to XYZ. If ABC exists on the current line, it is changed. If not, the contents of the buffer are searched, and the first occurrence is changed.

```
S'/'\'
```

This command causes EDT to change the first slash (/) to a backslash (\), and uses an apostrophe (') as the string delimiter.

```
S/A/B/.
```

This command changes every occurrence of A to B in the current line only.

Note that the final slash (/) string delimiter must be included here since the current line range is specified by the period (.).

```
S/001/100/%WH/Q
```

In this example, EDT attempts to change every occurrence of 001 to 100 in the entire text buffer but queries you before making each substitution.

```
S/.AND/.OR/%REST/-T
```

In this example, EDT changes every occurrence of .AND to .OR from the current line through the end of the text buffer but does not display any of the resulting lines.

```
*S/.MAC/.BAS//-T
*S N
150     IF THE FILE TYPE IS .BAS, THE COMPILER MAKES
```

In this example, EDT changes the first occurrence of .MAC to .BAS, but does not display the changed line. The S N command then changes the next occurrence of .MAC to .BAS and automatically displays the line.

Note in the first command line that two consecutive slashes (/) are used. The first of these indicates the end of the replacement string and the second that an option follows.

Notes

- EDT displays each line that contains a substitution on the terminal after it makes the substitution. If you specify a range that requires substitutions in more lines than you wish to have displayed, specify the /-Type option to inhibit the display.

- The S N or Substitute Next command format must immediately follow another Substitute command. When entered, S N repeats a single substitution as performed by the command it follows.

  The S N command format does not accept a range or any options. It allows you to perform one more substitution that has already been defined without requiring you to retype the command string.

- After EDT makes substitutions, it positions the line pointer at the last line in which a substitution was made.

  You can replace the slash (/) string delimiter with any printing character except a letter or number, for example, a question mark (?), semicolon (;), apostrophe ('), and so on.

  You can omit the final string delimiter if no range or options follow and if the omitted delimiter is the last character in the command line.

2.3.18 TYPE

Use the Type command to display lines of text at your terminal.

Formats

```
[T[YPE]] [range]

  or

T[YPE] [range] [/BR[IEF]][:n]
```

TYPE

Specifies the Type command. Note that this field is optional if you do not specify the /Brief option.

range

Specifies the lines that you want displayed at your terminal. If you omit the range argument, the range defaults to the line following the current line (see Notes).

/BRIEF

Specifies that EDT display only the first n characters of each line in the range. If you omit the n argument, EDT displays the first 10 characters.

Examples

```
'ABC':"DEF"
```

This command displays all lines of text from the first line that contains ABC through the first line that contains DEF, inclusive.

```
T 10:100/BR:5
```

This command displays the first five characters of each line with line numbers between 10 and 100, inclusive.

```
%WH
```

This command displays the entire text buffer contents.

```
*<RET>
120  DID GYRE AND GIMBLE IN THE WABE
*
```

This command, consisting only of a carriage return, advances the line pointer to the next line and prints the line. Line 120 becomes the new current line.

Notes

- You can issue a Type command that consists of only a range. The contents of that range are displayed on the terminal. However, if you issue this format of the Type command, you cannot include any command options.
- After EDT displays the lines in the range on your terminal, it positions the line pointer at the first line in the range.
- A special case of the Type command occurs when you omit both the command and the range; that is, when you enter a command line that consists only of a carriage return. EDT first advances the line pointer to the next line, then types that line. The line just typed becomes the current line. This variation of the Type command allows you to step through the lines in a file simply by pressing <RET> repeatedly.

2.3.19 **WRITE**

Use the Write command to create a file from the contents of a text buffer.

Format

```
WR[ITE] [range]  /FI:filespec
```

WRITE

Specifies the Write command.

range

Specifies the buffer contents that you want to store in the file. If you do not specify a range, EDT uses the entire contents of the current text buffer as a range.

/FI:filespec

Specifies the output file that EDT creates. You must use the /FI option to specify your output file. Section 1.3 contains a complete description of file specifications.

Example

```
WR =B 10:70/FI:INT.FOR
```

This command creates a file named INT.FOR that contains lines 10 through 70, inclusive, of buffer B.

Notes

- EDT does not reposition the line pointer after a Write command, nor does the Write command alter the contents of the range or buffer.
- Use the Write command to generate a file that contains the contents of alternate text buffers that you want to save. When you issue the Exit command, EDT saves only the contents of the main text buffer; all other buffers are erased. (See Section 2.3.4 for a description of the Exit command.)

  You can also use the Write command to save the contents of the main buffer if you did not specify an output file when you invoked EDT.

2.3.20 XEQ (Execute)

Use the Execute command to execute a previously entered sequence of EDT commands.

Format

X[EQ] range

XEQ

Specifies the Execute command.

range

Specifies the lines of text that contain the EDT commands to be executed.

Examples

```
X 10:30
```

where lines 10 through 30 contain:

```
10      'ABC'
20      S/40/70/.
30      <RET>
```

In this example, EDT searches for the next line that contains ABC, and attempts to change all occurrences of 40 to 70 within the line. The blank line implied by the <RET> symbol advances the line pointer past the now current line that contains ABC so that further X 10:30 commands locate the next succeeding line that contains ABC.

```
X =B
```

where buffer B contains:

```
=MAIN %BE
'ABC'
R
XYZ
^Z
```

This example returns the line pointer to the first line of the main buffer, locates and deletes the first line that contains an occurrence of ABC, inserts a line consisting of XYZ, and terminates the insert with the circumflex-Z (^Z) combination.

Notes

- Use the Execute command if you have identical or complex edits to perform, or if you have an established series of commands that are often issued.
- EDT positions the line pointer as directed by the commands contained in the range specified with the Execute command.
- The Execute command cannot alter the contents of the range containing the commands in any way.
- If the range specified with the Execute command contains an Insert or Replace command, use the following steps to terminate the Insert or Replace command properly:

    1. Create the sequence of commands that will be excuted by the Execute command, including the Insert or Replace command.
    2. Enter the new or replacement lines as part of the sequence of commands contained in the range.
    3. When all new or replacement lines are entered, enter a circumflex-Z (^Z) combination as part of the sequence of commands. When EDT executes the sequence of commands, the ^Z is interpreted as <CTRL/Z>.

- When the commands in the range have been executed, EDT displays the asterisk prompt.

2.4 CHARACTER MODE

When you issue the Change command from EDT's Command Mode, you invoke Character Mode. Character Mode allows video terminal users to work at the character and word levels, in addition to the line level provided by Command Mode. Character Mode does this by using the video terminal's cursor as a pointer to the current position in the buffer.

All Character Mode editing operations use the position of the cursor as a reference point. Unlike Command Mode's line pointer, which points to an entire line, the cursor in Character Mode points to one character within a line. Similarly, where Command Mode commands generally operate on no unit smaller than an entire line of text, Character Mode subcommands can operate on the single character indicated by the cursor. Character Mode can also operate on multiple characters, words, and lines.

Character Mode differs from Command Mode in several other respects. The most obvious difference is the lack of line numbers in Character Mode. A second difference is that Character Mode only operates in one buffer at a time. As a result of these two differences, Character Mode is not useful for moving lines of text from place to place and especially from buffer to buffer. You should use Command Mode for these operations. Character Mode is at its best when you must perform extensive editing operations within a line.

When you are working in Character Mode, the video terminal screen always displays the exact contents of the buffer. After each edit, Character Mode updates the display to reflect the current text.

When you issue the Change command, the video terminal screen goes blank, except for the current line of the buffer, which appears at the top of the screen. The cursor is positioned at the first character of the line. Although there is no indication on the screen, Character Mode is waiting for a subcommand (so called because Character Mode is a subfunction of the Change command). If you type something, whatever you type appears at the bottom of the screen, in a zone reserved for subcommands. Depending on the subcommands you issue, you can perform some editing operation within the line at the top of the screen, or cause Character Mode to display more lines of the buffer.

When Character Mode is in use, you have access to the entire range specified in the Change command. However, since your screen can only display a certain number of lines of text, the screen is like a window on the text buffer. If you direct the cursor to a line that is not in the current display, Character Mode scrolls the display up or down, as appropriate, to bring the specified line into view. (If you move the cursor a large number of lines, the display does not scroll; it vanishes, and the specified line is displayed at the top of the screen.)

Once the cursor is positioned at the beginning of the line you wish to change, you direct the cursor to the desired character within the line. You do this either by issuing subcommands or by using the arrow keys on your terminal's keypad (if your terminal is so equipped) to move the cursor. You can then perform any of these actions:

- Delete a specified number of characters, words, or lines
- Replace a specified number of characters, words, or lines with new text

- Insert text in front of the cursor
- Substitute the next (or previous) occurrence of a specified string with another string

At the end of the Character Mode session, you issue the Exit subcommand to terminate Character Mode and return to Command Mode.

When you use Character Mode, it is important that EDT's Terminal parameter matches the type of terminal you are working on. Sections 2.3.15 and 2.3.16 describe the Set and Show commands, which allow you to establish and display the value of the Terminal parameter.

In addition, you should be sure that your terminal is correctly described in the operating system's list of devices. Use the MCR Set command or (for RSX-11M-PLUS users) the DCL Show command to make sure your terminal is set correctly (see the RSX-11M/M-PLUS MCR Operations Manual or the RSX-11M-PLUS Command Language Manual, respectively).

The remainder of this section describes:

- Editing subcommands, which allow you to position the cursor and perform editing operations
- Methods of cursor manipulation other than those available with the editing subcommands
- Subcommand concatenation, which allows you to enter several subcommands on one line
- Two subcommands to terminate Character Mode

2.4.1 Editing Subcommands

Editing subcommands are divided into three basic groups:

- Subcommands that move the cursor, delete, or replace a specified number of characters, words, or lines
- The Insert subcommand, which allows you to insert text in front of the cursor position
- The Substitute subcommand, which allows you to substitute for the next (or previous) occurrence of a string

Each of these groups is treated individually in the sections that follow.

2.4.1.1 Subcommands That Move the Cursor, Delete, or Replace

Subcommands in this group are made up of three fields: action, repetition count, and unit. The action field specifies what you want to do; the repetition count field specifies how many times you want to do it, and in which direction; and the unit field specifies whether you want to do it to characters, words, or lines. The action and repetition count fields are optional, but you must specify a unit. You cannot separate the fields with spaces or tabs.

Format

```
[action][-][repcount]unit<RET>
```

action

Represents a 1-letter abbreviation (or a null field) that specifies the editing operation that you wish to perform.

If this field is null, Character Mode moves the cursor the specified number of units in the buffer, and displays the text if it has not already been displayed.

If this field is a D (for Delete), Character Mode deletes the specified number of units, starting at the current cursor position.

If this field is an R (for Replace), Character Mode first deletes the specified number of units, then waits for you to insert text in place of the deleted units. When you finish inserting the replacement text, enter a <CTRL/Z>. This directs Character Mode to terminate the insert operation and wait for another subcommand. (See Section 2.4.1.2 for a full description of the Insert subcommand.)

repcount

Represents an integer that specifies the repetition count. The repetition count tells Character Mode how many times to perform the action. If the repetition count is positive, the action takes place for the specified number of units following the cursor, including the unit at which the cursor is currently positioned. If the repetition count is negative, the action takes place for the specified number of units preceding the cursor, not including the unit at which the cursor is currently positioned.

If you leave this field null, the repetition count defaults to 1, and the action takes place on the unit at which the cursor is currently positioned. If you do not enter a minus sign (-) with the repetition count, the sign defaults to positive.

unit

Represents a 1-letter abbreviation that specifies the unit on which the action is to take place. The unit can be C (for character), W (for word), or L (for line).

A character is a single ASCII character, including spaces and tabs. If you specify the /NL option with the Change command, line terminators are also considered single characters and can be deleted. (See Section 2.3.1 for a description of the Change command.)

A word is any string of printing characters and trailing word terminators. A word begins at its first character and includes all characters in the word and all word terminators (spaces and/or tabs) up to the first character of the next word.

A line begins at the first character of the line and includes all characters in the line up to (and including) the line terminator. The line terminator can be a carriage return, line feed, form feed, or vertical tab.

Examples

D3W

This subcommand deletes three words, beginning with the word at which the cursor is currently positioned.

DW

In this subcommand, the repetition count defaults to 1; therefore, the subcommand deletes the single word at which the cursor is positioned.

-4W

In this subcommand, the action field is left null; therefore, the cursor moves backwards in the buffer by four words, coming to rest on the first character of the fourth word preceding its starting position.

DC

This subcommand deletes the single character at which the cursor is currently positioned.

D5C

This subcommand deletes the character at which the cursor is currently positioned and the four characters following that position, for a total of five characters.

D-5C

This subcommand deletes the character immediately preceding the cursor position and the four characters preceding that one, for a total of five. Note that this subcommand does not delete the character at which the cursor is positioned.

D-3W

This subcommand deletes the three words immediately preceding the word in which the cursor is positioned. Note that the word in which the cursor is positioned is not deleted if the repetition count is negative.

D2L

This subcommand deletes the line in which the cursor is positioned and the following line, for a total of two lines.

D-3L

This subcommand deletes the three lines immediately preceding the line in which the cursor is positioned. Following the operation, the cursor comes to rest at the beginning of the line in which it was originally positioned, that is, the first line following the deleted lines.

R2C

This subcommand first deletes two characters starting at the current cursor position, then waits for you to input replacement text. When you enter <CTRL/Z>, the replacement operation terminates and Character Mode waits for another subcommand.

In practice, when you issue a Replace subcommand, the specified characters and the remainder of the affected line vanish. You can then enter the replacement text, including carriage returns if necessary. When you enter <CTRL/Z>, the remainder of the affected line reappears in its proper position following the replacement text.

R3W

This subcommand first deletes three words, starting with the word in which the cursor is currently located, then waits for you to input replacement text. When you enter <CTRL/Z>, the replacement operation terminates and Character Mode waits for another subcommand.

When replacing words, it is important to remember that you must also replace the word terminator(s). A word includes all spaces or tabs until the first character of the next word, or the first line terminator.

2W3C

This subcommand string first moves the cursor two words forward from the current position, to the first character of the second word. It then moves the cursor three characters further, to the fourth character of the word.

This example and the following one illustrate subcommand concatenation. For more information on subcommand concatenation, see Section 2.4.3.

2L3WD2W

This subcommand string first moves the cursor down two lines to the beginning of the second line following its original position. It then moves the cursor three words into the line, to the beginning of the fourth word. Finally, it deletes two words, starting at the current cursor position; that is, the fourth and fifth words of the line.

Notes

- When you are deleting or replacing characters or words, normally you cannot delete beyond the end or beginning of the current line. For example, the cursor is two words from the end of the line, and you issue the subcommand:

  ```
  D5W
  ```

  In this situation, Character Mode does not delete the last two words on the current line and the first three on the next line, as you might expect. Only the last two words on the current line are deleted. The situation is the same if you try to delete or replace characters beyond the end or beginning of the current line.

  The situation changes, however, if you specify the /NL option when you issue the Change command. The /NL option directs Character Mode to treat line terminators as single characters that can be deleted. If you specified /NL, and you try to delete beyond the end or beginning of a line, you will succeed; Character Mode deletes the line terminator and joins the two lines that the line terminator used to separate.

  This distinction becomes important if you wish to delete or replace all the words or characters between the current cursor position and the end or beginning of the current line. If you issued the Change command without the /NL option, you can simply specify a repetition count that is larger than the number of words or characters you wish to delete. Character Mode will delete only to the end or beginning of the line, depending on whether the repetition count is positive or negative. If, however, you specified the /NL option, you must specify a repetition count that exactly matches the number of words or characters you wish to delete. If the repetition count you specify is too large, you will delete the line terminator and possibly some words in the following or preceding line.

  If you specified the /NL option and you delete or replace characters beyond the end or beginning of a line, remember that the line terminator counts as a single character and must be accounted for when you calculate the repetition count. However, if you are deleting words, you do not have to include the line terminator in the repetition count, because the line terminator is treated as part of the last word in the line.

- Spaces are not allowed between the elements of a subcommand. If, for example, you attempt to issue the subcommand

  ```
  D 5 W
  ```

  Character Mode returns the error message

  ```
  Illegal subcommand
  ```

- The DELETE key on your terminal can be used directly to delete characters from the text buffer. <DEL> deletes the character directly to the left of the cursor. Thus, a single <DEL> keystroke is equivalent to the subcommand:

  ```
  D-C
  ```

Use care when deleting a subcommand that you have entered incorrectly. If you inadvertently delete more characters than are in the subcommand, you will also delete some characters from the text buffer.

2.4.1.2 **The Insert Subcommand** - The Insert subcommand allows you to insert text at any point in the text buffer. The text can consist of any combination of characters, including carriage returns and other line terminators. When you want to terminate the insert operation, a <CTRL/Z> directs Character Mode to stop accepting text for insertion and wait for another subcommand.

Format

[loc]I<RET>

loc

Represents a Character Mode subcommand or combination of subcommands that results in a new cursor position. (Note, however, that a string search operation must be the last element entered in a subcommand string; see Section 2.4.3.)

I

Specifies that the text you enter following the <RET> be inserted immediately preceding the current cursor position. Character Mode continues to insert text until you enter a <CTRL/Z>.

Notes

- In practice, the Insert subcommand causes the remainder of the line in which the cursor is currently positioned to vanish, leaving the cursor at the position where the insert will occur.

  When you finish typing the insert text and enter a <CTRL/Z>, the remainder of the line reappears in the proper position. It is not necessary to type a <RET> before the <CTRL/Z>, as is the case when inserting in Command Mode. Any carriage returns that you enter while inserting become part of the insert.

  If you position the cursor under the first character of a line and issue the Insert subcommand, the entire line vanishes. You can then insert one or more complete lines, including line terminators. These lines will go into the text buffer above the line in which the cursor was originally positioned. When you enter a <CTRL/Z>, the line above which the new lines have been inserted reappears.

  If you position the cursor at the end of a line and issue the Insert subcommand, the resulting insert occurs between the last character of the line and the line terminator.

- A long insert in a long line can result in text disappearing off the right edge of the screen. The text is still in the buffer, however, and can be recovered by inserting a carriage return in the line.

2.4.1.3 **The Substitute Subcommand** - The Substitute subcommand works in much the same way as the Substitute command works in Command Mode, and follows similar format rules. It allows you to substitute a specified string with another string.

Format

[-]S/objstring/repstring[/]

-

Specifies that the direction of search for the object string should be toward the top of the buffer, rather than toward the bottom. If this field is null, the search is toward the bottom of the buffer.

S

Specifies the Substitute subcommand.

/

Represents the string delimiter. The delimiters can be slashes or any other printing, nonalphanumeric characters; however, the delimiters used in a given Substitute operation must match. Note that the final terminator is optional, as long as you do not place additional subcommands at the end of the Substitute subcommand. (For a complete description of subcommand concatenation, see Section 2.4.3.)

objstring

Specifies the object string that is to be replaced with the replacement string. The criteria that Character Mode uses to determine whether two strings match are those established by the Command Mode Set command (see Section 2.3.15).

repstring

Specifies the replacement string that is to replace the object string.

Notes

- When you issue the Substitute subcommand, the object string you specify becomes the last-specified object string. This means that you can later locate the next (or preceding) occurrence of the object string by entering paired quote marks ("") or apostrophes (''). See Section 2.4.2.2 for a full description of Character Mode string searches.

2.4.2 Cursor Manipulation

You can use the editing subcommands presented in Section 2.4.1.1 to move the cursor in the text buffer. There are three additional methods of moving the cursor:

- Subcommands that move the cursor to the beginning or end of a specified unit of text

- String searches, which move the cursor to the beginning of a specified object string
- Keypad arrows, on terminals that are equipped with them

2.4.2.1 **Cursor Subcommands** - The cursor manipulation subcommands consist of two letters. The first letter is B or E, and specifies the beginning or end of the unit specified by the second letter. The second letter can be W, for word; L, for line; or R, for range. The range is equivalent to the range that you specified when you issued the Command Mode Change command. Table 2-8 summarizes the cursor manipulation subcommands.

Table 2-8
Cursor Manipulation Subcommands

Subcommand	Function
BL	Moves the cursor to the beginning of the current line
BW	Moves the cursor to the beginning of the current word
BR	Moves the cursor to the first character in the range specified in the Change command
EL	Moves the cursor to the end of the current line
EW	Moves the cursor to the end of the current word
ER	Moves the cursor to the end of the range that was specified in the Change command

2.4.2.2 **String Searches** - String searches move the cursor to a specified object string, either before or after the beginning cursor position. The format of a string search request is as follows:

```
[-][n]'[objstring][']
```

-

Specifies that the search should proceed toward the top of the text buffer. If you leave this field null, the search proceeds toward the bottom of the text buffer.

n

Specifies which occurrence of the object string to locate. The value of n defaults to 1.

'

Represents the string delimiter. For string searches, the apostrophe (') and the quote mark (") are legal string delimiters. The delimiter that you use at the beginning of the string must match the one you use at the end of the string.

objstring

Specifies the object string. Following a successful search, the cursor is positioned at the first character in the object string. If the objstring field is left null, the search is made for the object string that was specified in the most recent string search operation or Substitute subcommand.

Examples

```
'BOMBARD'
```

This example moves the cursor to the first letter of the first occurrence of the word BOMBARD. If the cursor is already positioned at the first letter of BOMBARD, it moves toward the end of the buffer searching for another occurrence of BOMBARD.

```
-2'FLUGELHORN'
```

This example moves the cursor to the second occurrence of FLUGELHORN preceding the starting cursor position. If the starting cursor position is on the first letter of an occurrence of FLUGELHORN, that occurrence does not count in the total. However, if the cursor is in the middle of an occurrence of FLUGELHORN, that occurrence is counted.

```
''
```

This example moves the cursor to the first occurrence of the last string specified as the object string in a search or substitute operation. Repeated executions of the same subcommand will move the cursor to subsequent occurrences of the string in the buffer. This feature allows you to step through repeated occurrences of a given object string without reentering the string each time.

Notes

- To determine whether two strings match, Character Mode uses the criteria specified in a Command Mode Set command. Specifically, if Exact Case is in effect, both the characters and their cases must match before a match is declared. If Exact None (the default) is in effect, only the characters in the two strings must match. Sections 2.3.15 and 2.3.16 describe the Set and Show commands, which allow you to establish and examine the Exact parameter.

- When Character Mode carries out a string search operation, it always moves the cursor one character in the direction of the search before beginning. This means that a search operation does not find a particular object string if it starts from the first letter of that object string.

2.4.2.3 **Keypad Arrows** - Many terminals are equipped with a numeric keypad, and some keypads have a set of four directional arrows. These arrows provide yet another means of moving the cursor. The two vertical arrows move the cursor up or down a line each time you press

them, while the two horizontal arrows move the cursor one character to the left or right. The four arrows are therefore equivalent to the following cursor manipulation subcommands:

Arrow	Equivalent Subcommand
↓	L
↑	-L
→	C
←	-C

2.4.3 Subcommand Concatenation

You can enter two or more subcommands on a single line. This is called subcommand concatenation; a group of concatenated subcommands is called a subcommand string. Subcommand concatenation allows you to direct Character Mode to perform a series of editing operations without waiting for each operation to complete before entering the next subcommand.

When you concatenate subcommands, Character Mode carries out each subcommand in the order that you enter it. When Character Mode finishes a given operation, it checks to see if there are more characters on the subcommand line. If there are, Character Mode scans the characters until it assembles a complete subcommand or encounters an error. If it finds a complete subcommand, it carries it out; if it finds an error, it returns an error message and ignores the rest of the subcommand line.

You may, if you wish, separate concatenated subcommands with a space. However, because the syntax of Character Mode subcommands is so tightly defined, Character Mode does not require that subcommands be delimited with spaces or other characters.

There are two limitations on the concatenation of subcommands:

- You cannot enter a subcommand string longer than 67 characters.
- If your string of subcommands includes a string search, the string search must be the last subcommand entered. Character Mode ignores any subcommands entered on a line after a string search.

There are no other limitations on the subcommands you can string together. You can even embed Insert and Replace subcommands in a subcommand string. If you do so, the <CTRL/Z> you enter to terminate the insert or replacement sends Character Mode looking for another subcommand.

In practical terms, however, subcommand concatenation is useful only up to the point at which it becomes difficult to think that many moves ahead. Too much subcommand concatenation defeats one of the goals of Character Mode, which is to provide a fast, intuitive means of editing within lines. How much concatenation you do will depend on your expertise in Character Mode and your ability to think ahead.

The following examples illustrate some subcommand strings that are often useful. Some of the examples separate their subcommands with spaces in order to clarify the example; others do not.

Examples

```
EL I
```

This subcommand string directs Character Mode to go to the end of the current line and insert text.

```
EL DW
```

This subcommand string directs Character Mode to go to the end of the current line and delete the last word on the line.

```
ER L I
```

This subcommand string directs Character Mode to go to the end of the range (that is, to the last line of the range specified in the Change command); go down an additional line, to the [EOB] (End of Buffer) position; and insert text above the [EOB]. Use this subcommand string to insert text at the end of the range. If you do not include the L, directing Character Mode to the [EOB] marking, you insert text above the last line in the range, instead of below it.

```
-2LEL-2WRW
```

This subcommand string directs Character Mode to move the cursor two lines above its current position; to move to the end of that line; to back up two words to the third word from the end; and to replace that word. It is often easier to count from the end of the line to a word than from the beginning of the line.

```
3WEWRC
```

This subcommand string directs Character Mode to move the cursor three words beyond the current word; to go to the end of the new current word; and to replace the last character of that word.

2.4.4 Subcommands to Terminate Character Mode

If you wish to terminate Character Mode and return to Command Mode, issue the subcommand:

EX stands for Exit. It terminates Character Mode and returns the Command Mode asterisk prompt (*). The range that you specified with the Change command will contain the edits you made while in Character Mode. When you start editing in Command Mode, your current line will be the same as it was when you issued the Exit subcommand.

If you wish to terminate Character Mode and return directly to the monitor, issue the subcommand:

```
QUIT
```

Quit behaves like its Command Mode counterpart (see Section 2.3.10). No output files are generated when you issue the Quit subcommand.

2.5 SUMMARY OF EDT COMMAND MODE COMMANDS

Command Format	Description
C[HANGE] [range] [/NL]	Invokes Character Mode.
CO[PY] range-1 %TO range-2 [/Q] [/SEQ] [/UN]	Copies the lines in range-1 to a position ahead of the first line in range-2.
D[ELETE] [range] [/Q]	Deletes lines from the buffer.
EX[IT] [/RE:filespec]	Terminates EDT; writes contents of the main text buffer to specified output file.
F[IND] range	Moves the line pointer to the first line in the range.
INC[LUDE] [range] /FI:filespec [/SEQ] [/UN]	Locates a file and copies it into a text buffer.
I[NSERT] [range] [/SEQ] [/UN] . . <CTRL/Z>	Inserts text typed at the terminal in the buffer ahead of the first line in the range.
M[OVE] range-1 %TO range-2 [/Q] [/SEQ] [/UN]	Transfers the lines in range-1 to a position ahead of the first line in range-2.
PR[INT] [range] /FI:filespec	Generates an output file from the contents of the range. The output file contains EDT line numbers as part of the text.
QUIT	Terminates EDT; saves no edits or text buffers; generates no files.
[REPLACE] [range] [/SEQ] [/UN] . . <CTRL/Z>	First deletes the lines in the range, then inserts text typed at terminal into buffer in place of deleted text.

Command Format	Description
RES[EQUENCE] [range] [/SEQ] [/UN]	Assigns new line numbers to the lines in the range.
REST[ORE] /FI:filespec	Locates specified file created by a Save command; uses the file's contents to restore the status and contents of the text buffers.
SA[VE] /FI:filespec	Creates a file that contains the status and contents of the text buffers currently in use.
SE[T] { CA[SE] [UPPER / LOWER / NONE] / EXACT [CA[SE] / NONE] / TE[RMINAL] [HCPY / VT05 / VT50 / VT52 / VT55 / VT61 / LA30 / LA36] }	Establishes criteria that other EDT commands use in their operation; flags upper-case or lower-case characters; and establishes proper terminal parameters.
SH[OW] [BU[FFERS] / CA[SE] / EXACT / TE[RMINAL] / VE[RSION]]	Displays the values established by the Set command, as well as current buffer status and software version information.
S[UBSTITUTE]/str-1/str-2/[range] [/BR] [/Q] [/-T]	Changes str-1 to str-2. If range is specified, changes all str-1s in the range. If no range, changes only the first str-1 encountered.
S[UBSTITUTE] N[EXT]	Repeats the operation of the substitute command that it must immediately follow.
[T[YPE]] [range] or T[YPE] [range] [/BRief]	Displays the contents of the range on your terminal.
WR[ITE] [range] /FI:filespec	Creates an output file from the contents of the range.
X[EQ] range	Executes the EDT commands contained in the range.

2.6 ERROR REPORTING AND ERROR MESSAGES

As you use EDT, you may from time to time make errors in your use of the commands. EDT provides a set of error messages that inform you of the nature of the error you have made.

If you are unfamiliar with the operation of EDT, and you need more information than you get from a single error message, you can type a question mark (?) in response to the Command Mode prompt that appears following the error message. In many cases, EDT replies with further information about the error you have made. In some cases, even more information can be obtained by typing another question mark. This feature is available in Command Mode only.

The EDT error messages are listed below alphabetically. Those messages that provide further information in response to a question mark contain that information as well. These messages appear in the list below as they would on your terminal.

Most of EDT error messages are self-explanatory. Those that are not are followed by a brief explanation.

An argument is required.
*?

You must specify a valid argument for this command.

Bad option format.
*?

An editor option is of the form /xx.

Bad option value in this context.
*?

The switch option value specified is invalid in context of its usage.

Bad Range Format.
*?

The range you specified is not legal.

Explanation: You attempted to specify a range in an unacceptable format. This error message can often result from mistyping of commands.

Buffer Specification Conflict.
*?

The command you typed implied more than one buffer specification.
*?

For instance, this may occur by using "N:M" in the same line as "%L".

Cannot mix "-" and value in option.
*?

An option cannot have a modifier as well as a value.

Cannot recognize terminal type.
*?

The terminal type you specified was unrecognizable.

*?

The legal terminal types are HCPY, LA30, LA36, VT05, VT50, VT52, VT55, VT61.

Cannot yet be performed on hard copy devices.
*?

The CHANGE command will not work on hard copy devices.

Command I/O error.

Explanation: An I/O error occurred while EDT was trying to read a command line.

Destination range must be contiguous.
*?

A MOVE or COPY command can have only one destination.

FATAL - Editor consistency error.

Explanation: An internal EDT error occurred. The input file is not affected, but no output file is created. EDT terminates.

FATAL - I/O error on work file.

Explanation: EDT encountered an I/O error while trying to read from or write to its internal work file. The input file is unaffected, but no output file is generated. EDT terminates.

FATAL - Open error on work file.

Explanation: EDT was unable to open its internal work file. The input file is unaffected, but no output file is generated. EDT terminates.

FATAL - Work file consistency error.

Explanation: The editor detected an inconsistency in its internal work file. This condition can result from a system problem or an internal EDT error. The input file is unaffected, but no output file is generated. EDT terminates.

FATAL - work file memory error.

Explanation: While trying to read from or write to a work file, EDT encountered a system memory error.

File name syntax error.
*?

The file name you specified does not have the correct format.

File name table exhausted.
*?

You have explicitly referenced more files than the editor tables can accommodate.

*?

The INCLUDE or RESTORE has exhausted tables internal to the editor.

File name wild cards not allowed.
*?

An explicit file must be specified to the editor.

File open error.
*?

The editor is unable to correctly open the requested file.

Illegal command option.

Explanation: You specified a command option that was not legal for the command you issued.

Illegal form for N:M.

Explanation: You gave a range specification in the form N:M that did not follow syntax rules.

Illegal line number.
*?

A line number must be in the range 1 to 65535.

Illegal option name modifier.
*?

An option name modifier can consist only of a "-" preceding the option.

Illegal string delimiters.
*?

The string delimiter may not be an alphanumeric character.

Illegal subcommand.

Explanation: This error message appears in Character Mode. The subcommand you issued was not in the correct format.

Illegal termination of command line.
*?

A line was terminated by other than an end of line or comment.

Illegal text buffer name.
*?

A text buffer name may consist only of from one to six alphanumeric characters.

*?

A text buffer name must be preceded by either '=' or '%BUF'.

Input record too big.
*?

The editor was unable to read the input record because of its size.
*?

The input record is longer than 255 characters in length.
*?

The editor treats this condition as an EOF condition on the input file.

I/O error on source file.

Explanation: EDT encountered a system I/O error while attempting to read from a file.

"%LAST" has no meaning in this context.

Missing string quote.
*?

A legal string must be surrounded by a pair of matching quote marks.
*?

A quoted string consists of one or more characters delimited by " or '. For instance, "ABC" or 'DEF'.

Nested XEQ not yet implemented.
*?

This version of the editor allows no XEQ command nesting.

No destination range.
*?

In MOVE or COPY, a destination range must always exist.

No input file specified.
*?

"INCLUDE" or "RESTORE" requires the /FI switch to specify the input file.

Non-contiguous range not allowed.
*?

The CHANGE command will work only on a contiguous range of lines.

No output file specified.
*?

The command you specified requires an output file.

No previous string entered.
*?

'' or "" cannot be used unless a previous string was specified.

No range specification allowed.
* ?
A range specification is not meaningful in SAVE or RESTORE.
* ?
SAVE and RESTORE only work on the entire state of the edit.

No repetition count allowed.

Explanation: This error message occurs in Character Mode. You attempted to specify a repetition count in a subcommand that does not accept a repetition count, for example, EL.

No room for buffer.
* ?
The number of buffers allowed by the editor has been exceeded.

No room in output file.
* ?
The device is full, the editor is unable to generate the output file.

No string-type allowed.

No such file.
* ?
The input file you requested does not exist.

No such line found.
* ?
The line referenced in the range specification could not be found.
* ?
The line may exist but could not be found in the range specified.

No such string in this range.
* ?
The string referenced in the range specification could not be found.
* ?
The string may exist but could not be found in the range specified.

Not a SAVE file.
* ?
Cannot RESTORE a non-SAVEed file.
* ?
RESTORE can only be used on a file which was generated by the SAVE command.

No value allowed.
* ?
A value has no meaning for this option.

No value specified.
* ?
This option requires a value.
* ?
An option value consists of a colon followed by the value.

Object-string required with "%ALL".
* ?
When typing "%ALL", a string must be specified as a search object.
* ?
The string specification must follow "%ALL".

Offset must be non-zero.
* ?
A range offset must be non-zero i.e. - it may be +1 or -1, but not +0 or -0.

Only one value allowed.
* ?
This option only allows one value to be specified.

Output file error.
* ?
The editor was unable to generate the output file without error.

Range must have at least one line.
* ?
Zero lines cannot be moved or copied.

Range not allowed here.
* ?
The command you typed does not allow a range specification.

Range required.
* ?
The XEQ command always requires an explicit range specification.

Ranges overlap.
* ?
The range specifications in the MOVE or COPY command cannot overlap.
* ?
The source and destination ranges must have no lines in common.

Repetition count cannot be zero or negative.

Explanation: In a range specification of the form n;i, i cannot be zero or a negative number.

RESTORE must be first command issued.
*?

No editing is allowed preceding a RESTORE command.

*?

The RESTORE must be the first command of an editing session, nothing may precede it.

String search cannot have a modifier.

Explanation: This error message occurs in Character Mode. A string search in Character Mode cannot be combined with or modified by a command.

String too long.

Explanation: This error message occurs in Character Mode. The string specified in a string search cannot exceed 62 characters in length.

"SUBSTITUTE NEXT" is not legal here.
*?

"SUBSTITUTE NEXT" can be used only after "SUBSTITUTE".

*?

"SUBSTITUTE NEXT" has no meaning unless a prior "SUBSTITUTE" command has been executed to establish the object and replacement strings.

The memory internal to the editor has overflowed.
*?

The editor's memory is full. No additional text will fit. Use WRITE if you desire to save what already exits.

The /QUERY switch cannot be used here.
*?

The XEQ command and /QUERY are incompatible.

The RESTORE command is not allowed in this context.
*?

When you invoked the editor, you gave a file specification. This specification prohibits subsequent usage of RESTORE.

*?

Input the QUIT command and then invoke the editor again without any file specification. You can then use the RESTORE command.

The /SEQ and /UN options are not supported for this command.
*?

These switches are not implemented for EXIT, PRINT, and WRITE.

This feature is not implemented yet.
*?

You are using a feature which is not supported by this version of EDT.

Unrecognized command.
*?

The command you typed is not a legal editor command.

Unrecognized option name.
*?

The option name you specified could not be recognized.

XEQ range non-contiguous.
*?

A range specification in XEQ must always be a contiguous block of lines.

You must specify a source range.
*?

The MOVE or COPY command requires a source range specification.

CHAPTER 3

LINE TEXT EDITOR (EDI)

EDI is a line-oriented editor that allows you to create and modify text files. EDI operates on most ASCII text files; it is frequently used to create and maintain FORTRAN or MACRO source files.

EDI accepts over 50 commands that determine its mode of operation and control its actions on input files, output files, and working text buffers. The commands fall into the following seven categories:

- Setup commands - select operating conditions, close and open files, select data modes.
- Locator commands - control the position of the current line pointer and thus determine which text line is acted upon.
- Text modification command - change text lines.
- Macro commands - define, store, recall, and use sequences of EDI commands.
- File input/output commands - transfer text to and from input, output, and save files.
- Device output commands - send output to terminal or printer.
- Close and exit commands - terminate editing operations.

Commands are categorized in this chapter as Basic EDI Commands (Section 3.2), EDI Commands: Function Summary (Section 3.3), EDI Commands: Detailed Reference Summary (Section 3.4). Restrictions, system device considerations, and error messages for these commands are discussed in Sections 3.5 and 3.6.

3.1 USING EDI

This section gives background information about the RSX-11M/M-PLUS Line Text Editor that is important for you to know before you read the command descriptions.

3.1.1 Invoking EDI

You can invoke EDI using any of the methods for invoking a utility described in Chapter 1. If any format except ">EDI filespec" is used, EDI issues the following prompting message:

```
EDI>
```

At this point, you must enter a file specification for the file to be edited.

Entering File Specifications

Enter file specifications in the following format:

```
dev:[ufd]filename.filetype;version
```

The abbreviation "filespec" is used throughout this chapter to denote a file specification that you supply.

If the file specification is a new file (that is, the file specified cannot be found on the specified device), EDI assumes that you wish to create a new file with the given file name. EDI then prints the following comment lines:

```
[CREATING NEW FILE]
INPUT
```

and enters input mode. (EDI control modes are described in Section 3.1.2.)

If the message FILE DOES NOT EXIST is displayed, it means that the user file directory corresponding to the specified UIC is nonexistent.

EDI does not accept indirect command file specifications.

If you specify an existing file name, EDI prints:

```
[000nn LINES READ IN]
PAGE 0]
*
```

and waits in edit mode for you to issue the first command.

If the ">EDI filespec" format is used, the prompt message (EDI>) is not issued and EDI starts up in either input or edit mode, depending on the file name specified -- input mode if the file name is new, and edit mode if the file name already exists.

At program startup, after EDI has identified the input file and created the output file, the program is ready for commands. In edit mode, the first line available for editing is one line above the top of the input file or the block buffer. Therefore, you can insert text at the beginning of the input file or the block buffer by issuing an INSERT command. To manipulate the first line of text, on the other hand, you must perform a NEXT operation to make that line available.

Defaults in File Specifications

EDI uses a default if any of the elements of the file specification, except input filename, is omitted. In general, EDI processing creates an output file. When you are modifying an existing file, EDI uses that file and your modifications to create an output file. When the edit session is complete, the output file usually has the same file specification as the input file, except the file system renumbers the version to one greater than the previous version. The default values for input and output files are listed in Table 3-1.

Table 3-1
EDI Default File Specifications

Element	Default Value for Input File	Default Value for Output File
dev:	SY0:	Same as input device
[ufd]	UFD under which EDI is currently running	Same as input [ufd]
filename	No default--must be specified	Same as input filename
.filetype	Null	Same as input filetype
;version	Latest version	Latest version+1

3.1.2 Control Modes: Input and Edit

EDI runs in two control modes:

- Edit mode (command mode)
- Input mode (text mode)

Edit mode is invoked automatically when you invoke an existing file.

In edit mode, EDI issues an asterisk (*) as a prompt. EDI accepts and acts upon commands and data strings to open and close files; to bring in lines of text from an open file; to change, delete, or replace information in an open file; or to insert single or multiple lines anywhere in a file.

Input mode is invoked automatically at program startup if you specify a nonexistent file.

When in input mode, EDI does not issue an explicit prompt. Lines that you enter at the terminal are treated as text and are inserted into the output file. When you complete each input line with a carriage return, EDI outputs a line feed to the terminal.

To switch from edit mode to input mode, enter the INSERT command followed by a carriage return. To return to edit mode, enter a carriage return as the only character on an input line. EDI will issue the * prompt, signifying edit mode.

3.1.3 Text Access Modes

EDI provides two modes you can use to access and manipulate lines of text in the input file. (A line is defined as a string of characters terminated by a carriage return.) The two modes are:

- Line-by-Line Mode -- Allows access to one line of text at a time; backing up is not allowed.
- Block Mode -- Allows free access within a block of lines, on a line-by-line basis. Backing up within a block is allowed; backing up to the previous block is not allowed.

Block mode is the default text access mode.

In addition to these two text access modes, EDI provides a way to process text in terms of "pages." This feature is described under the heading Processing Text in Pages, in this section.

Line-by-Line Mode

In this mode, a single line is the unit of the input file available for modification. Line-by-line mode is entered by issuing a BLOCK OFF command and is terminated by issuing a BLOCK ON command.

The single available line -- the "current" line -- is specified by a pointer, which you can move sequentially through the file, starting just before the first line in the file. You can manipulate the line pointer using the locator commands and the text modification and manipulation commands listed in Tables 3-5 and 3-6. However, you cannot easily direct the pointer backward within the file.

When you open a file at the beginning of an editing session, you can specify that the first line be brought into memory and made available for modification. This line remains in memory until you request that a new line be brought in. The pointer then moves down the file until the line you requested is encountered. That line is brought into memory and, as the current line, can be modified. When a new line is brought in, the previous line is written into the output file, as are all lines that may be passed over in reaching the new current line.

Once the pointer moves past a given line, that line is no longer accessible unless you enter a TOF or TOP command (described in Section 3.4). TOF causes the input and output files to be closed, and the output file to become the new input file. TOF also has the effect of ending line-by-line mode.

Block Mode

In this mode, a portion of the input file is held in a buffer for editing until you request that the contents of the buffer be added to the output file.

In block mode you can access lines of text backward as well as forward within the buffer. Thus, you can back up to a previously edited line without having to reprocess the entire block or file, and without having to issue a TOF command.

When you finish editing a block, you can write it out and read in the next block with the RENEW command. However, you cannot access a previously edited block except by using TOF.

EDI buffer space is computed dynamically at run time. The number of lines initially read into the buffer is computed by using the formula:

```
buffersize/132
```

A block is the number of lines read into the buffer by a RENEW or READ command. This number is either:

1. specified by the user by means of the SIZE command (default is 38 lines if the SIZE command is not issued),

or

2. determined by the presence of a form feed at a point in the text where the number of lines is less than that specified in the SIZE command (or its default value, if SIZE was not issued).

When the current line pointer reaches End-Of-Block, the message [*EOB*] is displayed and the current line pointer points to the last line in the block. To move the current line pointer to the top of the block, use TOP.

Table 3-2 provides a brief summary of the differences between line-by-line and block mode.

Table 3-2
Line-by-Line vs. Block Mode

Line-by-Line Mode	Block Mode
One line available for modification at a time.	Entire block of lines available for modification at a time, on a line-by-line basis.
Lines can only be accessed forward through the file.	Lines can be accessed forward and backward within a block.
Search commands can be applied to search the entire file.	Search commands search only the block in memory. To search more data, you must read in another block.

Regardless of editing mode the line pointer always points to the first character in the line.

Processing Text in Pages

EDI provides features that allow you to access portions of a text file by page. A "page" is a segment of text delimited by form feed characters (the last page in a file is terminated by the end-of-file marker).

Two commands are provided to handle paged text -- FF, which defines a page boundary by inserting a form feed; and PAGE, which accesses a page of text. (The commands PAGE FIND and PAGE LOCATE do not refer to form feed-delimited pages -- they are actually global searches.)

EDI handles paged text in block mode. If block mode is not already in effect, it is entered when you issue a PAGE command.

If a form feed is encountered in text during a READ or RENEW operation, the page thus delimited, for purposes of the READ or RENEW, is interpreted as a block.

The message [PAGE n], issued after a READ or RENEW operation, gives the value of EDI's page counter. If your text contains no form-feed characters, the count is zero until the last block in the file is read into the buffer. Upon encountering the end-of-file, EDI increments the page count to 1.

3.1.4 Text Files

The following sections describe how data may be added to files, and the operations performed on output files.

Input and Secondary Files

EDI accepts input from:

- The input terminal (that is, commands and text entries)
- Files-11 volumes that contain any of the following:
  1. The file to be edited
  2. A secondary file
  3. A save file
  4. A macro file

The input file is always preserved.[1] Any system failure, EDI failure, or lack of space on the output volume does not cause the loss of the input file. Only the output file is affected. In cases of failure, the output file is not completely destroyed; it becomes a truncated version of the input file containing all of the edits to the point of failure.

In general, the current block buffer is not written to disk when an error of this type occurs.

Output Files

The output device defaults to the input device, as do the directory, filename, and filetype, with the version number incremented by one.

If you wish to change any of these parameters (except device and directory), specify a completely new file specification when closing a file or exiting at the end of an EDI session.

---

[1] To delete the input file, use the CLOSE-AND-DELETE command or the EXIT-AND-DELETE command; or use PIP.

3.1.5 Terminal Conventions

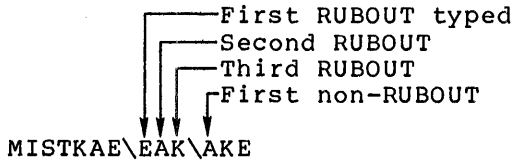
RSX-11 and EDI provide terminal keyboard functions that provide the means to:

- Delete characters on an input line (MCR function)
- Delete an entire input line (MCR function)
- Move the current line pointer forward in a file (EDI function)
- Move the current line pointer backward in a file (EDI function)
- Terminate an edit session and pass control to MCR (EDI function)

Character Erase (RUBOUT or DELETE; CTRL/R)

Typing the RUBOUT key (marked DELETE on some terminals) deletes individual characters if used before carriage return is pressed. During editing operations, RUBOUT does not affect previously prepared text.

When the RUBOUT key is typed it is echoed first as a backslash (\), and is followed by the previously typed character. Each successive RUBOUT typed results in the echo of an earlier typed character. When the first non-RUBOUT character is typed, it is echoed as a backslash (closing the RUBOUT sequence) followed by the typed character. For example:



For some CRT terminals, RUBOUT (or DELETE) works in a more obvious way. Each RUBOUT typed causes the cursor to backspace, erasing the previous character. Your CRT terminal may work this way if a certain option was selected when your system was generated.

Another useful system generation option is CTRL/R. If this option was selected, your system responds to CTRL/R by printing the incomplete input line. For example, at a hardcopy terminal you enter:

```
MISTKAE rubout rubout rubout CTRL/R
```

The echoed result is:

```
MISTKAE\EAK
MIST
```

Line Erase (CTRL/U)

CTRL/U deletes the line being input, if typed before the line is terminated with a carriage return. It is typed by holding down the CTRL key and pressing U. CTRL/U echoes as ^U followed by carriage return and line feed.

Carriage Return

The carriage return has the following effects, depending on how it is used:

- When issued in place of an input file specification, carriage return causes EDI to terminate.
- When issued in edit mode, carriage return causes the next line to be printed. That line becomes the current line.
- When issued in input mode as the only character in an input line, carriage return causes a return to edit mode.
- When issued alone after an INSERT command, carriage return invokes input mode.

ESCape or Altmode

When EDI is in edit mode, typing ESCape (or Altmode) causes the previous text line to be printed. That line becomes the current line. ESC can be used this way only in block mode, not in line-by-line mode.

CTRL/Z

CTRL/Z causes EDI to terminate. EDI writes the remainder of the input file into the output file and closes both files before terminating. Use CTRL/Z to terminate EDI in edit mode, input mode, or in the middle of an incomplete input line. CTRL/Z erases your last input line if you enter the command as a line terminator.

3.1.6 EDI Command Conventions

EDI uses asterisks (*) and ellipses (...) in special ways described in the following paragraphs. Also described below is the notation convention used to define EDI command abbreviations.

Use of Asterisk (*)

The asterisk character, *, can be used in place of any numeric argument. It evaluates to 32767 (decimal).

Example

The following command results in the printing of the remainder of the block buffer or file.

```
PRINT *
```

Use of Ellipsis (...) in Search Strings

In a number of the EDI commands, you must identify a string of characters to be located and/or changed. To reduce the necessary terminal entries, the more advanced user can employ the following special string constructs. In these special cases, the ellipsis (...) represents any number of intervening characters.

Case 1.	string1...string2	Any string that starts with string1, continues with any number of intervening characters, and ends with the first occurrence of string2.
Case 2.	...string	Any string that starts at the beginning of the current line and ends with the first occurrence of string.
Case 3.	string...	The first string that starts with string and ends at the end of the current line.
Case 4.	...	The entire current line.

Examples

In the following examples, the CHANGE command is used with the four cases of special string constructs given above. In each case the current line reads:

THIS IS A SAMPLE OF SPECIAL STRING CONSTRUCTS.

Case 1.

```
C /S A...E O/S AN EXAMPLE O
```

results in

THIS IS AN EXAMPLE OF SPECIAL STRING CONSTRUCTS.

Case 2.

```
C /...SPEC/THIS IS AN EXAMPLE OF SPEC
```

results in

THIS IS AN EXAMPLE OF SPECIAL STRING CONSTRUCTS.

Case 3.

```
C /STRING.../EDI STRING CONSTRUCTS.
```

results in

THIS IS A SAMPLE OF SPECIAL EDI STRING CONSTRUCTS.

Case 4.

```
C /.../EXAMPLES OF SPECIAL EDI CONSTRUCTS.
```

results in

EXAMPLES OF SPECIAL EDI CONSTRUCTS.

Command Abbreviations

EDI permits the use of abbreviated commands. Where these shorter forms are allowed, the command format specifications represent the shortest acceptable form in upper-case letters. Lower-case letters may be entered optionally. The following example shows the abbreviations allowed for the VERIFY command. The command format specification is:

```
Verify
```

The following abbreviations are valid for the VERIFY command:

```
V
VE
VER
VERI
VERIF
VERIFY
```

3.2 BASIC EDI COMMANDS

The basic EDI commands listed in Table 3-3 allow you to create a file, to modify a file by adding, deleting, or changing its contents, and to exit after the desired operations have been completed. More detailed description of each command follows the table. These commands are the most important EDI commands. As you become more familiar with EDI operations, the additional commands listed in Section 3.3 and described in Section 3.4 will allow you to use all of EDI's capabilities.

Table 3-3
Basic EDI Commands

Command	Command Format	Description
ADD	Add string	Append string to current line.
ADD & PRINT	AP string	Append string to current line, and print resultant line.
BOTTOM	BOttom	Move the current line pointer to the bottom of the current block (in block mode) and to the bottom of the file (in line-by-line mode).
CHANGE	[n]Change /string1/ string2[/]	Replace string1 with string2 n times in the current line.
CTRL/Z	Type a control Z	Close files and terminate editing session.
DELETE	Delete [n] or Delete [-n]	Delete current line and n-1 lines if n is (+); delete n lines preceding current line if n is (-). [-n] operates in block mode only.
DELETE & PRINT	DP [n] or DP [-n]	Same as DELETE, except new current line is printed.
<ESCape>	Type the ESC (or ALT) key	Print previous line, and make it new current line (block mode only). Same as NP-1.
EXIT	EXit [filespec]	Close files, rename output file, and terminate editing session.
INSERT	INsert [string]	Enter the string immediately following the current line. If no string is specified, EDI enters input mode.
LOCATE	[n]Locate string	Locate nth occurrence of string. In block mode, search stops at end of current block.

(continued on next page)

Table 3-3 (Cont.)
Basic EDI Commands

Command	Command Format	Description
NEXT	Next [n] or Next [-n]	Establish new current line n lines away from current line.
NEXT & PRINT	NP [n] or NP [-n]	Establish and print new current line.
PRINT	Print[n]	Print current line and the next n-1 lines. The last printed line is the new current line.
RENEW	RENew[n]	Write current block to output file and read new block from input file (block mode only).
<RETURN>	Carriage return	Print the next line, make it new current line; exit from input mode. Same as NP+1.
RETYPE	Retype string	Replace current line with string; or delete current line if string is null.
TOP	Top	Move the current line pointer to the top of the current block (in block mode) or top of file (in line-by-line mode). TOP creates a new version of the file each time it is invoked in line-by-line mode.
TOP OF FILE	TOF	Return to top of input file and save all pages previously edited. TOF creates a new version of the file each time it is invoked. TOF reads in a new block after writing the previous block to the output file.

3.2.1 ADD

This command causes the specified string to be appended to the current line.

Format

```
Add string
```

Example

The following command completes the line HAPPY DAYS ARE HERE

```
*A AGAIN.
```

Note that the space after the A is the command terminator. EDI will not insert the space into the line. If a space is to precede AGAIN., the command should be:

```
A<space><space> AGAIN.
```

3.2.2 ADD & PRINT

This command performs the same function as the ADD command, except that the new line is printed.

Format

AP string

Example

Using the same line as the ADD command, the following command causes the new line to be printed as follows:

```
*AP AGAIN.
HAPPY DAYS ARE HERE AGAIN.
```

3.2.3 BOTTOM

BOTTOM moves the current line pointer to the beginning of the last line of the current block (in block mode), or to the beginning of the last line of the file (in line-by-line mode). In block mode, the only processing EDI performs is line pointer positioning. In line-by-line mode, all the lines are copied from the input file to the output file until EOF is reached. If VERIFY ON is specified, the last line of the file block is displayed. Note, however, that if you deleted the last line before you issued BOTTOM, the pointer will be located past the text, and thus the last line will not be printed. BOTTOM performs the same function as END (see Section 3.4.14).

Format

BOttom

Example

```
*V ON
*BO
THIS IS THE LAST LINE
```

In this example, the current line pointer is moved to the bottom of the block buffer and the last line is displayed.

3.2.4 CHANGE

This command searches for string1 in the current line and, if found, replaces it with string2. If string1 is given but cannot be located in the current line, EDI prints [NO MATCH] and returns an * prompt. If string1 is null (not given), string2 is inserted at the beginning of the line. If string2 is null, string1 is deleted from the current line.

The search for string1 begins at the beginning of the current line and proceeds across the line until a match is found.

The characters that delimit string1 and string2 are normally slashes (/). However, any matching characters not contained in the specified string may be used. The first character following the command is the beginning delimiter; the next matching character ends the string. Thus, characters used as delimiters must not appear in the string itself. The closing delimiter is optional.

If you precede the command with a number n, the first n occurrences of string1 are changed to string2. After each replacement of string1 with string2, scanning restarts at the first character in the line. This allows you to generate a string of characters as shown in the example below.

If no match occurs, a [NO MATCH] message is displayed.

Format

```
[n]Change /string1/string2[/]
```

Example

```
TO SEPERATE THE THOUGHTS, USE SEPERATE SENTENCES.

   2C/SEPE/SEPA/

TO SEPARATE THE THOUGHTS, USE SEPARATE SENTENCES.
```

3.2.5 <CTRL/Z>

Typing CTRL/Z (holding the CTRL key down while typing the letter Z) terminates the editing session. If an output file is open when CTRL/Z is typed, all remaining lines in the block buffer and the input file are transferred (in that order) into the output file, all files are closed, and EDI exits. These actions occur whether EDI is in edit or input mode. If EDI is prompting for another file specification when CTRL/Z is entered, all files are closed (including any open secondary input file), and EDI exits. If you enter CTRL/Z as an input line terminator, that line is erased.

3.2.6 DELETE

This command causes lines of text to be deleted in the following manner:

1. If n is given and is a positive number, the current line and n-1 following lines are deleted. The new current line is the line following the last deleted line.

2. If n is given and is a negative number, the current line is not deleted, but the specified number of lines that precede it are deleted. The line pointer remains unchanged. A negative value for n can be used only in block mode.

3. If n is null, the current line is deleted, and the next line becomes the new current line.

Format

```
Delete [n]
   or
Delete [-n]
```

Example

To delete the previous five lines in the block buffer, type the following command:

```
*D -5
```

3.2.7 DELETE & PRINT

This command performs the same function as the DELETE command, except that the new current line is printed when all lines have been deleted.

Format

```
DP [n]
  or
DP [-n]
```

If n is not specified, +1 is assumed. A negative value for n can be used only in block mode.

Example

If the following lines are contained in a file:

```
THIS IS LINE 1
THIS IS LINE 2
THIS IS LINE 3
THIS IS LINE 4
```

and the line pointer is at the first line, the following command obtains the results shown below:

```
*DP 2
THIS IS LINE 3
```

3.2.8 <ESCape>

This command prints the previous line in the block (block mode only). That line becomes the current line. Thus, you can back up through a block, one line at a time, by typing a series of ESCapes. Typing ESCape is equivalent to typing NP-1 (NEXT & PRINT command).

3.2.9 EXIT

This command transfers all remaining lines in the block buffer and input file (in that order) into the output file, closes the files, and terminates the editing session. If a file specification is used, the output file is renamed to the specified filename.

Format

EXit [filespec]

Example

The command:

*EX

terminates the editing session without renaming the output file. It causes EDI to display:

[EXIT]

The output filename.filetype is the same as the input filename.filetype, with a version number one greater than that of the input file.

3.2.10 INSERT

This command inserts string immediately following the current line. The string becomes the new current line. If string is not specified, EDI enters input mode.

Format

Insert [string]

Example

*I TEXT INSERT IN EDIT MODE	Insert a line of text immediately after the current line.
*I TEXT INSERT 1 IN INPUT MODE TEXT INSERT 2 IN INPUT MODE ETC.	An I followed by a carriage return causes EDI to switch to input mode. A series of new lines may be input following the current line.
*	A carriage return as the only character in an input line causes EDI to return to edit mode and to prompt for a new command.

3.2.11 LOCATE

This command causes a search for the pattern string, beginning at the line following the current line. The string may occur anywhere in the line sought. The line pointer is positioned to the line containing the match. When the line is located, it is printed if VERIFY ON is in effect.

LOCATE applies to the block buffer if EDI is in block mode and to the input file if in line-by-line mode.

Format

[n] Locate string

Example

The following command can be used to locate the line HAPPY DAYS ARE HERE AGAIN.

```
*L PPY
```

EDI searches the file or block buffer and (if VERIFY ON is specified) prints the line when it is located. The current line pointer is set to the located line.

If string is not specified, the line following the current line is considered a match, and the line pointer is positioned there. If "n" is specified, the nth occurrence of string is located.

3.2.12 NEXT

This command moves the current line pointer backward and forward in the file. A positive number moves the current line pointer n lines beyond the current line; a negative number moves the current line pointer backward n lines.

Format

```
Next [n]
   or
Next [-n]
```

If n is not specified, a value of +1 is assumed. A negative n can be used only in the block mode.

Example

In the block mode, the following command moves the current line pointer back five lines:

```
*N -5
```

3.2.13 NEXT & PRINT

This command has the same effect as the NEXT command, except that the new current line is printed.

Format

```
NP [n]
  or
NP [-n]
```

The following conventions can be used in place of issuing a complete NP command:

Pressing the carriage return key is the same as an NP+1 command.

Pressing the ESCape (or ALTmode) key while in the block mode is the same as an NP-1 command.

If n is not specified, then a value of +1 is assumed.

Example

Assume the following four lines are contained in the file and the line pointer is at the first line.

```
LINE 1 OF THE FILE
LINE 2 OF THE FILE
LINE 3 OF THE FILE
LINE 4 OF THE FILE
```

If the following command is issued, EDI returns the following printout:

```
*NP 2
LINE 3 OF THE FILE
*
LINE 4 OF THE FILE
* <ESC>
LINE 3 OF THE FILE
* <ESC>
LINE 2 OF THE FILE
```

3.2.14 PRINT

This command prints out the current line and the next n-1 lines on the terminal; the last line printed becomes the new current line. If it is not specified, a value of 1 is assumed.

Format

```
Print [n]
```

Example

The following example illustrates both the PRINT and the TYPE commands:

Before


```
*TYPE 5
Line A
Line B
Line C
Line D
Line E
*
```

```
*PRINT 5
Line V
Line W
Line X
Line Y
Line Z
*
```

After

3.2.15 **RENEW**

This command writes the current block buffer into the output file and reads a new block from the input file. The optional value n is a repetition count: if you specify n, the process is repeated n times. The intermediate blocks are written into the output file, and the last block is left in the block buffer. If n is not specified, a single RENEW is performed. This command may be used only in block mode. Refer to Section 3.1.3 for information on how EDI block buffers are processed.

Format

```
RENew [n]
```

Example

```
*RENEW 10
```

Ten consecutive blocks are transferred from the input file to the block buffer. The initial contents of the block buffer and the next nine blocks are transferred to the output file. The current line pointer points to the first line in the tenth block, which is currently in the block buffer.

3.2.16 <RETURN>

In edit mode, this command prints the next line in the file or block buffer. That line becomes the current line. Thus, you can scan through a file or block, one line at a time, by typing a series of RETURNs. This command is equivalent to NP+1 (NEXT & PRINT command).

In input mode, a single carriage return causes EDI to return from input mode to edit mode.

3.2.17 RETYPE

This command replaces the current line with string. If string is not specified, the line is deleted.

Format

```
Retype [string]
```

Example

```
*R THIS IS A NEW LINE
```

The string THIS IS A NEW LINE replaces the current line.

3.2.18 TOP

TOP sets the current line pointer to the top of the current block (in block mode) or to the top of the file (in line-by-line mode). When the current line pointer is positioned via TOP, you can enter lines preceding the first line in the block or file.

TOP differs from TOF in the following ways:

- In line-by-line mode, TOP creates a new file and moves the current line pointer to the top of the file. Unlike TOF, it does not cause EDI to return to block mode.
- In block mode, TOP moves the current line pointer to the top of the current block and does not create a new output file. TOF moves the current line pointer to the top of the file and creates a new output file.

Format

```
Top
```

Example

```
*TOP
```

This command directs the current line pointer to the top of the current block in block mode.

3.2.19 TOP OF FILE (TOF)

This command creates a new version of the file and returns the current line pointer to the first line of the file. TOF processing copies the input file into the output file, closes both, then opens the latest version of the file as the input file. If you issue this command when in line-by-line mode, EDI switches to block mode after saving the edited data. The first block is read into the block buffer.

Format

```
TOF
```

Example

```
*TOF
```

This command writes the previously edited pages into the output file, resets the current line pointer to the top of the input file, and reads the first block into the block buffer.

3.3 EDI COMMANDS: FUNCTION SUMMARY

EDI commands can be arranged by functional similarity; for example, all the commands you use to locate a string can be grouped under the function heading "Locator Commands." This section contains summaries of the following command categories:

- Setup commands - select operating conditions, close and open files, select data modes.
- Locator commands - control the position of the current line pointer, and thus determine which text line is acted upon.
- Text modification commands - change text lines.
- Macro commands - define, store, recall, and use sequences of EDI commands.
- File input/output commands - transfer text to and from input/output, and save files.
- Device output commands - send output to terminal or printer.
- Close and exit commands - terminate editing operations.

3.3.1 Setup Commands

The setup commands allow you to enable or disable certain special features of EDI. Among these features are the block and line-by-line text access modes, already mentioned, and the automatic verification of LOCATE commands. Setup commands are listed in Table 3-4.

Table 3-4
EDI Setup Commands

Command	Format	Description
BLOCK ON/OFF	BLock [ON] or BLock OFF	Switch text access modes.
CONCATENATION CHARACTER	CC [letter]	Change concatenation character to specified character (default is &).
OPEN SECONDARY	OPens filespec	Open specified secondary file.
OUTPUT ON/OFF	OUtput ON or OUtput OFF	Continue or discontinue transfer to output file (line-by-line mode).
SELECT PRIMARY	SP	Reestablish primary file as input file.
SELECT SECONDARY	SS	Select opened secondary file as input file.
SIZE	SIZE n	Specify maximum number of lines to be read into block buffer.
TAB	TAb [ON] or TAb OFF	Turn automatic tabbing on or off.
UPPER CASE ON/OFF	UC [ON] or UC OFF	Enable or disable conversion of lower-case characters entered from terminal to upper-case characters.
VERIFY ON/OFF	Verify [ON] or Verify OFF	Select whether locator and change commands are verified.

3.3.2 Locator Commands (Line-Pointer Control)

During editing operations, EDI maintains a pointer that identifies the current line (that is, the line to which any subsequent editing operations refer). Commands that modify the line pointer's location are called locator commands; they are listed in Table 3-5.

The locator commands allow you to:

- Set the line pointer to either the top or bottom of the input file or block buffer.
- Move the line pointer a specified number of lines away from its current position.
- Move the line pointer to a line containing a given text string.

In edit mode, the carriage return and ESCape (or ALTmode) keys act to relocate the line pointer. A single carriage return moves the pointer to the next line. A single ESCape moves the line pointer back one line (in block mode only). In each case the line is printed.

If VERIFY ON is in effect, the located line is printed after a BOTTOM, END, FIND, PAGE FIND, PAGE LOCATE, or SEARCH & CHANGE command.

Table 3-5
EDI Locator Commands

Command	Format	Description
BEGIN or TOP	Begin Top	Set current line to the line preceding top line in file (line-by-line mode) or block buffer (block mode). Both commands create copies of the file each time they are invoked in line-by-line mode. The commands are equivalent.
BOTTOM or END	BOttom End	Set current line to last line in file or block buffer. The commands are equivalent.
<ESCape> (or ALTmode)	Type ESC (or ALT) key	Print previous line and make it new current line. (Block Mode only.)
FIND	[n]Find [string]	Search current block or input file, beginning at line following current line for the nth occurrence of string. String must begin in column 1. Set line pointer to located line.
LOCATE	[n]Locate string	Locate nth occurrence of string. In block mode, search stops at end of block.
NEXT	Next [n] Next [-n]	Establish new current line n lines away from current line.
NEXT & PRINT	NP [n] or NP [-n]	Establish and print new current line.
PAGE (Block Mode Only)	PAGe n	Enter block mode. Read page n into block buffer. If n is less than current page number, do TOF first. Pages are delimited by form feed characters.
PAGE FIND (Block Mode Only)	[n]PFind string	Search successive blocks for the nth occurrence of string. String must start in column 1.
PAGE LOCATE (Block Mode Only)	[n] PLocate string	Search successive blocks for the nth occurrence of string. String may occur anywhere in line.
<RETURN>	Carriage return	Print the next line, make it the current line; exit from input mode.
SEARCH & CHANGE	SC /string1/string2[/]	Locate string1 and replace it with string2.

3.3.3 Text Modification and Manipulation Commands

The text modification and manipulation commands enable you to modify text. Table 3-6 lists these commands.

Table 3-6
EDI Text Modification and Manipulation Commands

Command	Format	Description
ADD	Add string	Append string to current line.
ADD & PRINT	AP string	Append string to the current line and print resultant line.
CHANGE	[n]Change/string1/ string2[/]	Replace string1 with string2 in the current line n times.
DELETE	Delete [n] or Delete [-n]	Delete current line and n-1 lines if n is (+); delete n lines preceding current line if n is (-). [-n] operates in block mode only.
DELETE & PRINT	DP [n] or DP [-n]	Same as DELETE except new current line is printed.
ERASE	ERASE [n]	Erase the current line if in line-by-line mode. Erase the current block buffer and the next (n-1) blocks, if in block mode.
FORM FEED	FF	Insert form feed into block buffer (used to delimit a page).
INSERT	Insert string	Enter string following current line, or enter input mode if string is not specified.
LINE CHANGE	[n]LC /string1/ string2[/]	Change all occurrences of string1 in current line (and n-1 lines) to string2.
OVERLAY	Overlay [n]	Delete n lines, enter input mode, and insert new line(s) as typed, in place of original line(s).

(continued on next page)

Table 3-6 (Cont.)
EDI Text Modification and Manipulation Commands

Command	Format	Description
PASTE	PAste /string 1/ string2[/]	Search all remaining lines in fiel or block buffer for string1 and replace with string2.
RETYPE	Retype	Replace the current line with string; or delete the current line if string is null.
TOP OF FILE	TOF	Return to the top of the input file and save all pages previously edited.
UNSAVE	UNSave [filespec]	Insert all lines from specified file following current line. If filespec is not specified, SAVE.TMP is used.

3.3.4 **Macro Commands**

These commands allow you to define, store, recall, and use macros. A macro is a series of EDI commands that, once defined, can be executed repeatedly using just a few keystrokes. Table 3-7 lists the macro commands.

Table 3-7
EDI Macro Commands

Command	Format	Description
MACRO	MACRO x definition	Define macro number x. Up to three macros can be defined.
MACRO CALL	MCall	Retrieve macro definitions stored in file MCALL;n.
MACRO EXECUTE	[n]Mx [a]	Execute macro x [n] times, while passing numeric argument [a]. The value x may be 1, 2, or 3.
MACRO IMMEDIATE	[n] <definition>	Define and execute a macro n times. Store it as macro number 1.

3.3.5 File Input/Output Commands

Input/output commands control the movement of text to and from input/output, and save files. Table 3-8 lists these commands.

Table 3-8
EDI Input/Output Commands

Command	Format	Description
FILE	FILe filespec	Transfer lines from input file to both the output file and the specified file until a form feed or end-of-file is encountered. (Line-by-line mode only.)
READ	REAd [n]	Read next n blocks of text into block buffer. If buffer contains text, new text is appended to it.
RENEW	RENew [n]	Write the current block to the output file and read new block from the input file.
SAVE	SAve [n] [filespec]	Save current line and the next n-1 lines in the specified file. If filespec is not given, lines are saved in file SAVE.TMP. SAVE puts the temporary file in the UFD and on the device of the file you are editing. You can override the default by specifying a different device and UFD.
WRITE	Write	Write contents of block buffer to output file and erase block buffer.

3.3.6 Device Output Commands

These commands direct output to your terminal or to a pseudo-device (CL:). They are listed in Table 3-9.

3.3.7 CLOSE and EXIT Commands

The CLOSE and EXIT commands terminate EDI operations and write the remainder of the input file into output file. Table 3-10 lists these commands.

Table 3-9
EDI Device Output Commands

Command	Format	Description
LIST ON TERMINAL	LIst	Print on the terminal all lines remaining in block buffer (block mode) or input file (line-by-line mode), beginning at current line.
LIST ON PSEUDO-DEVICE	LP	Same as LI, except that printing is performed on pseudo-device CL:.
PRINT	Print [n]	Print the current line and the next n-1 lines. The last printed line is the new current line.
TYPE	TYpe [n]	Print next n lines. In line-by-line mode, identical to PRINT command. In block mode, line pointer remains at current line, unless end-of-block was reached.

Table 3-10
EDI Close Operation Commands

Command	Format	Description
CLOSE	CLose [filespec]	Transfer remaining lines in block buffer and input file, to output file and close files. If file specification is used, output file is renamed. EDI> prompt is issued.
CLOSE SECONDARY	CLOSES	Close secondary file.
CLOSE & DELETE	CDl [filespec]	Same as CLOSE, except that input file is deleted. EDI> prompt is issued.
CTRL/Z	Type a control Z	Close files, and terminate edit session.
EXIT	EXit [filespec]	Close files, rename output file, and terminate edit session.
EXIT & DELETE	EDx [filespec]	Transfer remaining lines in block buffer and input file to output file, and close file. Rename file if file specification is given. Delete input file and terminate EDI.
KILL	KILL	Close input and output files, delete output file. EDI> prompt is issued.

3.4 EDI COMMANDS: DETAILED REFERENCE SUMMARY

This section lists each EDI command in alphabetical order. Each command description comprises the function of the command and the command format. Most descriptions include examples and usage information. The exceptions are the basic commands, which are described in detail in the preceding section. In this section, only the function and format of basic commands are described.

3.4.1 ADD

ADD causes the specified string to be appended to the current line.

Format

```
Add string
```

For examples and information describing how to use ADD, refer to Section 3.2.1.

3.4.2 ADD & PRINT (AP)

ADD & PRINT performs the same function as ADD, except that the new line is printed.

Format

```
AP string
```

For examples and information describing how to use ADD & PRINT, refer to Section 3.2.2.

3.4.3 BEGIN

BEGIN sets the current line pointer to the beginning of the file in line-by-line mode, or to the beginning of the block buffer in block mode. The current line is one line preceding the top line in the file or block buffer. Thus, you can insert text at the beginning of a file or block.

If EDI is in line-by-line mode, BEGIN copies the input file into the output file, closes both, then opens the latest version of the file. BEGIN performs the same function as TOP.

Format

```
Begin
```

Example

```
*B
```

In this example, the current line pointer is moved to the top of the block buffer (block mode is assumed).

3.4.4 BLOCK ON/OFF

This command allows you to switch between block mode and line-by-line mode. When you enter BLOCK ON, block mode becomes active, and the next block of text is brought into the block buffer. When you enter BLOCK OFF, the current block being processed is written to the output file, and line-by-line mode becomes active. The first line from the next sequential block in the input file becomes the current line.

If you enter an unnecessary BLOCK command (for example, BLOCK ON when EDI is already in block mode), the command is ignored.

BLOCK ON is the default text access mode. It is assumed when neither ON nor OFF is specified.

Format

```
BLock [ON]
    or
BLock OFF
```

Example

```
*BLOCK ON
```

This command causes EDI to switch to block mode. The next block of text is read into the block buffer.

3.4.5 BOTTOM

BOTTOM sets the current line pointer to the beginning of the last line of the block (in block mode) or of the input file (in line-by-line mode).

Format

```
BOttom
```

For examples and information on how to use BOTTOM, refer to Section 3.2.3.

3.4.6 CHANGE

CHANGE searches for string1 in the current line and, if found, replaces it with string2.

Format

```
[n] Change /string1/string2[/]
```

For examples and information on how to use CHANGE, refer to Section 3.2.4.

3.4.7 CLOSE

This command transfers all remaining lines in the block buffer and input file (in that order) into the output file, and closes both files. If a file specification is included, the output file is

renamed to the specified file. EDI then returns to its initial command sequence, prompts with EDI>, and waits for you to type another file specification.

If a secondary file was opened during the editing session and was not closed, it remains open.

Format

CLose [filespec]

Example

```
*CL
EDI>
```

This command closes both input and output files, and EDI returns to the initial command sequence.

3.4.8 CLOSE SECONDARY (CLOSES)

Use this command when you have finished extracting text from a secondary input file. You must enter CLOSES before you can use another secondary file as input.

Format

CLOSES

3.4.9 CLOSE & DELETE (CD)

This command transfers all remaining lines in the block buffer and the input file (in that order) into the output file, and closes both files. The input file is then deleted. If a file specification is included, the output file is renamed to the specified file. This command acts like CLOSE, except that the input file is deleted.

If a secondary file was opened during the editing session and was not closed, it remains open.

Format

CDl [filespec]

3.4.10 CONCATENATION CHARACTER (CC)

The concatenation character allows you to give commands on one input line. By default, the concatenation character is &. To reference text containing an & (for example in LOCATE or CHANGE commands), you must change the concatenation character to some other character.

If the CC command is used without an argument, the concatenation character is changed to &.

Format

CC [letter]

Example

```
*CC :
*L A&B:C /A&B/ABC/
CONCATENATION TEST CONTAINING A&B.
CONCATENATION TEST CONTAINING ABC.
*CC
```

In this example, the string to be located contains an ampersand. Therefore, the concatenation character must be changed to a different character before EDI can locate the line.

The first command line changes the default concatenation character from & to :.

The second command line instructs EDI to locate the string A&B and change that string A&B to ABC. (Note: this line contains two commands that are concatenated by the new concatenation character, :.)

The third command line changes the concatenation character back to the normal default value, &.

3.4.11 CTRL/Z

CTRL/Z is an MCR function that terminates the edit session. You enter CTRL/Z by pressing the CTRL key and the Z character key at the same time. For usage information on CTRL/Z, refer to Section 3.2.5.

3.4.12 DELETE

DELETE deletes a specified number of lines from a file.

Format

```
Delete n
```

For examples and information on how to use DELETE, refer to Section 3.2.6.

3.4.13 DELETE & PRINT (DP)

DELETE & PRINT performs the same function as DELETE, except that it displays the new current line after the specified lines are deleted.

Format

```
DP n
```

For examples and information on how to use DELETE & PRINT, refer to Section 3.2.7.

3.4.14 **END**

END sets the current line pointer to the beginning of the last line of the block or input file. If EDI is in block mode, only line pointer positioning occurs. In line-by-line mode, all lines are copied from the input file to the output file until EOF is reached. The last line in the block or file is displayed if VERIFY ON is in effect. Note, however, that if the last line was deleted before issuance of END, the pointer will be located past the text, and thus the last line will not be printed. END performs the same function as BOTTOM.

Format

End

Example

```
*V ON
*END
THIS IS THE LAST LINE
```

This command moves the current line pointer to the bottom of the block buffer (block mode is assumed).

3.4.15 **ERASE**

In line-by-line mode, this command erases the current line. In this mode n may only be 1. In block mode, this command erases the current block buffer and the next n-1 blocks. If n is not specified, +1 is assumed.

Format

ERASE [n]

Example

```
*ERASE 5
```

This command causes the contents of the current block buffer and the next 4 blocks to be erased. These blocks are not written into the output file.

3.4.16 **<ESCape>**

This command prints the previous line in the block (block mode only). That line becomes the current line. Thus, you can back up through a block, one line at a time, by typing a series of ESCapes. Typing ESCape is equivalent to typing NP-1 (NEXT & PRINT command).

3.4.17 **EXIT**

EXIT writes all remaining records to the output file, closes the files, and terminates the edit session.

Format

EXIT [filespec]

For examples and information on how to use EXIT, refer to Section 3.2.9.

3.4.18 **EXIT & DELETE (ED)**

This command functions in the same way as the CLOSE & DELETE command, except that EDI exits when finished.

Format

EDx [filespec]

Example

```
*EDX NEWFILE.DOC
[EXIT]
>
```

3.4.19 **FILE**

This command -- legal in line-by-line mode only -- transfers lines from the input file to both the output file and a specified file, beginning with the current line, until a form feed character is encountered as the first character in a line or until end-of-file is reached. At that time the specified file is closed. The form feed character is not included in the specified file. During the transfer, the original file remains intact (i.e., all lines written to the specified file are also written to the normal output file, including the form feed). When the command is complete, the current line in the input file is one line beyond the form feed.

BLOCK OFF must be in effect for FILE to work properly.

If the specified file does not already exist, a new file is created. If the specified file does exist, the latest version of the file contains the new data.

Format

FILe filespec

Example

```
*FIL SEC.DAT
```

EDI writes the contents of the input file, from the current line to the end, into both the output file and the file SEC.DAT.

3.4.20 **FIND**

This command searches the block buffer or input file for a string, beginning at the line following the current line. The string must begin in column 1 of the line matched. The line pointer is positioned at the line containing the match. When the line containing the string is found, it is printed if VERIFY ON is in effect.

FIND applies to the block buffer if EDI is in block mode and to the input file if EDI is in line-by-line mode.

If a string is not specified, the line following the current line is considered a match. If n is specified, the nth occurrence of the string is found.

Format

[n]Find [string]

Example

```
*V ON
*F LOOK

LOOK AT THE FIRST CHARACTER IN THE LINE.
```

In this example, EDI searches the block buffer (or file) for a line that begins with LOOK and prints the line when it is found.

3.4.21 FORM FEED (FF)

This command allows you to insert form feeds into the text, in order to delimit EDI pages. The form feed is inserted after the current line, and the new current line becomes the line containing the form feed.

Format

FF

Example

```
*P
THIS IS THE LAST LINE ON THE PAGE
*FF
```

In this example, a form feed is inserted into the text following the current line.

3.4.22 INSERT

INSERT inserts a string immediately following the current line. The string becomes the current line.

Format

Insert [string]

For examples and information on how to use INSERT, refer to Section 3.2.10.

3.4.23 KILL

This command returns EDI to the initial command sequence without retaining the output file. When this command is executed, the input file is closed, and the output file is deleted.

Format

KILL

Example

```
*KILL
EDI>
```

In this example, the output file is deleted, and EDI displays the prompt:

```
EDI>
```

At this point, you can return control to MCR by means of CTRL/Z or enter a file specification for a file to be edited.

3.4.24 LINE CHANGE (LC)

This command is similar to CHANGE, except that all occurrences of string1 in the current line are changed to string2. A numeric value n preceding the command changes the current line and the next n-1 lines. If string2 is null, all occurrences of string1 are deleted. New lines are printed if the VERIFY ON command is in effect.

If string1 is given but EDI cannot locate the string in the current line, EDI prints [NO MATCH] and returns an * prompt.

Format

```
[n]LC /string1/string2[/]
```

Example

If the current line is:

```
THES ES THE LINE TO BE ESSUED.
```

The following command would correct the errors:

```
*V ON
*LC /ES/IS
THIS IS THE LINE TO BE ISSUED
```

3.4.25 LIST ON TERMINAL (LI)

This command prints on your terminal all remaining lines in the block buffer (block mode) or all remaining lines in the input file (line-by-line mode), beginning at the current line. At the end of the listing, the current line pointer is repositioned to the top of the input file or block buffer.

If terminal host synchronization is installed at system generation, you can control printing functions using CTRL/O, CTRL/S, and CTRL/Q. To suppress printing at any point, type CTRL/O. Printing can be suspended temporarily with CTRL/S, and resumed with CTRL/Q.

Format

```
LIst
```

Example

```
*LI
```

This command causes all remaining lines in the block buffer or all remaining lines in the input file to be printed on the terminal.

3.4.26 LIST ON PSEUDO-DEVICE (LP)

This command functions in the same manner as the LIST ON TERMINAL command, except that the remaining lines in the block buffer (block mode) or the remaining lines of the input file (line-by-line mode) are listed on the pseudo-device CL:. In most systems, CL: is set to the system line printer.

Format

```
LP
```

Example

```
*LP
```

This command causes all remaining lines in the block buffer or all remaining lines in the input file to be printed on the pseudo-device CL:.

3.4.27 LOCATE

LOCATE searches for a string beginning at the line following the current line. The string can occur anywhere in the lines searched.

Format

```
[n] Locate string
```

For examples and information on how to use LOCATE, refer to Section 3.2.11.

3.4.28 MACRO

This command is used to define macros. Space is available for three macro definitions. The definition portion can be any legal EDI command or string of legal EDI commands connected by the concatenation character.

If a numeric argument is to be passed to the macro at execution time, a percent sign (%) must be inserted in the macro definition at the point where the numeric argument is to be substituted. Then the value passed via the MACRO EXECUTE command replaces the percent sign when the macro is executed.

A MACRO definition may contain more than one percent sign. If it does, the single numeric value given in a MACRO EXECUTE command replaces each percent sign. However, a macro may not have two or more independent arguments.

Format

MACRO x definition

x

Specifies the macro number (1, 2, or 3).

Examples

To find the nth occurrence of the string ABC in the current block and replace that occurrence and all remaining occurrences within the block with the string DEF, the following macro could be used:

```
*MACRO 1 %L ABC&PA /ABC/DEF
```

The following command executes the macro and searches for the tenth and succeeding occurrence of ABC.

```
*M 1 10
```

The following macro definition and subsequent invocation could be used to change all occurrences of the strings ABC and GHI to DEF and JKL, respectively. The substitution is made in the current block and the next four blocks (five blocks in all).

```
*MACRO 1 PA /ABC/DEF/&PA /GHI/JKL/&RENEW   (MACRO command)
*5M 1                                      (MACRO EXECUTE command)
```

3.4.29 MACRO CALL (MC)

This command allows you to retrieve up to three macro definitions previously stored in a file. The macro definitions must contain only the "definition" portion of the MACRO command. The macro definitions are stored in successively numbered macros: the first definition becomes macro 1, etc.

The file used to store the macro definitions must be the latest version of file MCALL -- that is, MCALL;n. The filetype must be null or blank. If the macro definitions to be loaded are in a file of another name, you can use the PIP COPY command, with the /NV switch, to rename the file. Refer to Chapter 4 for descriptions of PIP commands.

Format

MCall

Strings of concatenated EDI commands can be written as EDI macro definitions, and up to three EDI macro definitions can be stored in file MCALL;n. The MC command is used to call the latest version of file MCALL and move the three definitions into the macro storage area. Then you can execute the desired macro without having to type the complete command.

Macro calls may not be nested.

The concatenation character may precede, but not follow, a macro call.

Example

```
*MC
```

This command retrieves the macro definitions stored in file MCALL;n, where n represents the latest version of the file MCALL.

3.4.30 MACRO EXECUTE

This command executes macro x n times, while passing it the optional numeric argument a. If a macro numeric argument is defined via the percent sign (%) in the macro definition, the numeric argument contained in this command is passed for each execution of the macro. Before a macro can be executed, it must either have been defined via a MACRO command, or called via a MACRO CALL command.

Using this command, any one of the three macro definitions stored in the EDI macro storage area can be executed any number of times.

Format

```
[n]Mx [a]
```

n

Specifies the number of times the macro is to be executed.

x

Specifies the macro number (1, 2, or 3).

a

Specifies the numeric argument to be passed when the macro is executed (ignored if % argument is not present in macro definition).

Examples

```
*2M1
```

Execute macro number 1 twice.

```
*3M2 5
```

Execute macro number 2 three times, passing the numeric argument 5 each time the macro is executed.

The example in Section B.3.4 illustrates how to use the EDI macro commands in editing a file.

3.4.31 MACRO IMMEDIATE

This command defines and executes a macro in one step. The definition is enclosed within angle brackets and is identical to that of the MACRO command. The definition is copied into the macro 1 storage area

and immediately executed n times. The macro may also be subsequently executed by entering an M1 command. The command is equivalent to the two macro commands:

```
MACRO 1 definition
nM1
```

Format

```
n<definition>
```

Example

```
*<L ABC&C /ABC/DEF>
```

This command causes EDI to search the current block buffer for the string ABC and, when it locates ABC, to change the string to DEF. This macro is stored as macro number 1.

The example in Section B.3.3 illustrates the use of the MACRO IMMEDIATE command.

3.4.32 NEXT

NEXT moves the current line pointer backward and forward in the file. A positive number moves the current line pointer forward; a negative current line number moves it backward.

Format

```
Next [n]
   or
Next [-n]
```

For examples and information on how to use NEXT, refer to Section 3.2.12.

3.4.33 NEXT & PRINT

NEXT & PRINT performs the same function NEXT performs, except that the new current line is displayed.

Format

```
NP [n]
   or
NP [-n]
```

For examples and information on how to use NEXT & PRINT, refer to Section 3.2.13.

3.4.34 OPEN SECONDARY

This command opens the specified secondary input file. The primary input file, if any, remains open. Subsequent text is read from the primary input file until the secondary input file is selected via the SELECT SECONDARY command (SS) for input.

Format

OPens filespec

Example

```
*OPENS RICKS.MAC
*SS
*READ 1
```

The file RICKS.MAC is opened as a secondary input file and, selected for input; then the first block is read in.

3.4.35 OUTPUT ON/OFF

This command, used only in the line-by-line mode, allows you to continue or discontinue the transfer of text to the output file. OUTPUT ON is the default condition; it is automatically reestablished each time a CLOSE command is issued.

Format

```
OUtput ON
    or
OUtput OFF
```

If neither ON or OFF is specified, ON is assumed.

Example

```
*BLOCK OFF
*OUTPUT OFF
*N 5
*OUTPUT ON
```

This example shows how to bypass five lines of text in the input file so that these lines are now written into the output file.

The first command sets line-by-line mode.

The second command disables the transfer of text to the output file.

The third command bypasses five consecutive lines of text from the input file.

The fourth command reenables the transfer of text to the output file.

3.4.36 OVERLAY

This command deletes n lines and replaces them with any number of lines that you type. If n is not specified, the current line is deleted and replaced with the lines typed. When you enter the OVERLAY command, EDI enters input mode. All text that you type goes into the file until you enter a carriage return as the only character in an input line.

Format

Overlay [n]

Example

```
*O 2
```

This command deletes two lines and causes EDI to enter input mode.

3.4.37 PAGE

This command causes EDI to enter block mode, if not already in block mode, and read page n into the block buffer. A page is delimited by form feeds. If n is less than the current page number, a TOF command is performed first. TOF processing writes the input file to the output file, closes both files, then opens the latest version of the file.

If n is greater than the current page number, the necessary number of RENEW commands is executed to read page n into the block buffer.

Format

Example

```
*PAG 1
[00050 LINES READ IN]
[00050 LINES READ IN]
[00050 LINES READ IN]
[00050 LINES READ IN]
[00017 LINES READ IN]
[PAGE 1]
*
```

This example shows a quick way to get to the last block in a file that contains no form feed page delimiters. EDI's page count is not incremented unless it encounters form feed characters or an end-of-file mark. Thus, in a file without form feeds (i.e., most files), EDI renews the block buffer until it encounters an end-of-file mark. Note that the final block contains 17 lines of text.

3.4.38 PAGE FIND

This command performs the same function as the FIND command, except that successive blocks are searched until the nth occurrence of the string has been found. The contents of the block buffer and the blocks between the current block and the block in which the nth occurrence of the string is located are copied into the output file.

The string must begin in column 1 of the matched line. The line is printed if VERIFY ON is in effect. This command may be used only in block mode.

Format

```
[n]PFind string
```

3.4.39 PAGE LOCATE

This command causes a search of the current block, starting at the line following the current line, and of successive blocks until the nth occurrence of the string has been located. Text from the current block buffer is written into the output file. The string can occur any place in the lines checked. The line is printed if the VERIFY ON command is in effect. This command may be used only in block mode.

Format

[n]PLocate string

This command is used in the same manner as the LOCATE command, except that the specified string can be in a block other than the current block.

PL leaves the current line pointer at end-of-file if it cannot locate string.

3.4.40 PASTE

This command is identical to the LINE CHANGE command, except that all lines remaining in the input file or block buffer are searched, and all occurrences of string1 are replaced with string2. Modified lines are printed if the VERIFY ON command is in effect. If string1 is given, but no match is found, then EDI returns an * prompt. When the command completes, the line pointer is at the top of the buffer or input file.

Format

PAste /string1/string2[/]

Example

If the lines remaining in the block buffer contain the following text:

```
YIGER, YIGER, BURNING BRIGHY
IN YHE FORESYS OF YHE NIGHY
```

they can be corrected with the following command:

```
*PA/Y/T
```

If the VERIFY ON command is in effect, all corrected lines are printed. To discontinue printing, type CTRL/O.

3.4.41 PRINT

PRINT displays the current line and the next n-1 lines at the terminal; the last line printed becomes the current line.

Format

Print [n]

For examples and information on how to use PRINT, refer to Section 3.2.14.

3.4.42 READ

This command reads the next n blocks of text into the block buffer. If a block is already in the buffer, the new block(s) is (are) appended to it.

EDI must be in block mode before this command can be executed.

A READ command may not exceed the buffer capacity. If you issue a READ that is too large, EDI fills its buffer and then issues the following message:

```
[BUFFER CAPACITY EXCEEDED BY]
<offending line>
[LINE DELETED]
```

You may get this message after issuing a READ n command, where n is 2 or larger, unless you have used the SIZE command to reduce the number of lines per block below its initial number.

Format

```
REAd [n]
```

If n is not specified, a value of 1 is assumed. The value of n must be positive.

Example

```
*SIZE 15
*READ 4
```

This example reads four 15-line blocks of the input file into the block buffer.

3.4.43 RENEW

RENEW writes the current block buffer into the output file and reads a new block from the input file. Renew is used only in block mode.

Format

```
RENew [n]
```

For examples and information on how to use RENEW, refer to Section 3.2.15.

3.4.44 <RETURN>

In edit mode, <RETURN> represents the carriage return that displays the next line in the file or block buffer. In input mode, entering the carriage return returns EDI to edit mode. For information on EDI command modes, refer to Section 3.1.2. For information on <RETURN>, refer to Section 3.2.16.

3.4.45 RETYPE

RETYPE replaces the current line with string.

Format

```
Retype [string]
```

For information on how to use RETYPE, refer to Section 3.2.17.

3.4.46 SAVE

This command causes the current line, and the next n-1 lines, to be saved in the specified file. If the file already exists, a new version is created.

If no file is specified, the save file generated has the name SAVE.TMP. SAVE puts the temporary file in the UFD and on the device of the file you are editing.

The input file or buffer information that is transferred to the SAVE file remains intact. The new current line is the last line saved. The SAVE command does not delete lines in the block buffer or input file.

Format

```
SAve[n] [filespec]
```

Example

You can save and later insert small groups of lines in several places in an output file by using the SAVE and UNSAVE commands. For example, a file called EDIT.MAC contains six lines that you want to insert at several points in another file called HELP.MAC. The procedure is:

1. Start an editing session using EDIT.MAC as the input file.
2. Locate the lines to be inserted into HELP.MAC.
3. Issue a SAVE 6 command. (This copies the six lines to be saved into the file SAVE.TMP.)
4. Issue a KILL command to terminate the editing session.
5. Start a new editing session using HELP.MAC as the input file.
6. Locate each place the six lines are to be inserted and issue the UNSAVE command.
7. Make further edits to the input file, as desired, or EXIT.

EDI does not delete the save file; the save file remains on the specified volume until deleted.

3.4.47 SEARCH & CHANGE

This command causes a search for string1 in the block buffer (block mode) or input file (line-by-line mode), beginning at the current line. The string may occur anywhere in the line. When string1 is located, it is replaced by string2. The located line becomes the current line.

If string1 is not specified, EDI prints the error message for illegal string construction. The new current line is printed if the VERIFY ON command is in effect. If string1 is given but EDI cannot locate the string, EDI returns an * prompt, and the line pointer is positioned at the end-of-file or the bottom of the block buffer.

Format

SC /string1/string2[/]

Example

If the following incorrect line is contained in the current block:

```
THES IS THE LINE TO BE ISSUED.
```

the following command can correct the error:

```
*V ON
*SC /THES/THIS/
THIS IS THE LINE TO BE ISSUED.
```

The corrected line is printed since the VERIFY ON command is in effect.

3.4.48 SELECT PRIMARY

This command selects the primary file for input. It allows you to reestablish the primary input file as the file from which text is read.

Format

SP

Example

```
*OPENS SECOND.MAC
*SS
*RENEW 10
*CLOSES
*SP
```

This example directs EDI to:

1. Open the secondary file SECOND.MAC.
2. Select SECOND.MAC as the secondary input file.
3. Read ten consecutive block buffers from the secondary input file into the block buffer. The first nine blocks are automatically transferred to the output file.

4. Close the secondary input file SECOND.MAC. The secondary file need not be closed before the primary file is reselected for input.

5. Reselect the primary input file for input.

3.4.49 SELECT SECONDARY

With this command you select the secondary file as the input file.

Format

SS

Example

To add text to the output file from a secondary input file, you must first open the secondary input file and select it for input. The use of the SS command is illustrated in the example presented in the previous section.

3.4.50 SIZE

This command allows you to specify the maximum number of lines to be read into the block buffer on a single READ or RENEW command. The default value for SIZE is dependent on your exact system configuration. Initially EDI determines how much buffer space it has and divides that by 132(10), the maximum line size, to set the number of lines read in. In no case can it be less than 38 lines. (See discussion of Block Mode in Section 3.1.3.)

Format

SIZE n

Example

```
*SIZE 50
```

This command conditions EDI to read 50 lines into the block buffer during a single READ or RENEW command.

3.4.51 TAB ON/OFF

This command turns automatic tabbing on or off. The automatic tab feature is useful for MACRO-11 language input. TAB OFF is the default at the start of an editing session. When TAB ON is in effect, a tab (equivalent to eight spaces) is automatically inserted at the beginning of each input line unless the line either begins with a label followed by a colon, or contains a semicolon in the first column.

Format

```
TAb [ON]
   or
TAb OFF
```

If neither ON nor OFF is specified when a TAB command is issued, ON is assumed.

Example

```
*TAB ON
*I
; THIS IS A SAMPLE OF TABBING.
THIS LINE GETS A TAB
1: THIS ONE DOESN'T
END

*TAB OFF
*N -3
*P 4
; THIS IS A SAMPLE OF TABBING.
        THIS LINE GETS A TAB
1:  THIS ONE DOESN'T
        END
```

3.4.52 TOP

TOP sets the current line pointer to the top of the block buffer (in block mode) or to the top of the file (in line-by-line mode). In line-by-line mode, TOP creates a new version of the file. When the current line pointer is positioned via TOP, you can insert lines preceding the first line in the file.

Format

```
Top
```

For examples and information on how to use TOP, refer to Section 3.2.18.

3.4.53 TOP OF FILE (TOF)

TOF returns the current line pointer to the first line of the file and leaves you in block mode. TOF copies the input file to the output file, closes both, and opens the latest version of the file as the input file.

Format

```
TOF
```

For examples and information on how to use TOF, refer to Section 3.2.19.

3.4.54 TYPE

This command is similar to the PRINT command (Section 3.2.14). In line-by-line mode, the two are identical. In block mode, TYPE does not move the line pointer after displaying the requested text, except if the end-of-block is encountered. In this case the line pointer remains at the last line before the end-of-block.

If n is not specified, a value of 1 is assumed.

Format

TYpe [n]

Example

See the example of the PRINT command (Section 3.2.14).

3.4.55 UNSAVE

This command retrieves all the lines in a specified file and copies them after the current line. If no file is specified, the file defaults to SAVE.TMP. The new current line pointer is positioned at the last line retrieved from the file. The file used in this command can be any text file; it is often the file created with a SAVE command.

Format

UNSave [filespec]

Example

File SEC.DAT;1 contains a group of lines to be inserted after the current line. The following command performs the desired operation.

```
*UNS SEC.DAT;1
```

Section B.3.2 contains an example using the SAVE and UNSAVE commands.

3.4.56 UPPER CASE ON/OFF

This command allows you to enter lower-case characters from a terminal and have them converted to upper-case characters. If UPPER CASE OFF is issued, all input characters are accepted as they are entered (no conversion is performed), except that all EDI commands are converted to upper-case characters. (If entered as lower-case characters, EDI commands are still echoed in lower case.)

Format

```
UC [ON]
  or
UC OFF
```

If neither ON nor OFF is specified, then ON is assumed. UC ON is the default when EDI is started.

Example

```
*UC OFF
*I this line is entered in lower case
*UC ON
*I this line is converted to upper case
```

Assuming that the input terminal is capable of generating lower-case input, the command in the example above would create the following lines in the output file.

```
this line is entered in lower case
THIS LINE IS CONVERTED TO UPPER CASE
```

However, in both instances, the characters are converted to upper case before the file is closed.

To create a file containing lower-case characters, use the MCR SET /LOWER-=TI: command and the EDI UC OFF command.

3.4.57 VERIFY ON/OFF

This command controls the display of lines specified by the LOCATE and CHANGE commands. Use VERIFY ON to display a line located by the LOCATE command or to display a line changed by the CHANGE command. Use VERIFY OFF to inhibit the display of these lines. EDI is in the VERIFY ON mode at the start of editing.

Format

```
Verify [ON]
    or
Verify OFF
```

If neither ON nor OFF is specified, ON is assumed

Example

```
*V OFF
*L VERIFY
*P
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
*N -2
*V ON
*L VERIFY
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
```

In this example, the PRINT command is issued to demonstrate that the desired line has been located when VERIFY is OFF, but when the LOCATE command is reissued with VERIFY ON, EDI automatically prints the line.

3.4.58 WRITE

This command causes the entire contents of the block buffer to be written into the output file. The block buffer is then erased.

EDI must be in block mode before this command can be executed.

Format

Write

Example

```
*W
*REA 2
```

In this example, the block buffer is written into the output file and the block buffer is erased. Then, the next two blocks are read into the block buffer.

3.5 EDI USAGE NOTES

The following points contain general information involving restrictions on use of EDI, system device considerations, and general usage rules.

- EDI can operate only on Files-11 format files and rejects all other file formats.
- The output file generated by EDI always resides on the same device as the input file. The output file cannot be directed to another device. For example:

  To edit a file on DECtape and store the resulting file on disk, do one of the following:

  1. Transfer the file to disk and perform the editing there.
  2. Edit the file on DECtape and then use PIP to transfer the file to disk.

- To use a device other than SY0:, mount it via the MCR MOUNT command.
- To edit a version of a file other than the latest one, explicitly state the desired version number in the file specification. This file is opened as the input file; the version number of the output file is one greater than the latest version of the file.
- Some EDI commands (such as TOF, and TOP when it is used in line-by-line mode) implicitly generate multiple versions of a file. In the execution of such commands, EDI copies the remainder of the input file into the output file and closes both of them. It then opens the latest version of the file and uses it as input. This ensures the editing of the latest version of the file and provides periodic backup. To delete any unwanted versions, use PIP with the /PURGE switch.
- EDI accepts variable-length input lines up to 132 characters long.
- The record type of output files edited by EDI is always variable-length.
- EDI preserves the record attributes of the input file. For example, the FORTRAN carriage control attribute is preserved in the output file.

- Line feed characters may be entered in files but are interpreted by EDI as termination characters. You should avoid using them since they cause unpredictable results when the file is edited a second time.

- EDI cannot process a file that contains embedded carriage control characters, such as PIP directory listings and TKB map files. To reformat such a file for EDI processing, copy the file to a DOS-11 volume and then back to your original volume using FLX. EDI can then process the file.

3.6 EDI ERROR MESSAGES

The four classes of EDI error messages are:

- Command level information messages
- File access error messages
- Error messages requiring EDI restart
- Fatal error messages.

The following sections describe all the messages that may be displayed in each class. If the recovery procedure is not evident, a suggested user action is given.

3.6.1 Command Level Information Messages

Messages in this class indicate information that is designed to be helpful to you or to identify errors that were encountered in the previous command. All messages in this class are enclosed within square brackets and followed by a prompt for a new command. For example, the following output occurs if a DELETE command encounters an end-of-buffer in block mode:

```
[*EOB*]
*
```

The messages in this class follow.

```
[BUFFER CAPACITY EXCEEDED BY]
offending line
[LINE DELETED]
```

Explanation: A READ, UNSAVE, INSERT, or OVERLAY command has exceeded the capacity of the block buffer. The line that caused the overflow is displayed and deleted.

User Action: If a new file is being created, empty the buffer with a WRITE command and continue the editing session.

If an existing file is being edited, it may be possible to continue via a RENEW or WRITE command. Otherwise, use the CLOSE command to close the output file and save all edits. Reopen the output file as the input file and, using the SIZE command, reduce the number of lines read into each buffer; then, using the PAGE LOCATE command, search to the position in the file where editing is to continue.

Occasionally, this message results when you try to open a file that was not created by EDI. You can overcome this difficulty via the SIZE command procedures that follow:

1. Type KILL.
2. When EDI prompts for a new file specification, enter a nonexistent filename. EDI creates a new file and enters input mode.
3. Type carriage return to enter edit mode.
4. Using the SIZE command, reduce the number of lines read into each buffer.
5. Use the KILL command to abandon the file.
6. When EDI prompts for a new file specification, enter the name of the desired file.

[CREATING NEW FILE]
INPUT

Explanation: The input file specified in the command does not exist and EDI has created a new file. EDI automatically enters input mode and waits for the input of text lines.

User Action: If you intend to create a new file, continue entering new lines as required. Otherwise, enter edit mode by typing carriage return; use the KILL command to delete the undesired new file. When EDI prompts for a new file specification, enter the correct file specification.

[ILL CMD]

Explanation: Either EDI does not recognize the command, or you have entered a command that is not compatible with the current mode (e.g., a READ command in line-by-line mode).

[ILL NUM]

Explanation: Either you have supplied a non-numeric character when a numeric is required, or you have given a negative number when a positive number is required.

[ILL STRING CONST]

Explanation: A search string specified in a CHANGE, LC, PASTE, or SC command does not contain a matching string termination character (e.g., PASTE /ALPHABETA, instead of PASTE /ALPHA/BETA).

[ILLEGAL IN BLOCK ON MODE]

Explanation: You have tried to execute a command that is illegal in block mode, such as FILE or OUTPUT ON/OFF.

[ILLEGAL FILE NAME GIVEN IN CLOSE OR EXIT]

or

[FILE WAS NOT RENAMED]

Explanation: A syntactically incorrect file specification was given in a CLOSE or EXIT command, the attempt to rename the output file failed, or the attempt to EXit or Close to rename the file to another device failed.

User Action: Use PIP to rename the file, if desired.

[MACRO NOT DEFINED]

Explanation: You have tried to execute a macro with the M command, but the specified macro has not been defined.

User Action: Use the MACRO command to define the desired macro and then execute it with the M command.

[MACRO NUMERIC ARG UNDEFINED]

Explanation: You have tried to execute a macro without supplying a numeric argument. The macro definition, however, contains the % character and thus demands a numeric argument.

User Action: Retype the command, specifying the appropriate numeric argument.

[MCALL FILE DOES NOT EXIST]

Explanation: An MCALL command has been executed to define a set of macros, but the file MCALL cannot be found.

User Action: The desired set of macro definitions may exist under another User File Directory. If this is the case, use PIP to copy or rename the MCALL file into the current directory.

[NO INPUT FILE OPEN]

Explanation: A PAGE, READ, or RENEW command has been attempted while a new file is being created (that is, while there is no input file). These commands can be executed only when an existing file is being edited.

[NO MATCH]

Explanation: A CHANGE command has specified a string to be changed that is not in the current line.

[OVERLAYING PREVIOUSLY DEFINED MACRO]

Explanation: A MACRO command has resulted in the redefinition of a previously defined macro. This message lets you know that the previous definition is no longer in effect.

[SAVE FILE DOES NOT EXIST]

Explanation: The file specified in an UNSAVE command cannot be located.

User Action: If you provided a file specification, make sure it is correct. If you did not give a file specification, this message means that no previous SAVE command (without file specification) was issued.

[SECONDARY FILE ALREADY OPEN]

Explanation: You may have tried to open a secondary input file while another secondary file is still open. Or you may have a secondary file open when you issue a CLOSE or KILL command, or when EDI encounters an error and is forced to restart. The former case represents an error: the latter informs you that you still have a secondary file open.

User Action: Close the secondary input file using the CLOSES command, and then open the desired secondary file with the OPENS command.

[SECONDARY FILE CURRENTLY SELECTED FOR INPUT]

Explanation: A CLOSE or KILL command has been issued, or an error has caused EDI to restart, when a secondary input file is open and selected for input.

User Action: Issue an SP command, then a CLOSES command.

[SYNTAX ERROR]

Explanation: A command has been entered that is syntactically incorrect.

[TOO MANY CHARS]

Explanation: A CHANGE, LC, PASTE, SC, or ADD command has resulted in a line that contains too many characters. EDI limits the length of a line to 132 characters.

User Action: Retype the line to ensure that the line is valid.

[*BOB*]

Explanation: The beginning-of-buffer has been reached. The current line pointer is positioned just before the first line in the buffer. Thus, new text lines can be entered before the first line.

[*EOB*]

Explanation: The end-of-buffer has been reached. The current line pointer now points to the end of the buffer. Thus, if new lines are inserted, they appear after the last text in the buffer.

[*EOF*]

Explanation: The end-of-file has been reached on the input file.

User Action: If the editing session is complete, use the CLOSE or EXIT command to close the output file. Otherwise, use the TOF command to return to the first block in the file.

3.6.2 File Access Error Messages

Messages in this class mean that you have tried to access directories, files, or devices that are not present in the host system. Each message is prefixed with:

EDI --

After the message is displayed, EDI returns to command level and prints an asterisk to request input.

The messages in this class follow.

EDI -- BAD FILE NAME

Explanation: The file name was not accepted by EDI. The most common error is a file name containing embedded blanks.

User Action: Make sure that the file name is correct, and reenter it.

EDI -- DEVICE NOT IN SYSTEM

Explanation: You have given a FILE, OPENS, SAVE, or UNSAVE command, specifying a device that does not exist in the host system.

User Action: Reenter the command line, specifying only devices available in the system.

EDI -- FILE DOES NOT EXIST

Explanation: You have given a FILE or SAVE command, specifying a user file directory that does not exist on the specified volume.

NOTE

The remaining error messages in this class should not occur and represent failures in EDI. If such errors persist, submit a Software Performance Report.

EDI -- BAD DEVICE NAME

EDI -- DEVICE NOT READY

EDI -- FILE ALREADY OPEN

EDI -- RENAME NAME ALREADY IN USE

EDI -- RENAME ON TWO DIFFERENT DEVICES

EDI -- WRITE ATTEMPT TO LOCKED UNIT

3.6.3 Error Messages Requiring EDI Restart

The error messages described in this section are caused by conditions that make it impossible for EDI to continue the current editing session. EDI closes all open files (with the exception of any open secondary input file), reinitializes, and then prompts for the next file to be edited.

As with file access warning messages, each message in this class is prefixed with:

```
EDI --
```

After the appropriate message has been displayed, EDI prompts with:

```
EDI>
```

You may terminate the editing session at this point by typing carriage return or CTRL/Z, or you may continue by entering another file specification. If a secondary file was open when the error condition was encountered, it remains open.

The messages in this class follow.

EDI -- BAD RECORD TYPE - FILE NO LONGER USABLE

Explanation: The record type defined in the header block of the input file (primary input, secondary input, UNSAVE, or MCALL) is not supported by File Control Services (FCS); thus, the file cannot be used for input to EDI.

User Action: The referenced file has been created without using FCS, or the file structure on the volume is damaged. In the latter case, verify the file structure with the verification utility (VFY) to determine the extent of the damage. VFY is described in Chapter 13.

EDI -- FILE IS ACCESSED FOR WRITE

Explanation: The input file (primary input, secondary input, UNSAVE, or MCALL) is currently being written by another task.

User Action: Wait for the write to complete, then reenter the command line.

EDI -- FILE IS LOCKED TO WRITE ACCESS

Explanation: The output file (text output, FILE, or SAVE) is currently accessed for read by one or more tasks and is locked against all writers.

User Action: Wait for all readers of the file to finish, then reenter the command line.

EDI -- ILLEGAL RECORD ATTRIBUTES - FILE NOT USABLE

Explanation: The record attributes defined in the header block of the input file (primary input, secondary input, UNSAVE, or MCALL) are not supported by FCS; thus, the file cannot be used for input to EDI.

User Action: The referenced file has been created without using FCS, or the file structure on the volume is damaged. In the latter case, run the file structure verification utility (VFY) to determine the extent of the damage. VFY is described in Chapter 13.

EDI -- PRIMARY FILE NOT PROPERLY CLOSED

Explanation: When the primary input file was last written, a close check was specified, and the writing task did not properly close the file (e.g., the task was aborted). Thus, the file attributes were not written, and the file may contain inconsistent data.

User Action: Exit from EDI by typing carriage return or CTRL/Z. Run the Peripheral Interchange Program (PIP) and use the /UN switch to unlock the file. Reinitiate EDI and try to recover the data in the file.

EDI -- PRIVILEGE VIOLATION

Explanation: A privilege violation occurred during a file access for one of the following reasons:

1. The specified volume is not mounted.
2. The UIC under which EDI is running does not possess the necessary privileges to access the specified directory.
3. The UIC is not privileged to access the specified file.

User Action: If the volume is not mounted, then mount it using the MCR MOUNT command. Otherwise, reinitiate EDI under a UIC that has appropriate access privileges to both the specified directory and file.

EDI -- RECORD IS TOO LARGE FOR USER BUFFER

Explanation: The input file (primary input, secondary input, UNSAVE, or MCALL) being accessed was not created by EDI (or SLP) and contains records that are too large. The maximum record length supported by EDI is 132 bytes.

EDI -- SECONDARY FILE NOT PROPERLY CLOSED - NOT USABLE

Explanation: When the secondary input file was last written, a close check was specified, and the writing task did not properly close the file (e.g., the task was aborted). Thus, the file attributes were not written, and the file may contain inconsistent data.

User Action: Run PIP and use the /UN switch to unlock the file. Reinitiate EDI and try to recover the data in the file.

EDI -- BAD DIRECTORY SYNTAX

Explanation: Directory field ([g,m]) is in improper format.

NOTE

The remaining error messages in this class should not occur and represent failures in EDI. If such errors persist, submit a Software Performance Report.

EDI -- DUPLICATE ENTRY IN DIRECTORY

EDI -- END OF FILE

EDI -- ILLEGAL RECORD ACCESS BITS - FILE NOT USABLE

EDI -- ILLEGAL RECORD NUMBER - FILE NOT USABLE

3.6.4 Fatal Error Messages

The fatal error messages represent system and/or hardware conditions that make it impossible for EDI to continue execution. All files are closed and EDI terminates its execution. The output file may be truncated. Each error message is prefixed with:

```
EDI --
```

and followed by the exit message:

```
[EXIT]
```

on the next line.

The advanced user may work with the following procedures on the truncated version of an output file to save the editing performed before the fatal error occurred.

1. Use PIP to rename the truncated version of the output file to avoid confusion.

2. Restart the editing session to the original input file.

3. Issue an OPENS command, specifying the renamed file as the secondary file.

4. Issue an SS command to select the secondary file for input.

5. Issue an ERASE command to erase the first block of the input file (unless the truncated output file did not contain the entire first block).

6. Issue as many READ 1 and WRITE commands as necessary to reach the EOF on the secondary file.

7. Issue an SP command to select the primary file for input.

8. Issue a CLOSES command to close the secondary file.

9. Issue a WRITE command to ensure that the last block was written into the output file.

10. Issue as many READ 1 and ERASE commands as necessary to bypass all input file blocks that are complete in the renamed file.

11. Continue the normal editing session.

The messages in this class follow.

EDI -- CALLER'S NODES EXHAUSTED

Explanation: System dynamic storage has been depleted, and insufficient space is available to allocate the control blocks necessary to open, close, read, or write a file.

User Action: This probably is a system failure, but it could also represent a transient overload condition. Wait until system load has diminished and reinitiate EDI.

EDI -- DEVICE FULL

Explanation: Insufficient space exists on the output volume to extend an output file (text output FILE, or SAVE).

User Action: Determine which volume is being written. If it is required that the specified file be written on this volume, then space must be made available. Use PIP to purge (/PU) or delete (/DE) unwanted files.

EDI -- FILE HEADER CHECKSUM ERROR

Explanation: An input file (primary input, secondary input, UNSAVE, or MCALL) has a header block that does not contain a proper checksum.

User Action: The file structure on the specified volume is damaged. Run the file structure verification utility (VFY) to determine the extent of the damage. VFY is described in Chapter 13.

EDI -- FILE HEADER FULL

Explanation: Insufficient retrieval pointer space exists in the header block to extend an output file (text output, FILE, or SAVE).

User Action: Split the file into two or more files and process them separately.

EDI -- FILE PROCESSOR DEVICE WRITE ERROR

Explanation: This error message may indicate that the device specified for an output file is write-locked.

User Action: Unlock the device if it is write-locked. Otherwise, a hardware problem may exist. Consult the DIGITAL field service representative.

EDI -- INDEX FILE FULL

Explanation: The file header block is not available to create an output file (text output, FILE, or SAVE). When a volume is initialized, the maximum number of files that may be created on the volume is established. Your write request would have exceeded this maximum.

User Action: Determine which volume is being referenced. If it is required that the specified file be created on this volume, then space must be made available. Use PIP to purge (/PU) or delete (/DE) unwanted files.

NOTE

The following error messages signify hardware problems. If possible, remove all important files from the volume.

EDI -- BAD BLOCK ON DEVICE

EDI -- FILE PROCESSOR DEVICE READ ERROR

EDI -- HARDWARE ERROR ON DEVICE

EDI -- PARITY ERROR ON DEVICE

NOTE

The remaining error messages in this class should not occur and represent failures in EDI.

EDI -- BAD DIRECTORY FILE

EDI -- BAD PARAMETERS ON A QIO

```
EDI -- INVALID FUNCTION CODE ON A QIO

EDI -- NO BLOCKS LEFT

EDI -- REQUEST TERMINATED

EDI -- UNEXPECTED ERROR - EDITOR WILL ABORT

EDI -- WRITE ATTRIBUTE DATA FORMAT ERROR

TASK "...EDI" TERMINATED
```

PART 3
FILE MANIPULATION UTILITIES

CHAPTER 4

PERIPHERAL INTERCHANGE PROGRAM (PIP)

The Peripheral Interchange Program (PIP) is an RSX-11 file utility program that transfers data files from one standard Files-11 device to another. PIP also performs file control functions. Some of the functions PIP performs are:

- Copy files from one device to another
- Delete files
- Rename files
- List file directories
- Set the default device and UIC for PIP operations
- Unlock files
- Spool files

You invoke the PIP utility using any of the methods for invoking a utility described in Chapter 1. You invoke PIP file control functions by means of switches and subswitches.

4.1 PIP COMMAND STRING

You request PIP functions by entering PIP command strings through the initiating terminal or by means of an indirect command file. (Using indirect command files is described in Chapter 1.) The format of PIP command strings differs for each function. Therefore, the command string formats are described in separate sections.

4.1.1 PIP Defaults for File Specification Elements

With the exception of the version number, PIP generally uses the last value encountered in the command line as the default. That is, PIP uses values you enter to set defaults, and changes the default when you change the value. Exceptions to this are noted in the descriptions of each switch.

The following command string sets a new default value for each file specified on the command file:

```
>PIP TI.MAC;5,T2,.TSK,;6/LI

T1.MAC;5
T2.MAC;5
T2.TSK;5
T2.TSK;6
```

Table 4-1 summarizes the rules PIP uses to set defaults.

Table 4-1
PIP Default File Specifications

Element	Default Value
dev:	For the first file specification, the unit on which the system disk is mounted (SY:) or the default that you specify with /DF (see Section 4.2.2.5). For subsequent file specifications, either you explicitly specify a new device or PIP assumes the device from the previous specification.
[ufd]	For the first file specification, your current UIC, that is, the UIC under which you log on, the UIC you specify with the MCR SET command, or the default you specify with /DF (see Section 4.2.2.5). For subsequent file specifications, either you explicitly specify a new ufd or PIP assumes the ufd from the previous specification. An asterisk (wildcard) specification is valid (see Section 4.1.3).
filename	No default for the first file specification. For subsequent file specifications, the last filename that you explicitly specified. An asterisk (wildcard) specification is valid (see Section 4.1.3).
.filetype	No default for the first file specification. For subsequent file specifications, the last file type that you explicitly specified. An asterisk (wildcard) specification is valid (see Section 4.1.3).
;version	The default for input files is the most recent version number. The default for output files is the next higher version number, or version 1 if the file does not already exist in the output directory. An exception is the PIP file delete funciton, which requires an explicit version number or a wildcard specification. An explicit version number is defined to be of the form ;n where n is greater than 0 (n>0). A version number of ;-1 may be used to specify the oldest version of a file. A version number of ;0 or ; may be specified to signify the most recent version. In certain cases, the asterisk (or wildcard) may be specified, as described in Section 4.1.3.

4.1.2 PIP Switches and Subswitches

PIP provides several file control switches and subswitches. A switch specification consists of a slash (/) followed by a 2- or 3-character switch name. The switch specification is optionally followed by a subswitch name separated from the switch name by a slash. The subswitch name can have arguments that are separated from the subswitch name by a colon (:). Each is preceded by a slash.

All but one of the PIP switches operate on lists of file specifications. The exception is the ID switch, which is used by itself.

Section 4.1.2.1 describes switches and Section 4.1.2.2 describes subswitches.

4.1.2.1 **Switches** - PIP accepts some switches with no file specification. However, when you use a switch in a command line, it must follow the file or UFD specification. It cannot come before the file name, file type, version, or UFD of the file on which it is to operate.

You may specify switches once for an entire list of file specifications. For example:

```
string1,string2,string3/DE
```

The DE switch applies to all of the strings. PIP deletes every specified file from its User File Directory.

You specify switch arguments as octal (default), decimal, or alphabetic characters, depending on the switch. The sections that explain the individual PIP switches discuss these values.

4.1.2.2 **Subswitches** - Subswitches are local. They apply only to the file specification which immediately precedes them. In the following example, the New Version subswitch (/NV) is applied to the file ASDG.MAC. (The NV subswitch is used with the Rename switch, RE.)

```
PIP>*.SMP=PRT2.QRT,ASDG.MAC/NV,KG.BAC/RE
```

PIP renames the files PRT2.QRT and KG.BAC, but they maintain their associated version numbers. File ASDG.MAC is also renamed, but the version number is forced to a number one greater than the latest version of file ASDG.SMP.

When you explicitly apply a subswitch to a file specification, you implicitly apply the switch with which the subswitch is associated. On a command line with more than one file specification, the explicit subswitch affects only the file to which it is applied. The implicit switch affects all the files on the command line.

Example

```
PIP>FILE1/GR:R/WO,FILE2/GR:RW
```

This command is equivalent to:

```
PIP>FILE1/GR:R/WO,FILE2/GR:RW/PR
```

The command results in the following file protection:

a. FILE1
SYSTEM	--	Unchanged
MEMBER	--	Unchanged
GROUP	--	Read access
WORLD	--	No access

b. FILE2
SYSTEM	--	Unchanged
MEMBER	--	Unchanged
GROUP	--	Read/write access
WORLD	--	Unchanged

(For more information on altering the protection level of a file, see Section 4.2.2.14).

4.1.3 **Wildcards**

PIP allows you to specify wildcards by means of an asterisk (*) character in the file specification. The asterisk (*) character in one or more fields of a file specification stands for "all"; for example, all files, file types, or versions. However, using wildcards is restricted in some cases. The following sections describe the uses of wildcards for input and output files.

4.1.3.1 **Wildcards in Output File Specifications** - Using wildcards in the output file specifications is restricted. For the following PIP functions, the output file specification cannot have any wildcards:

- Copying a single file
- Concatenating files to a specified file
- Appending to an existing file
- Updating (rewriting) an existing file
- Listing a directory

If you use wildcards in the output file specification for any of the above functions, the meaning of the command line would be ambiguous or imply an infinite number of output files. For example:

```
PIP>[200,200]*.*;*=TEST.DAT
```

PIP would try to create an infinite number of files in [200,200] from one single file.

When you make copies of several files, the output specification must be *.*;* or default.

For the Rename (/RE) and Enter (/EN) switches, the output specification may have wildcards mixed with specified fields. For either switch, the equivalent field of the input file specification is used.

For all cases in which wildcards are allowed in the output file specification, the wildcard UFD form [*,*] (but not [n,*] or [*,n]) is used to indicate that the output UFD is to be the same as the input UFD.

4.1.3.2 **Wildcards in Input Specifications** - PIP provides the following wildcard features for input file specifications:

- *.*;* means all versions of all files.
- *.DAT;* means all versions of all files of file type DAT.
- TEST.*;* means all versions and all types of files named TEST.
- TEST.DAT;* means all versions of file TEST.DAT.
- *.* means the most recent version of all files.
- *.DAT means the most recent version of all files of file type .DAT.
- TEST.* means the most recent version of all file types for files named TEST.

PIP also provides the following wildcard UFD features:

- [*,*] means all group, member number combinations from 1 to 377 octal.
- [n1,*] means all member numbers under group n1.
- [*,n2] means all group numbers for member n2.

4.2 PIP COMMAND FUNCTIONS

PIP copies Files-11 files and performs file control functions. Section 4.2.1 describes the copying function and Section 4.2.2 describes the file control functions.

4.2.1 Copying Files-11 Files

To copy Files-11 files, you can enter the PIP command line without specifying any switches.

The simplest format for the PIP command line is:

```
outfile=infile
```

outfile

The output file specification. If the output file name, file type, and version are either null or *.*;*, the input file name, file type, and version are preserved (see NV and SU subswitches). If you enter any portion of the output file specification (file name, file type, or version), wildcards cannot be used in this specification. Similarly, for a copy command, if you enter any portion of the output specification, you can enter only one file as the input file.

infile

The input file specification. If the file name, file type, and version fields are all null, then *.*;* is the default.

One switch that you can specify when copying Files-11 files is the Merge switch. The Merge switch (/ME) creates a new file from two or more existing files. PIP assumes /ME when you explicitly specify an output file, two or more input files, and no switches. Because the basic copy function and the Merge switch are logically related, the Merge switch is described here rather than below with the other switches.

The most general format of the PIP command line is:

```
outfile=infile1[,infile2,...infilen][[/ME][/subswitch]]
```

outfile

The output file specification.

infile

The input file specification.

/ME

Specifies the Merge switch.

/subswitch

Specifies any of the subswitches that you can enter as part of the basic command line or with the Merge switch. Table 4-2 describes these subswitches. Subswitches can appear in either the output or input file specification. If you place the subswitch in an input file specification, it applies only to that file. If you place the subswitch in the output file specification, it applies to the entire list of input specifications.

Examples

1. PIP>DK1:SAMP.DAT=DK2:TEST.DAT

   Copies the latest version of file TEST.DAT (in the current UFD) from DK2: to DK1: as SAMP.DAT.

2. PIP>DK1:[*,*]=DK0:[11,*]

   Copies all files from all members in group number 11 of DK0: to DK1:. The files are in the same UFD on DK1: that they were in on DK0:.

3. PIP>LP:=*.LST

   Copies the latest version of all files with a type of .LST in the current UFD to the line printer. If the Print Spooler is installed on your system, use the SP switch instead of this command. The command line using /SP is in the format:

   ```
   PIP>*.LST/SP
   ```

Note: Transparent spooling (that is, PIP>LP:=files) is implemented on RSX-11M-PLUS only. When you specify LP: as your output device, the data to be printed is written into an intermediate file and then the file is given to the Queue Manager, which handles the spooling. Making intermediate files allows you to dismount the volume the files are on without having to wait until after they have been printed.

4. PIP>DK1:SAMP.DAT=DK2:TEST.DAT;1,NEW.DAT;2/ME

   Concatenates version 1 of file TEST.DAT and version 2 of file NEW.DAT from DK2:, generating file SAMP.DAT on DK1:, using the current UFD.

5. PIP>DK1:=SY:TESTPROG.MAC,.OBJ

   Copies the latest versions of TESTPROG.MAC and TESTPROG.OBJ from the system device (SY:) to DK1:, using the current UFD for both SY: and DK1:.

6. PIP>DK1:=DK0:*.DAT;*

   Copies all versions of all of the files of file type .DAT in the current UFD from DK0: to DK1:.

7. PIP>DT0:=[200,10]*.*;*

   Copies all files under [200,10] from the default device to DT0:, using the current UFD.

8. PIP>DP0:[200,10]=DT0:*.*

   Copies all files from DT0: in the current UFD to DP0:[200,10]. Note that the user must have write access to [200,10].

Table 4-2
PIP Copy Command and Merge Subswitches

Subswitch	Description
/BL:n[.]	Blocks Allocated -- This subswitch specifies the number of contiguous blocks (n) to allocate initially to the output file. You can specify n as either an octal or decimal value (decimal values must be followed by a decimal point). You use the /BL:n subswitch when you are copying a contiguous file and changing its size.
/CO	Contiguous Output -- This subswitch specifies that the output file be contiguous. When you are copying contiguous files from magnetic tape (for example, task images), specify both /CO and /BL:n. You must specify /BL:n because PIP cannot determine the length of the input file when it allocates a file. (PIP allocates file space before the copy operation is executed. The length of magnetic tape input files is on the trailing label of such tapes.)

(continued on next page)

Table 4-2 (Cont.)
PIP Copy Command and Merge Subswitches

Subswitch	Description
/-CO	Noncontiguous output -- This subswitch specifies that the output file does not have to be contiguous. If you do not specify any of the above switches, PIP defaults to the size and attributes of the input file.
/FO	Set File Ownership -- This subswitch specifies that the owner of the input file also owns the output file. If you do not specify /FO, the owning UIC of all new files is the UIC under which PIP is running, regardless of which directory the files belong to. You can use this subswitch with both copy and merge commands. Examples 1. If PIP is running under the UIC [1,1], the command: DK0:[200,200]=DK1:[200,220]TEST.DAT creates a new file in the [200,200] directory on DK0:, but the file is owned by UIC [1,1]. However, the command: DK0:[200,200]=DK1:[200,220]TEST.DAT/FO creates a file owned by UIC [200,200]. When you specify /FO, PIP must be running under a UIC that has write-access to all output directories. 2. DK1:[*,*]/FO=DP0:[13,10],[32,10],[34,10] Copies all the files from the specified input directories to the corresponding directories on DK1:. The file owners are the output directories. 3. DK1:[*,*]=DK0:[*,10]*.MAC/FO Copies all the .MAC files from all group numbers with member 10 to DK1:, preserving the UFD and setting the file owner for each file to that UFD.
/SU	Supersede -- This subswitch allows you to copy one or more input files to a file whose file name, file type, and version already exist in a User File Directory. The existing file is deleted and a new one is created with the data from the input file(s).

(continued on next page)

Table 4-2 (Cont.)
PIP Copy Command and Merge Subswitches

Subswitch	Description
/SU (Cont.)	The output file's name, type, and version number remain the same, but its file identification number (File-ID) is different. Also, the attributes for the output file are taken from the first input file and the number of blocks allocated to the output file can be different (less than or more than) the number of blocks allocated to the existing file.
/NV	New Version -- This subswitch forces the output version number of the file being copied to become one greater than the latest version of the file already in the output directory. If the file does not already exist in the output directory, a version number of 1 is assigned. Figure 4-1 shows the results when you specify /NV. (Specifying /NV is not necessary when both the input and output files are under the same file directory.)

4.2.2 Performing File Control Functions

PIP provides several switches and subswitches for file control processing. These switches and subswitches perform such functions as deleting files, displaying the contents of a User File Directory, and specifying file protection values.

You can specify two PIP switches, /ID and /LI, on a command line with no file specifications (that is, they may be entered by themselves).

You can specify only one switch in a command line. However, more than one subswitch can be specified.

The values that you specify with the switches and subswitches default to octal. You can specify decimal values by adding a decimal point after the value.

Table 4-3 lists PIP switches and subswitches and summarizes the functions performed by them. The subswitches are listed with their respective switches. (The switches and subswitches are described in detail following the table.)

Directories Before COPY

INPUT DIRECTORY [2Ø1,2Ø1]	OUTPUT DIRECTORY [1ØØ,1ØØ]
RICK.DAT;1	RICK.DAT;2 RICK.DAT;4

Directories After COPY Without /NV Switch Set
(version number preserved)

INPUT DIRECTORY [2Ø1,2Ø1]	OUTPUT DIRECTORY [1ØØ,1ØØ]
RICK.DAT;1	RICK.DAT;2 RICK.DAT;4 RICK.DAT;1

The command used was:

```
DK1:[1ØØ,1ØØ] = DK2:[2Ø1,2Ø1]RICK.DAT;1
```

Directories After COPY With /NV Switch Set

INPUT DIRECTORY [2Ø1,2Ø1]	OUTPUT DIRECTORY [1ØØ,1ØØ]
RICK.DAT;1	RICK.DAT;2 RICK.DAT;4 RICK.DAT;5

The command used was:

```
DK1:[1ØØ,1ØØ] = DK1:[2Ø1,2Ø1]RICK.DAT;1/NV
```

NOTE

The version specified with the /NV subswitch must be explicit or default; no wild cards allowed.

Figure 4-1 Results of Copy Command With and Without /NV Specified

Table 4-3
PIP Switches and Subswitches

Switch	Subswitch	Function
/AP		Appends file(s) to the end of an existing file.
	/FO	Specifies the file ownership for a file.
/BR		Lists a directory file in brief format (an alternate mode for the LI switch).
/BS:n[.]		Defines the blocksize for magnetic tape.

(continued on next page)

Table 4-3 (Cont.)
PIP Switches and Subswitches

Switch	Subswitch	Function
/CD		Allows the output file to take the creation date of the input file rather than the date of transfer.
/DE		Deletes one or more files.
	/LD	Lists the deleted files.
/DF		Changes PIP's default device and/or UFD.
/EN		Enters a synonym for a file in a directory file.
	/NV	Forces the version number of a file to one greater than the latest version.
/EOF[:block:byte]		Specifies the end-of-file pointer for a file.
/FI:filenum:seqnum		Accesses a file by its file identification number (File-ID).
/FR		Displays the amount of free space on the specified volume and the largest contiguous free space on that volume.
/FU[:n[.]]		Lists a directory file in full format (an alternate mode for the LI switch).
/ID		Identifies the version of PIP being used.
/LI		Lists directory files.
/ME		Concatenates two or more files into one file.
	/BL:n[.]	Allocates a number (n) of contiguous blocks.
	/CO	Specifies that the output file(s) be contiguous.
	/FO	Specifies the file ownership for a file.
	/NV	Forces the version number of a file to one greater than the latest version.
	/SU	Supersedes (replaces) an existing file.
/NM		Suppresses certain PIP error messages.

(continued on next page)

Table 4-3 (Cont.)
PIP Switches and Subswitches

Switch	Subswitch	Function
/PR		Changes the protection status of a file.
	/FO	Specifies the ownership for a file.
	/GR[:RWED]	Sets the read/write/extend/delete protection at the group level.
	/OW[:RWED]	Sets the read/write/extend/delete protection at the owner level.
	/SY[:RWED]	Sets the read/write/extend/delete protection at the system level.
	/WO[:RWED]	Sets the read/write/extend/delete protection at the world level.
/PU[:n[.]]		Deletes obsolete version(s) of a file.
	/LD	Lists the deleted files.
/RE		Renames a file.
/RM		Removes a file entry from a directory.
/RW		Rewinds a magnetic tape.
/SB		Spans the block boundaries of records when copying from magnetic tape to disk.
/SD		Selectively deletes files by prompting for your response before deleting.
/SP[:n[.]]		Spools file(s) to the line printer for printing.
/SR		Allows shared reading of a file that has already been opened for writing by another user or task.
/TB		Lists the total number of blocks used for a directory, along with the total number of blocks allocated and the number of files in that directory (an alternate mode for the LI switch).
/TR		Truncates file(s) to logical end-of-file.
/UF		Creates a User File Directory entry on the volume to which a file is being transferred.
/UN		Unlocks a file.
/UP		Updates (rewrites) an existing file.
	/FO	Specifies the ownership for a file.

4.2.2.1 **/AP -- Append Switch** - The Append switch (/AP) opens an existing file and appends the input file(s) to the end of it. Specify the Append switch in the following format:

```
outfile=infile1[,infile2,...infilen]/AP[/FO]
```

outfile

The output file specification. Wildcard specifications are not allowed in the output file specification. The file type and the record attributes for the output file remain the same after the input file or files have been appended to it. The file name and file type for the output file must be specified explicitly.

infile

The input file specification. If the file name, file type, and version are null, then *.*;* is the default.

/AP

The Append switch.

/FO

The Set File Ownership subswitch, which specifies that the owning UIC of the output file is the same directory to which the input file belongs. If you do not specify /FO, the owning UIC of the output file is unchanged, regardless of which directory the input files belong to. See Section 4.2.1 for examples of using /FO.

Example

```
PIP>DK1:FILE1.DAT;1=FILE2.DAT;1,FILE3.DAT;1FILE4.DAT;1/AP
```

Opens FILE1.DAT;1 on DK1: and appends the contents of FILE2.DAT;1, FILE3.DAT;1 and FILE4.DAT;1 to it.

Note: If the output file is contiguous before the appending, it may not be contiguous afterwards.

4.2.2.2 **/BS:n -- Block Size Switch** - The Block Size switch defines the block size for magnetic tapes. This switch allows you to read or write bigger blocks onto magnetic tape, thereby saving some of the space taken by interrecord gaps. Specify /BS using the following format:

```
outfile/BS:n=infile
```

or

```
outfile=infile/BS:n
```

outfile

The output file specification.

infile

The input file specification.

/BS:n[.]

Specifies the Block Size switch where n is an octal or decimal number specifying the number of bytes in a block.

In the output specification, /BS specifies the block size of the output file. In the input specification, /BS specifies the block size of the input file. If the block size specified is smaller than the actual block size, an I/O error occurs.

4.2.2.3 **/CD -- Creation Date Switch** - The Creation Date switch (/CD), used in a file transfer command, allows the output file to take the date on which the input file was created rather than the date of transfer. You cannot use this switch with the Merge switch (/ME). Specify the CD switch in the following format:

outfile/CD=infile

outfile

The output file specification.

infile

The input file specification.

/CD

The Creation Date switch.

Example

```
DIRECTORY DB1:[200,200]
21-NOV-78 14:02

FILE.DAT;7          12.     6-OCT-78 16:13


PIP>TEST.DAT/CD=FILE.DAT


DIRECTORY DB1:[200,200]
21-NOV-78 14:05

FILE.DAT;7          12.     6-OCT-78 16:13
TEST.DAT;1          12.     6-OCT-78 16:13
```

The command creates a new file, TEST.DAT, from FILE.DAT and gives it the creation date of FILE.DAT rather than the transfer date.

4.2.2.4 **/DE -- Delete Switch** - The Delete switch (/DE) deletes files from a User File Directory. Optionally, you can specify that the deleted files be listed on your terminal. Specify /DE in the following format:

infile1[,infile2,...infilen]/DE[/LD]

infile

The input file specification.

/DE

The Delete switch.

/LD

The List Deleted files subswitch.

You must specify a version number or a wildcard in its place when using the Delete switch.

Use a version number of ;-1 to specify the oldest version of a file. Use a version number of ;0 or ; to specify the most recent version.

Examples

1. PIP>TEST.DAT;-1/DE

   Deletes the oldest version of file TEST.DAT.

2. PIP>TEST1.DAT;0,TEST2.DAT;/DE

   Deletes the latest version of files TEST1.DAT and TEST2.DAT.

Wildcards in the file name or file type fields are illegal when a version of ;-1, ;0, or ; is specified.

You must issue the file specification because a null file name, file type, and version do not default to *.*;*.

The input file specification can take all the usual forms, including wildcards (even in the group, member number [ufd]). The only special requirement is that the version field must always be explicit or *.

Examples

1. PIP>TEST.DAT;5/DE

   Deletes version 5 of file TEST.DAT in the current default directory on the default device.

2. PIP>TEST.DAT;1,;2/DE

   Deletes versions 1 and 2 of file TEST.DAT in the current default directory on the default device.

3. PIP>*.OBJ;*,*.TMP;*/DE/LD

   Deletes all versions of all files of the file type .OBJ and .TMP from the current default directory on the default device. Lists all deleted files of file type .TMP.

4. PIP>*.OBJ;*/LD,*.TMP;*/DE

   Deletes all versions of all files of the file type .OBJ and .TMP from the current default directory on the default device. Lists all deleted files of both file types.

4.2.2.5 **/DF -- Default Switch** - The Default switch (/DF) changes the default device and/or UFD.

The usual default device of PIP is SY0:.

The normal default UFD is the UIC under which PIP is currently running. /DF alters only the default UFD. It does not affect the UIC under which PIP is running, nor does it circumvent file protection.

Specify /DF in one of the following formats:

```
dev:[group,member]/DF

     or

dev:/DF

     or

[group,member]/DF
```

dev:

If specified, the new default device to be applied to subsequent PIP command strings. You must specify the device if you do not specify a UFD.

[group,member]

If specified, the new default UFD to be applied to subsequent PIP command strings. You must specify a UFD if you do not specify a device.

/DF

The Default switch.

Examples

1. PIP>[27,27]/DF

   Sets the default UFD to [27,27].

2. PIP>DK1:/DF

   Sets the default device to DK1:.

3. PIP>DK1:[27,27]/DF

   Sets the default device to DK1: and the default UFD to [27,27].

4.2.2.6 **/EN -- Enter Switch** - The Enter switch (/EN) lets you enter a synonym for a file in a directory or directories on the same device. This allows the file to be accessed by more than one name. Also provided is a subswitch, New Version (/NV), which forces the version number of the file being entered into the directory to a number one greater than the latest version of the file. Specify the Enter switch in the following format:

```
outfile=infile1[,infile2,...infilen]/EN[/NV]
```

outfile

The file specification of the new directory entry. The output file specification has a special property in that the file name, file type, or version may be explicit, wildcard (*), or defaulted (null). A filename, filetype, or version field that is either wildcard (*) or default (null) means that the corresponding field of the input file is to be used.

infile

The file specification for the input file in the format:

dev:[ufd]filename.filetype;version/sw[/subsw]

If you specify a device in either the input or output file specification, that device sets the default for the other side. If you do not specify a device on either the input or output side, the current default device is assumed to be the default device. If both the input side and the output side explicitly reference different devices, PIP signals an error and requests that the line be reentered.

The default input file specification is *.*;*.

/EN

The Enter switch.

/NV

The New Version subswitch. The NV subswitch may appear on either side of the equal sign. If it appears on the output side, all of the files being entered are forced to a version number one greater than the latest version of the file. If it appears on the input side, only files that have the NV subswitch appended to them are forced to a number one greater than the latest version.

Example (see Figure 4-2)

```
PIP>[101,101]TWIG/EN=[200,200]RICK.DAT;1
```

Before

DIRECTORY [200,200]	DIRECTORY [101,101]
RICK.DAT;1	JEN.OBJ;2 LAU.OBJ;3

Figure 4-2 Sample Directories Before and After Execution of /EN

After

DIRECTORY [200,200]	DIRECTORY [101,101]
RICK.DAT;1	JEN.OBJ;2 LAU.OBJ;3 TWIG.DAT;1

NOTE

The directory items for RICK.DAT;1 and TWIG.DAT;1 both reference the same file.

Figure 4-2 (Cont.) Sample Directories Before and After Execution

4.2.2.7 **/EOF -- End-of-File Switch** - The End-of-File switch (/EOF) allows you to specify where the file's end-of-file will be. This helps in certain situations (for example, system crashes) when a file contains useful information but its EOF pointers are wrong, preventing you from obtaining the information.

EOF is an unprotected file attribute. If you are the file owner or have a system-level UIC, you do not need read- or write-access to read or change this attribute. If you are group or world to the file owner's UIC, you need read-access to read the attribute and write-access to change it.

Specify /EOF in the following format:

```
infile1/EOF[:block:byte][,...infilen/EOF[:block:byte]]
```

infile

The input file specification.

The file specification must be issued because a null file name, file type, and version do not default to *.*;*.

block

The block number where the EOF pointer is to be placed. The EOF pointer cannot be placed beyond the highest number of blocks allocated to the file. The block number can be octal or decimal.

byte

The byte location of EOF or the first unused byte of the block. The byte number can be octal or decimal. The maximum value for byte is 777(8).

If you do not enter either of the values for block and byte, PIP places EOF at the last byte of the last block allocated to the file. If you specify a value for either block or byte that is greater than the maximum value allowed, PIP signals an error.

Note that /EOF is local to each file specification and therefore does not default from left to right.

Example

```
PIP>A.TMP/EOF:17:253,AA.TMP/EOF
```

is equivalent to

```
PIP>A.TMP/EOF:17:253,AA.TMP/EOF:22:777
```

where the file AA.TMP has 22 blocks allocated.

4.2.2.8 **/FI -- File Identification Switch** - The File Identification switch (/FI) allows you to access an existing file by its file identification number (File-ID). Specify /FI in the following format:

```
outfile=/FI:filenum:seqnum
```

/FI

The File Identification switch.

filenum

The file number of the file.

seqnum

The sequence number of the file.

The file identification number of the file is assigned by RSX-11 when the file is created. To find the file identification number of a file, use the Full List switch (/FU). /FU displays the file identification number and other information describing the file.

You can use /FI to create a directory entry for a file. For example:

```
FOO.TSK=/FI:301:27/EN
```

Also, you can copy a file using /FI:

```
A.B=/FI:301:27
```

To list the directory file whose identification number is 301,27, use /FI in the format:

```
/FI:301:27/LI
```

4.2.2.9 **/FR -- Free Switch** - The Free switch (/FR) displays the amount of available space on a specified volume and the largest contiguous space on that volume. Specify /FR in the following format:

```
[dev:]/FR
```

If you do not specify dev:, PIP defaults to SY0:.

The format of the information from /FR is shown below.

```
dev: HAS xxxx. BLOCKS FREE, yyyy. BLOCKS USED OUT OF zzzz.
LARGEST CONTIGUOUS SPACE = nnnn. BLOCKS
```

4.2.2.10 **/ID -- Identify Switch** - The Identify switch (/ID) identifies the version of PIP being used. Specify /ID in the following format:

```
/ID
```

When you specify this switch, the version number is listed on the input terminal as follows:

```
PIP VERSION Mvvee
```

vv

The version number.

ee

The edit number.

Example

```
PIP>/ID

PIP VERSION M1332
```

4.2.2.11 **/LI -- List Switch** - The List switch (/LI) lists one or more files contained in a User File Directory, along with their status information. Three alternate mode switches (/BR, /FU, and /TB) allow you a choice of directory listing formats. Table 4-4 describes these switches. Specify /LI in the format:

```
[listfile=]infilel[,infile2,...infilen]/LI[/switch]
```

listfile

The listing file specification in the format:

```
dev:[ufd]filename.filetype;version
```

If listfile is not specified, it defaults to TI:.

infile

The input file specification in the format:

```
dev:[ufd]filename.filetype;version
```

The default for infile is *.*;*.

/LI

The List switch. This switch lists the following information:

1. filename.filetype;version
2. number of blocks used (decimal)
3. file code:

```
(null) = noncontiguous
    C  = contiguous
    L  = locked
```

4. creation date and time

5. summary line, which includes the number of blocks used/allocated and files printed

/switch

The alternate mode switches of the List switch described in Table 4-4.

Table 4-4
List Switches

Switch	Description
/BR	This switch specifies the brief form of directory listing. This switch lists only the file name, file type, and version.
/FU[:n[.]]	This switch specifies the full directory format. Because the /FU format involves protected file attributes, you may need read-access to get a full directory listing of a file. If you are the file owner or have a system-level UIC, you do not need read-access. If you are group or world to the file owner's UIC, you need read-access to read the protected attributes of the file. (To change the protection level attribute, see Section 4.2.2.14, below.) If specified, n is the number of characters per line. If not specified, the number defaults to the buffer size of the output device. This switch lists the following information: 1. filename,filetype;version 2. file identification number in the format: (file number, file sequence number) 3. number of blocks used/allocated (decimal) 4. file code: (null) = noncontiguous C = contiguous L = locked 5. creation date and time 6. owner UIC and file protection in the format: [group,member] [system,owner,group,world]

(continued on next page)

Table 4-4 (Cont.)
List Switches

Switch	Description
/FU[:n[.]] (Cont.)	These protection fields can contain the values R, W, E, or D. where: R = Read access permitted W = Write access permitted E = Extend privilege permitted D = Delete privilege permitted 7. date and time of the last update plus the number of revisions. 8. summary line, which contains the number of blocks used, the number of blocks allocated, and the number of files printed.
/TB	This switch specifies the summary line in the following format: TOTAL OF nnnn./mmmm. BLOCKS IN xxxx. Files where: nnnn = blocks used mmmm = blocks allocated xxxx = number of files

Figure 4-3 contains sample directory listings in the various formats.

Examples

1. PIP>/LI

   Lists the directory of the current default device and UIC. (This is equivalent to TI:=*.*;*/LI.)

2. PIP>LP:=[*.*]/FU:132

   Lists on the line printer in full format (132-column listing), all of the directories on the current default device.

   Note: Only RSX-11M-PLUS has transparent spooling (see Section 4.2.1).

3. PIP>TI:=TEST.DAT/FU

   Lists on TI: the full directory listing for the latest version of TEST.DAT in the current default device and directory.

4. PIP>JUL13.DIR=[200,200]*.*/LI

   Lists the latest version of all files in directory [200,200] on the current default device to file JUL13.DIR in the default directory on the default device.

Total Blocks (/TB) Format

```
STORAGE USED/ALLOCATED FOR DIRECTORY DK2:[200,270]
15-JUL-75 15:46

        TOTAL OF 145./150. BLOCKS IN 5. FILES
```

Brief (/BR) Format

```
DIRECTORY DK2:[200,270]

CKTST.MAC;6
IOTST.MAC;4
IOTST.TSK;1
CKTST.TSK;1
CKTST.MAC;7
```

Standard (/LI) Format

```
DIRECTORY DK2:[200,270]
15-JUL-75 15:46

CKTST.MAC;6          3.          15-JUL-75 15:39
IOTST.MAC;4          4.          15-JUL-75 15:39
IOTST.TSK;1          69.      C  15-JUL-75 15:39
CKTST.TSK;1          69.      C  15-JUL-75 15:40
CKTST.MAC;7          0.       L  15-JUL-75 15:40

        TOTAL OF 145. BLOCKS IN 5. FILES
```

Full (/FU) Format

```
DIRECTORY DK2:[200,270]
15-JUL-75 15:46

CKTST.MAC;6          (10,10)          3./3.             15-JUL-75 15:39
  [200,270][RWED,RWED,RWED,R]
IOTST.MAC;4          (11,11)          4./4.             15-JUL-75 15:39
  [200,270][RWED,RWED,RWED,R]
IOTST.TSK;1          (7,12)           69./69.        C  15-JUL-75 15:39
  [200,270][RWED,RWED,RWED,R]
CKTST.TSK;1          (12,13)          69./69.        C  15-JUL-75 15:40
  [200,270][RWED,RWED,RWED,R]
CKTST.MAC;7          (13,14)          0./5.          L  15-JUL-75 15:40
  [200,270][RWED,RWED,RWED,R]

        TOTAL OF 145./150. BLOCKS IN 5. FILES
```

Figure 4-3 Directory Listing Examples

5. PIP>LP:=[11,*]*.CMD;*/LI

   Lists on the line printer all versions of all files with the file type .CMD in all directories in group 11.

   Note: Only RSX-11M-PLUS has transparent spooling (see Section 4.2.1).

6. PIP>LP:/BR=[11,11]*.CMD;*,*.DAT;*,*.MAC;1

   Lists on the line printer in brief format all versions of all files with a file type of .CMD; all versions of all files with a file type of .DAT; and all files of file type .MAC with a version number of 1. These files all reside in the directory [11,11] on the current default device.

   Note: Only RSX-11M-PLUS has transparent spooling (see Section 4.2.1).

4.2.2.12 **/ME -- Merge Switch** - The Merge switch (/ME) creates a single file from two or more existing files. Merge is used in copying Files-11 files and is described in Section 4.2.1.

4.2.2.13 **/NM -- No Message Switch** - The No Message switch (/NM) suppresses the PIP error message, NO SUCH FILES(S), when you are manipulating files. Specify the NM switch in the following format:

```
infile1[,infile2,...infilen][/sw]/NM
```

infile

The input file specification.

/sw

Any combination of appropriate switches and subswitches, for example, the LI, DE, or PU switches and any of their respective subswitches.

/NM

The No Message switch.

Example

```
PIP>*.MAC;*,TEST.DAT;1,FILES.OBJ;*/DE/NM
```

If none of these files exists in the default directory, you will not get the error message, NO SUCH FILE(S), when PIP tries to delete them.

4.2.2.14 **/PR -- Protect Switch** - The Protect switch (/PR) provides the facility to set the protection status of a file. File protection is provided for four categories:

System

Specifies which categories of access the system UICs are allowed to the file (that is, UICs with group numbers less than or equal to 10 octal).

Owner

Specifies which categories of access the owner has allowed himself.

Group

Specifies which categories of access other members in the same group have.

World

Specifies categories of access given all other UICs.

For each category, you can specify whether that category can read, write, extend, or delete the file. To alter the protection level of a file, you can use either /PR subswitches (/SY, /OW, /GR, /WO) or octal representations (PR:n). For either method, if you are the file owner or have a system-level UIC, you can alter the protection level without having read- or write-access. However, because the protection level of a file is a protected attribute, you cannot alter the protection level if you are group or world to the file owner's UIC. (You can read protected attributes if you have read-access.)

Specify the PR switch in the following format:

```
infile/PR[/SY[:RWED]][/OW[:RWED]][/GR[:RWED]][/WO[:RWED]][/FO]
```

infile

The file specification for the file whose protection is being changed, in the format:

```
dev:[ufd]filename.filetype;version/switch
```

File specification must be issued because a null file name, file type, and version do not default to *.*;*.

/PR

The Protect switch.

/SY,/OW,/GR, and /WO

The subswitches which specify protection level for a file. These subswitches specify which protection level is to be altered (others are left intact). The values which follow the switch are any of the four letters, R, W, E, and D (for read, write, extend, and delete), in any order. They specify which privileges the respective categories can have. If you enter the subswitch and do not specify a value, no privileges are granted for that category.

The subswitches are identified as follows:

/SY is the System subswitch.
/OW is the Owner subswitch.
/GR is the Group subswitch
/WO is the World subswitch.

Protection can also be specified by an optional octal value on the PR switch itself, in the format:

/PR:n

where n is the octal representation of the protection to be assigned to the file. This octal number is taken as the new protection word. (See the RSX-11M Mini-Reference for the list of octal codes.) The format of the protection word is shown in Figure 4-4.

/FO

The Set File Ownership subswitch, which provides the facility to set the ownership of a file to that of the UIC of the directory in which it is entered. (You can change the file ownership at the same time you set the protection value.) If there are files in the [200,200] directory which are owned by another UIC, the command:

PIP>[200,200]*.*;*/PR/FO

causes all files to be owned by [200,200].

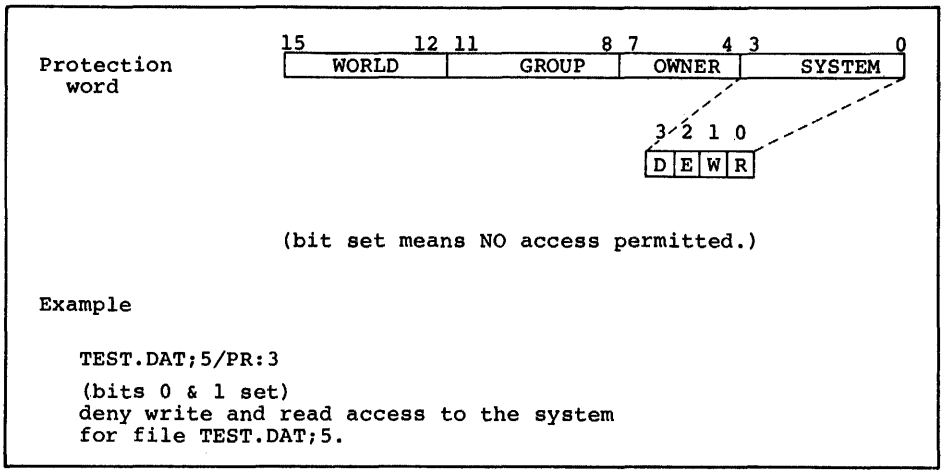


Figure 4-4 Format of Protection Word

Examples

1. PIP>TEST.DAT;5/PR/OW:RWE/GR:RWE:/WO

   Sets the protection level so that the owner and group have RWE privileges (not delete), world has no access privileges, and system privileges are unchanged.

2. PIP>[*,*]*.*;*/PR:0

   Sets the protection level of all files so that all categories are granted all access privileges.

3. PIP>DK0:[*,*]*.*;*/PR/FO

   Causes all file owners to be the same UIC as the UFD in which the files are entered.

4.2.2.15 **/PU -- Purge Switch** - The Purge switch (/PU) deletes a specified range of obsolete versions of a file. Optionally, you can specify that the deleted files be listed on your terminal.

Specify the Purge switch in the following format:

infile1[,infile2,...infilen]/PU[:n][/LD]

infile

The file specification for the file to be deleted. The file specification takes the form:

dev:[ufd]filename.filetype

/PU[:n[.]]

The Purge switch. If you specify the optional value n and the latest version of the file is m, then all existing versions less than or equal to m-n are deleted (see Figure 4-5). Although it is useful to think of this command as deleting all but the n most recent versions, it is important to understand that if any versions are already deleted between m-n and m, fewer than n versions will be retained. The most recent version of the file is always retained.

If you omit the value n, PIP defaults to 1, and all but the latest version of the file are deleted. If n is greater than the number of versions of the specified files, no files are deleted.

The value n is local and defaults from left to right. This means that if you specify n at the end of the command line, it only applies to the infile immediately preceding it. All other infiles default to one. However, n applies to all following infiles until you make a new specification for n.

A version number is not required when using the Purge switch. If specified, it is ignored.

/LD

The List Deleted files subswitch.

Examples

1. PIP>*.OBJ,*.MAC/PU:2/LD

   Deletes all but the highest version of all files with a file type of .OBJ, and all but the two highest versions of all files with a file type of .MAC. Lists all of the deleted files having a file type of .MAC.

2. PIP>*.OBJ/PU:2/LD,*.MAC

   Deletes all but the two highest versions of all files with file types of .OBJ and .MAC. Lists all deleted files.

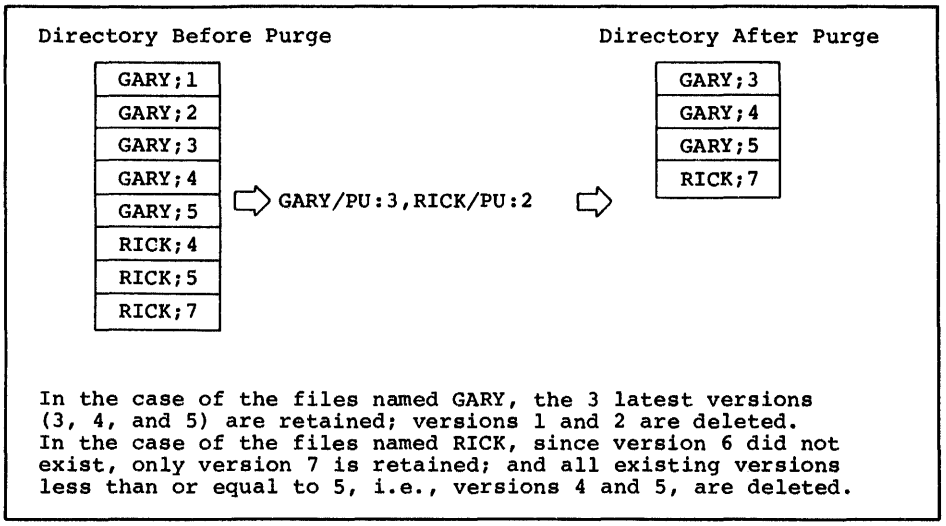


Figure 4-5 Use of the Purge Switch

4.2.2.16 **/RE -- Rename Switch** - The Rename switch (/RE) changes the name of a file. There is also a New Version subswitch (/NV) that forces the renamed file to have a version number one greater than the latest version of the previously existing file with the same name. Specify the Rename switch in the following format:

outfile=infile1[,infile2,...infilen]/RE[/NV]

outfile

The file specification to be given to the new file. The output file specification has a special property in that the file name, file type, and version are each allowed to be explicit, wildcard (*), or defaulted (null). A UFD, filename, filetype, or version field that is either wildcard (*) or defaulted (null) means that the corresponding field of the input file is to be used. Thus, the Rename switch can change one or more fields while preserving the others. Enter the output specification in the following format:

dev:[ufd]filename.filetype;version

infile

The file specification of the file to be renamed. The input file specifications are standard and allow wildcards in all fields, including UFD. Enter this specification in the following format:

```
dev:[ufd]filename.filetype;version
```

A null file name, file type, and version defaults to *.*;*.

/RE does not transfer data. The file is entered in the new directory and deleted from the old directory. The directories must be on the same device because data is not transferred. You can move files out of one directory into another, preserving the file name, file type, and version, or changing them if desired. (This is permitted only if PIP is running under a UIC with write privileges for each of the directories involved.)

If you specify a device on either the input or output side, that device sets the default for the other side. If both the input side and the output side explicitly reference different devices, PIP signals an error and requests that you reenter the line.

/RE

The Rename switch.

/NV

The New Version subswitch. The NV subswitch forces the version number of the renamed file to a number one greater than the latest version for the file.

The NV subswitch may appear on either side of the equal sign. If it appears on the output side, all of the version numbers of files being renamed are forced to a number one greater than the latest version for the file. If it appears on the input side, only the file that has the subswitch appended to it has its version number forced to one greater than the latest version of the file.

Examples

1. PIP>TESTFILE.DAT;1=TEST.DAT;5/RE

   Renames TEST.DAT;5 to TESTFILE.DAT;1.

2. PIP>BACKUP.*;*=TEST1.1;*,TEST2.2;*,TEST3.3;*/RE

   Renames all versions of all files with the file names TEST1, TEST2, and TEST3 to BACKUP, preserving the file type and version of each file.

3. PIP>*.*;1=*.*;*/RE

   Renames all copies of all files to version 1.

NOTE

There should be only one version of each of these files before renaming. Otherwise, PIP continues to rename files until you abort the task.

4. PIP>[200,220]=[200,200]/RE

   Renames all files from [200,200] to [200,220], preserving the file name, file type, and version of each file.

5. PIP>EXAMPLE.*;*=TEST.*;*/RE

   Renames all versions of all files with the file name TEST to the file name EXAMPLE, preserving the file type and version of each file.

6. PIP>SAVE.DAT/RE/NV=OUTPUT.DAT;1

   Renames OUTPUT.DAT;1 and forces the version number to one greater than the latest version of SAVE.DAT. Figure 4-6 illustrates the results with and without the NV switch.

```
Directory Before Rename

SAVE.DAT;2
SAVE.DAT;3
SAVE.DAT;4
OUTPUT.DAT;1
OUTPUT.DAT;2

Directory After Rename Without /NV Switch Set

SAVE.DAT;2
SAVE.DAT;3
SAVE.DAT;4
SAVE.DAT;1
OUTPUT.DAT;2

Directory After Rename With /NV Switch Set

SAVE.DAT;2
SAVE.DAT;3
SAVE.DAT;4
SAVE.DAT;5
OUTPUT.DAT;2
```

Figure 4-6 Results of Rename Switch With and Without /NV Specified

4.2.2.17 **/RM -- Remove Switch** - The Remove switch (/RM) removes an entry from a User File Directory, but does not delete the file associated with that entry. Remove is particularly useful for deleting directory entries which, for whatever reason, point to nonexistent files. It is also used to delete synonyms generated by the Enter switch. If the last entry for an existing file is removed, that file can be located only by using the VFY utility with its LO switch (see Chapter 13). Specify the Remove switch in the format:

```
infile1[,infile2,...infilen]/RM
```

infile

The file specification for the directory file entry to be removed. The file specification takes the form:

dev:[ufd]filename.filetype;version

The file specification must be issued because a null file name, file type, and version do not default to *.*;*.

/RM

The Remove switch.

Example

PIP>DK1:[10,10]RICKSFILE.DAT;1/RM

Removes the file entry RICKSFILE.DAT;1 from the directory [10,10] on DK1:.

4.2.2.18 **/RW -- Rewind Switch** - The Rewind switch (/RW) directs PIP to rewind magnetic tape. (/RW cannot be used for DECtapes.) You can apply this switch to both input and output specifications. When you specify /RW with the output specification, it erases the tape. You can use this technique to erase a tape before writing files on it.

Specify the Rewind switch in the following format:

outfile/RW=infile

or

outfile=infile/RW

outfile

The output file specification.

infile

The input file specification.

/RW

The Rewind switch.

When you apply /RW to the input specification, it rewinds the tape before opening the input file. The magnetic tape processor performs the following process when it searches for a file to open:

1. Searches from the current position to end of tape
2. Rewinds the tape
3. Searches from the beginning of tape to the point where search processing began

You can use /RW with the input specification to save search time. If you know a file is behind the tape's current position, /RW rewinds the tape before searching for the file to open. This saves the time that

otherwise would have been taken to search for the file between the current position and the end of the tape.

4.2.2.19 **/SB -- Span Blocks Switch** - The Span Blocks switch (/SB) allows you to control whether records copied from disk to magnetic tape or vice versa will cross block boundaries. If you omit this switch, the file is copied with records possibly crossing block boundaries. If you specify /-SB, the records will not cross block boundaries.

Specify the Span Blocks switch in the following format:

```
outfile/SB = infile
```

outfile

The disk output file.

infile

The magnetic tape input file.

/SB

The Span Blocks switch.

Example

```
PIP>DK1:T3/-SB=MM0:T3
```

Copies T3 records to the disk from magnetic tape and does not cross block boundaries.

4.2.2.20 **/SD -- Selective Delete Switch** - The Selective Delete switch (/SD) prompts for your response before deleting a file that you have specified in the command line as a candidate for deletion. The response choices are carriage return (<CR>) or control-Z (^Z), or Y, N, G, or Q, each followed by either a carriage return (<CR>) or control-Z (^Z). Table 4-5 describes the effect of each combination of letter and terminator.

Specify the SD switch in the following format:

```
infile1[,infile2,...infilen]/SD
```

infile

The input file specification in the form:

```
dev:[ufd]filename.filetype;version
```

The file specification must be issued because a null file name, file type, and version do not default to *.*;*.

/SD

The Selective Delete switch.

Table 4-5
Response Choices for the Selective Delete (/SD) Switch

Letter	Terminator	Operation
Y	(<CR>)	Delete this file and continue
Y	(^Z)	Delete this file and exit from PIP
N	(<CR>)	Save this file and continue
N	(^Z)	Save this file and exit from PIP
	(<CR>)	Save this file and continue
	(^Z)	Save this file and exit from PIP
Q	(<CR>)	Save this file and return to command mode
Q	(^Z)	Save this file and exit from PIP
G	(<CR>)	Delete this and all remaining candidates, list deleted files, and return to PIP command mode
G	(^Z)	Delete this and all remaining candidates, list deleted files, and exit from PIP

Examples

1. PIP>MYFILE.DAT;*/SD

```
DELETE FILE DB1:[200,200]MYFILE.DAT;1 [Y/N/G/Q]?  Y(<CR>)
DELETE FILE DB1:[200,200]MYFILE.DAT;2 [Y/N/G/Q]?  G(<CR>)

THE FOLLOWING FILES HAVE BEEN DELETED:
DB1:[200,200]MYFILE.DAT;2
DB1:[200,200]MYFILE.DAT;3
PIP>
```

Deletes MYFILE.DAT;1 and PIP goes to the next candidate, MYFILE.DAT;2. Deletes this file and all remaining versions of MYFILE.DAT. Lists the deleted files and returns you to PIP command mode.

2. PIP>TEST.*;*/SD

```
DELETE FILE DB1:[200,200]TEST.DAT;1 [Y/N/G/Q]?  N(<CR>)
DELETE FILE DB1:[200,200]TEST.TXT;3 [Y/N/G/Q]?  Q(^Z)
```

Saves TEST.DAT;1. PIP goes on to the next candidate, TEST.TXT;3. Saves this file and all remaining files with file name TEST. Returns you to monitor control mode.

4.2.2.21 **/SP -- Spool Switch** - The Spool switch (/SP) directs a file to a line printer for printing. This switch applies only if you have the Print Spooler task (RSX-11M) or the Queue Manager (RSX-11M/M-PLUS) installed. (See Chapter 6 for an explanation on using the Queue Manager and Appendix C for a description of the Print Spooler.)

Specify the Spool switch in the following format:

```
infile1[,infile2,...infilen]/SP[:n]
```

infile

The file specification of the file to be spooled for printing. The file specification takes the form:

```
dev:[ufd]filename.filetype;version
```

The file specification must be issued because a null file name, file type, and version do not default to *.*;*.

In RSX-11M and RSX-11M-PLUS only, if the file is specified by its file identification number (File-ID), it will be printed. File identification numbers are discussed in Section 4.2.2.8.

/SP

The Spool switch.

n

The number of copies you want spooled. (If a deleting spooler was specified during SYSGEN, only one copy of a file is printed, regardless of the value of n. The file is deleted after the first copy has been printed.) If n is omitted, a value of 1 is assumed.

Example

```
PIP>RICK1.LST;1,KATHY.LST;1,/FI:12:22/SP
```

Spools the files RICK1.LST;1, KATHY.LST;1, and the file whose file identification number (File-ID) is 12:22 for asynchronous printing.

4.2.2.22 **/SR -- Shared Reading Switch** - The Shared Reading switch (/SR) allows you to read a file that has already been opened for writing by another task. You have no guarantee that you will get the information you want since the EOF pointer may be incorrect at the time you open the file. Specify the SR switch in the following format:

```
outfile=infile/SR
```

outfile

The output file specification.

infile

The input file specification.

/SR

The Shared Reading switch.

Example

```
PIP>TI:=[210,20]FILES.DAT/SR
```

Enables you to read FILES.DAT even though another task may have already opened it for writing.

4.2.2.23 **/TR -- Truncate Switch** - The Truncate switch (/TR) allows you to truncate files back to their logical end-of-file point. Note that RMS-11 files other than those that are fixed-length, variable-length, or sequenced cannot be truncated. Specify /TR in the following format:

```
infile1[,infile2,...infilen]/TR
```

infile

The input file specification.

The file specification must be issued because a null file name, file type, and version do not default to *.*;*.

/TR

The Truncate switch.

Example

```
PIP>*.MAC/TR

     Directory before TRUNCATE

A.MAC;1     3.      20-SEP-78 14:02
B.MAC;1     2.      20-SEP-78 15:38
C.MAC;2     5.      28-SEP-78 09:54

     TOTAL OF 10/15.  BLOCKS IN 3.  FILES


     Directory after TRUNCATE

A.MAC;1     3.      20-SEP-78 14:02
B.MAC;1     2.      20-SEP-78 15:38
C.MAC;2     5.      28-SEP-78 09:54

     TOTAL OF 10./10.  BLOCKS IN 3.  FILES
```

4.2.2.24 **/UF -- User File Directory Switch** - The User File Directory switch (/UF) creates a UFD entry in the Master File Directory (MFD) on the volume to which you are transferring a file. You must also transfer ownership of the file in order to access the file. Use the FO subswitch to transfer file ownership, and use [*,*] as the UFD in the output file specification if you want to assign the UIC under which you are running to all the files being transferred.

Specify the UF switch in the following format:

```
outfile/UF[/FO]=infile,...infilen
```

outfile

The file specification for the output file.

infile

The file specification for the input file.

/UF

The User File Directory switch.

/FO

The File Ownership subswitch. /FO is described above in the section, /AP -- Append Switch.

To use /UF, you must have write-access to the Master File Directory of the volume on which the files are being written. If that volume is a system volume, you must have a system-level UIC to use /UF. If the volume to which you are writing files is your own private volume, use the following procedure to change your UIC so that you can write to it.

1. Log onto the system under your UIC.
2. Reset your UIC to a privileged class using the SET command:

   SET /UIC=[group,member]

   where group and member specify a privileged class.

A typical use of /UF is creation of a backup volume. In the following command, you are writing all files with file types .OBJ and .MAC in UFD [104,20] to a backup volume called DK6:.

```
PIP>DK6:[*,*]/UF/FO=SY:[104,20]*.MAC,[104,20]*.OBJ
```

4.2.2.25 **/UN -- Unlock Switch** - The Unlock switch (/UN) unlocks (gives permission to open) a file that was locked because it was improperly closed. If a program using File Control Services (FCS) has a file open with write-access and exits without first closing the file, the file is locked against further access as a warning that it may not contain proper information. Typically, the following information is not written to the file:

1. The current block buffer being altered
2. The record attributes which contain the end-of-file information

Using the Unlock switch, you can access the file, determine the extent of the damage, and, if possible, take corrective action. Specify the Unlock switch in the following format:

```
infile1[,infile2,...infilen]/UN
```

infile

The file specification for the file to be unlocked. The file specification takes the form:

dev:[ufd]filename.filetype;version

The file specification must be given because a null file name, file type, and version do not default to *.*;*.

You must run PIP under the UIC of the file owner or under a system-level UIC.

/UN

The Unlock switch.

Example

PIP>DK1:[100,100]RICK1.OBJ;3/UN

Unlocks a file RICK1.OBJ;3 in directory [100,100] on device DK1:.

4.2.2.26 **/UP -- Update Switch** - The Update switch (/UP) is similar to the basic PIP copy function or the Merge switch except that an existing file is opened and new data is written into it from the beginning. Existing data in the output file is destroyed and replaced by the data that constitutes the input file(s). Unlike the Supersede switch (/SU, Section 4.2.1), /UP does not delete the existing file before rewriting the data and its file identification number (File-ID) remains the same. Also, the number of blocks allocated to the output file can be the same or greater, but never less than the number of blocks allocated to the existing file. However, as with /SU, the file's name, type, and version number remain the same.

Specify the Update switch in the format:

outfile=infile1[,infile2,...infilen]/UP[/FO]

outfile

The file specification for the file to be rewritten. The file specification takes the form:

dev:[ufd]filename.filetype;version

As in the Merge and the Append switches, the output file specification must be explicit, that is, no wildcards are allowed.

The characteristics and record attributes of the output file are taken from the first input file.

infile

The file specification for the file to be copied into the file that is being rewritten. Specify infile in the format:

dev:[ufd]filename.filetype;version

A null file name, file type, and version default to *.*;*

/UP

The Update switch.

/FO

The Set File Ownership subswitch which specifies that the owning UIC of the output file corresponds to the directory into which the file was entered. If you do not specify /FO, the owning UIC of all new files is the UIC under which PIP is running, regardless of the directory into which the file was entered. Refer to Section 4.2.1 for examples on using the FO subswitch.

Example

```
PIP>DK1:SAMPLE.DAT;1=TEST1.DAT;1,TEST2.DAT;1,TEST3.DAT;1/UP
```

Opens SAMPLE.DAT;1 on DK1: and replaces the data currently in the file with the contents of files TEST1.DAT;1, TEST2.DAT;1 and TEST3.DAT;1.

4.3 PIP ERROR MESSAGES

Errors encountered by PIP during processing are displayed in the following format:

```
PIP -- <main error message>

<filename or filespec> - <secondary error message>
```

The file name or file specification, if present, identifies the file or set of files being processed when the error occurred. If the error was detected by the operating system, file system, or device driver, the secondary error message is included to explain the cause of the error.

PIP error messages are contained in message files on the system device. If PIP cannot access the message files, errors are reported in the following format:

```
PIP -- ERROR CODE nn.

<filename or filespec> - <Driver Code -mm.>

     or

<QIO Error Code -qq.>
```

nn

One of the PIP error codes contained in Table 4-6.

-mm

One of the standard system, file primitive, or file control service codes listed in the IAS/RSX-11 I/O Operations Reference Manual.

-qq

One of the directive error codes listed in IAS/RSX-11 I/O Operations Reference Manual.

The PIP error messages, their descriptions and suggested user actions are as follows:

PIP -- ALLOCATION FAILURE - NO CONTIGUOUS SPACE

Explanation: Not enough contiguous space was available on the output volume for the file being copied.

User Action: Delete all files that are no longer required on the output volume, and reenter the command line. Also, use the BRU or DSC utilities to compress the files on your disk. BRU is described in Chapter 10 and DSC is described in Chapter 11.

PIP -- ALLOCATION FAILURE ON OUTPUT FILE

or

PIP -- ALLOCATION FAILURE - NO SPACE AVAILABLE

Explanation: Not enough space was available on the output volume for the file being copied.

User Action: Delete all files that are no longer required on the output volume, and reenter the command line. Also, use the BRU or DSC utilities to compress the files on your disk. BRU is described in Chapter 10 and DSC is described in Chapter 11.

PIP -- BAD USE OF WILD CARDS IN DESTINATION FILE NAME

Explanation: A wildcard * was specified for an output file name when use of a wildcard was explicitly disallowed.

User Action: Reenter the command line with the proper output file explicitly specified.

PIP -- CANNOT FIND DIRECTORY FILE

Explanation: Specified UFD does not exist on this volume.

User Action: Reenter the command line, specifying the correct UFD or the correct volume.

PIP -- CANNOT FIND FILE(S)

Explanation: The file(s) specified in the command were not found in the designated directory.

User Action: Check the file specification and reenter the command line.

PIP -- CANNOT RENAME FROM ONE DEVICE TO ANOTHER

Explanation: You attempted to rename a file across devices.

User Action: Reenter the command line, renaming the file on the input volume, then enter another command to transfer the file to the intended volume.

PIP -- CANNOT TRUNCATE THIS FILETYPE

Explanation: PIP can only truncate files containing fixed-length, variable-length, and sequenced records.

User Action: Check the file specification and reenter the command line.

PIP -- CLOSE FAILURE ON INPUT FILE

or

PIP -- CLOSE FAILURE ON OUTPUT FILE

Explanation: The input or output file could not be properly closed. The file is then locked to indicate possible corruption.

User Action: Reenter the command line. If the error recurs, run a validity check of the file structure using the Verify utility (VFY) on the volume in question to determine if it is corrupted. VFY is described in Chapter 13.

PIP -- COMMAND SYNTAX ERROR

Explanation: Command did not conform to syntax rules.

User Action: Reenter the command line with the correct syntax.

PIP -- DEVICE NOT MOUNTED/ALLOCATED

Explanation: The device was not mounted, or another user had mounted the device.

User Action: Mount the device, and reenter the command line.

PIP -- DIRECTORY WRITE PROTECTED

Explanation: PIP could not remove an entry from a directory because the device was write-protected, or because of a privilege violation.

User Action: Enable the unit for write operations or have the owner of the directory change its protection.

PIP -- ERROR FROM PARSE

Explanation: The specified directory file does not exist.

User Action: Reenter the command line with the correct UIC specified.

PIP -- EXPLICIT OUTPUT FILENAME REQUIRED

Explanation: Self-explanatory.

User Action: Reenter the command line with the output filename explicitly specified.

PIP -- FAILED TO ATTACH OUTPUT DEVICE

or

PIP -- FAILED TO DETACH OUTPUT DEVICE

Explanation: An attempt to attach/detach a record-oriented output device failed. This is usually caused by the device being offline or non-resident.

User Action: Ensure that the device is online and reenter the command line.

PIP -- FAILED TO CREATE OUTPUT UFD

Explanation: PIP could not create an entry in a directory because the device was write-protected or because of a privilege violation.

User Action: Enable the unit for write operations or have the owner of the directory change its protection.

PIP -- FAILED TO DELETE FILE

or

PIP -- FAILED TO MARK FILE FOR DELETE

Explanation: You attempted to delete a protected file.

User Action: Request PIP under the correct UIC and reenter the command line.

PIP -- FAILED TO ENTER NEW FILE NAME

Explanation: You specified a file that already exists in the directory file, or you did not have the necessary privileges to make entries in the specified directory file.

User Action: Reenter the command line, ensuring that the file name and UFD are specified correctly, or request PIP under the correct UIC and reenter the command line.

PIP -- FAILED TO FIND FILE(S)

Explanation: The file(s) specified in the command line were not found in the designated directory.

User Action: Check the file specification and reenter the command line.

PIP -- FAILED TO GET TIME PARAMETERS

Explanation: An internal system failure occurred while PIP was trying to obtain the current date and time.

User Action: Reenter the command line. If the problem persists, submit a Software Performance Report.

PIP -- FAILED TO OPEN STORAGE BITMAP FILE

Explanation: PIP could not read the specified volume's storage bit map, probably because of a privilege violation.

User Action: Retry by running PIP under a system UIC, or have the system manager change the protection on the storage bit map.

PIP -- FAILED TO READ ATTRIBUTES

Explanation: The volume you specified was corrupted or you did not have the necessary privileges to access the file.

User Action: Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check of the file structure Verification utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 13.

PIP -- FAILED TO REMOVE DIRECTORY ENTRY

Explanation: PIP could not remove an entry from a directory because the unit was write-protected or because of a privilege violation.

User Action: Enable the unit for write operations or have the owner of the directory change its protection.

PIP -- FAILED TO RESTORE ORIGINAL DIRECTORY ENTRY - FILE IS LOST

Explanation: PIP has removed a file from a directory, failed to enter it into another directory (using /RE), and failed to replace the original directory entry.

User Action: Run the lost check of the file structure Verification utility (VFY) to recover the filename. VFY is described in Chapter 13.

PIP -- FAILED TO SPOOL FILE FOR PRINTING

Explanation: Not enough system dynamic memory was available, or the spooler task is not installed.

User Action: Wait for spooler queue to empty, or install the spooler task and reenter the command line.

PIP -- FAILED TO TRUNCATE FILE

Explanation: The volume you specified is corrupted or you did not have the necessary privileges (write, extend) to truncate this file.

User Action: Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check of the file structure Verification utility (VFY) against the Volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 13.

PIP -- FAILED TO WRITE ATTRIBUTES

Explanation: Volume is corrupted or you did not have the necessary privileges to write the file attributes.

User Action: Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check of the file structure Verification utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 13.

PIP -- FILE IS LOST

Explanation: PIP has removed a file from its directory, failed to delete it, and failed to restore the directory entry.

User Action: Run the lost check of the file structure Verification utility (VFY) to recover the filename. VFY is described in Chapter 13.

PIP -- FILE NOT LOCKED

Explanation: The UN switch was entered for a file that was not locked.

User Action: Reenter the command line, specifying the correct file.

PIP -- GET COMMAND LINE - BAD @ FILE NAME

Explanation: An illegal indirect command file name was specified.

User Action: Reenter the command line, specifying the correct name for the indirect command file.

PIP - GET COMMAND LINE - FAILED TO OPEN @ FILE

Explanation: PIP could not find the specified indirect command file.

User Action: Check the specification for the indirect command file and reenter the command line.

PIP -- GET COMMAND LINE - I/O ERROR

Explanation: An I/O error occurred during an attempt to read a command line.

User Action: Check the command to ensure that you entered it correctly, then retry the command. If the error persists, submit a Software Performance Report.

PIP -- GET COMMAND LINE - MAX @ FILE DEPTH EXCEEDED

Explanation: The maximum level of nesting for indirect command files (4) was exceeded.

User Action: Reduce the level of nesting.

PIP -- ILLEGAL COMMAND

Explanation: Command was not recognized by PIP.

User Action: Reenter the command line with the PIP command correctly specified.

PIP -- ILLEGAL SWITCH

Explanation: Specified switch was not a legal PIP switch.

User Action: Reenter the command line with the correct switch specification.

PIP -- ILLEGAL "*" COPY TO SAME DEVICE AND DIRECTORY

Explanation: You attempted to copy all versions of a file into the same directory that is being scanned for input files. This would result in an infinite number of versions of the same file so is not allowed.

User Action: Reenter the command line, renaming the files or copying them into a different directory.

PIP -- ILLEGAL USE OF WILDCARD VERSION

Explanation: The use of a wildcard version number in the attempted operation would result in inconsistent or unpredictable output.

User Action: Reenter the command line with different options or with an explicit or default version number.

PIP -- INPUT FILES HAVE CONFLICTING ATTRIBUTES

Explanation: The input files specified in a Merge, Update, or Supersede command had conflicting attributes or the attributes of the input file(s) specified in an Append command conflicted with those of the output file.

User Action: The message is a warning only. The specified action was completed despite the conflict. In a Merge, Update, or Supersede command, the attributes of the output file will be those of the first input file. In an Append command, the attributes of the output file are unchanged. The resulting file should, however, be suspect because its attributes may not correctly represent all the records in the file.

PIP -- I/O ERROR ON INPUT FILE

or

PIP -- I/O ERROR ON OUTPUT FILE

Explanation: One of the following conditions may exist;

- The device is not online
- The device is not mounted
- The hardware has failed
- The volume is full (output only)
- Input file is corrupted

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

PIP -- NOT A DIRECTORY DEVICE

Explanation: A directory-oriented command was issued to a device that does not have directories (such as a printer).

User Action: Reenter the command line without specifying a UFD.

PIP -- NOT ENOUGH BUFFER SPACE AVAILABLE

Explanation: PIP did not have enough I/O buffer space to perform the requested command.

User Action: Have the system manager install PIP in a larger partition or increase the size specified by the INC switch on the MCR INSTALL command. See the RSX-11M/M-PLUS MCR Operations Manual.

PIP -- NO SUCH FILE(S)

Explanation: The file(s) specified in the command were not found in the designated directory.

User Action: Check the file specification and reenter the command line.

PIP -- ONLY [*,*] IS LEGAL AS DESTINATION UIC

Explanation: A UFD other than [*,*] was specified as the output file UFD for a copy operation.

User Action: Reenter the command line with [*,*] specified as the output UFD.

PIP -- OPEN FAILURE ON INPUT FILE

or

PIP -- OPEN FAILURE ON OUTPUT FILE

Explanation: The specified file could not be opened. One of the following conditions may exist:

- The file is protected against access
- A problem on the physical device (for example, device down)
- The volume is not mounted
- The specified file directory does not exist
- The named file does not exist in the specified directory

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

PIP -- OUTPUT FILE ALREADY EXISTS -- NOT SUPERSEDED

Explanation: An output file of the same name, type, and version as the file specified already exists.

User Action: Retry the copy with /NV to assign a new version number of /SU to supersede the output file.

PIP -- TOO MANY COMMAND SWITCHES - AMBIGUOUS

Explanation: Too many switches were specified, or the switches conflict.

User Action: Reenter the command line, specifying the correct set of switches.

PIP -- VERSION MUST BE EXPLICIT OR "*"

Explanation: The version number of the specified file must be expressed explicitly or as a wildcard *.

User Action: Reenter the command line with the version number correctly expressed.

4.4 PIP ERROR CODES

Table 4-6 identifies the error codes PIP issues when it does not have access to the message files. The descriptions and suggested user actions for these error codes are identical to those described in Section 4.3.

Table 4-6
PIP Error Codes

Error Code	Error Message
1	COMMAND SYNTAX ERROR
2	ILLEGAL SWITCH
3	TOO MANY COMMAND SWITCHES - AMBIGUOUS
4	ONLY [*,*] IS LEGAL AS DESTINATION UIC
5	ILLEGAL COMMAND
6	ILLEGAL "*" COPY TO SAME DEVICE AND DIRECTORY
7	BAD USE OF WILDCARDS IN DESTINATION FILE NAME
8	EXPLICIT OUTPUT FILE NAME REQUIRED
9	ALLOCATION FAILURE - NO CONTIGUOUS SPACE
10	ALLOCATION FAILURE - NO SPACE AVAILABLE
11	ALLOCATION FAILURE ON OUTPUT FILE
12	I/O ERROR ON INPUT FILE
13	I/O ERROR ON OUTPUT FILE
14	ILLEGAL USE OF WILDCARD VERSION
15	FAILED TO CREATE OUTPUT UFD
16	INPUT FILES HAVE CONFLICTING ATTRIBUTES
17	OPEN FAILURE ON INPUT FILE
18	OPEN FAILURE ON OUTPUT FILE
19	CLOSE FAILURE ON INPUT FILE
20	CLOSE FAILURE ON OUTPUT FILE
21	FAILED TO DETACH OUTPUT DEVICE
22	DEVICE NOT MOUNTED/ALLOCATED
23	OUTPUT FILE ALREADY EXISTS - NOT SUPERSEDED
24	FAILED TO MARK FILE FOR DELETE
25	FILE IS LOST
26	VERSION MUST BE EXPLICIT OR "*"
27	ERROR FROM PARSE
28	FAILED TO DELETE FILE
29	CANNOT FIND DIRECTORY FILE
30	FAILED TO ATTACH OUTPUT DEVICE
31	FAILED TO GET TIME PARAMETERS
32	NOT A DIRECTORY DEVICE
33	FAILED TO WRITE ATTRIBUTES
34	FAILED TO READ ATTRIBUTES
35	FILE NOT LOCKED
36	FAILED TO ENTER NEW FILE NAME
37	FAILED TO RESTORE ORIGINAL DIRECTORY ENTRY - FILE IS LOST
38	CANNOT RENAME FROM ONE DEVICE TO ANOTHER
39	FAILED TO SPOOL FILE FOR PRINTING
40	CANNOT SPOOL BY FILE ID (RSX-11D only)
41	FAILED TO OPEN STORAGE BITMAP FILE
42	FAILED TO FIND FILE(S)
43	CANNOT FIND FILE(S)
44	NO SUCH FILE(S)
45	FAILED TO REMOVE DIRECTORY ENTRY
46	DIRECTORY WRITE PROTECTED
47	NOT ENOUGH BUFFER SPACE AVAILABLE
48	FAILED TO TRUNCATE FILE
49	CANNOT TRUNCATE THIS FILETYPE

CHAPTER 5

FILE TRANSFER PROGRAM (FLX)

The File Transfer Program (FLX) is a file utility program that transfers files from one volume to another. In addition, when a file is being transferred from a volume with a different format than the one the file is going to, FLX converts the format of the file to conform to the format of the volume to which the file is transferred. FLX allows you to:

- List directories of cassettes, RT-11, or DOS-11 volumes
- Delete files from DOS-11 and RT-11 file-structured volumes
- Initialize cassettes, RT-11, or DOS-11 volumes

FLX performs file transfers (and format conversions, as appropriate) from:

- DOS-11 to Files-11 volumes
- Files-11 to DOS-11 volumes
- DOS-11 to DOS-11 volumes
- Files-11 to Files-11 volumes
- Files-11 to RT-11 volumes
- RT-11 to RT-11 volumes
- RT-11 to Files-11 volumes

Valid DOS-11 devices are:

Device Abbreviation	Device
CT	TU60 tape cassette
DK	RK05 cartridge disk
DT	TU56 DECtape
MM	TE16, TU16, TU45, or TU77 magnetic tape
MS	TS04 magnetic tape
MT	TU10 or TS03 magnetic tape
PP	PC11 paper tape punch
PR	PC11 or PR11 paper tape reader

Valid RT-11 devices are:

Device Abbreviation	Device
DD	TU58 DECtape II data cartridge
DK	RK05 cartridge disk
DL	RL01 cartridge disk
DM	RK06 or RK07 cartridge disk
DT	TU56 DECtape
DX	RX01 flexible disk
DY	RX02 flexible disk

FLX supports all Files-11 devices, including RSX-format cassettes. Files-11 volumes are the default volumes. They are volumes that you have initialized using the MCR command, INITVOL. DOS-11 and RT-11 volumes are initialized using FLX. On RSX-11M-PLUS only, DOS-11 and RT-11 volumes must be mounted with foreign characteristics before you can use FLX.

The switches for the three formats are: /RS for Files-11 format, /DO for DOS-11 format, and /RT for RT-11 format. These switches are described in Section 5.2.

You can use FLX interactively or by means of an indirect command file. FLX allows only one level of indirect command file specification.

You can invoke FLX in two ways: by specifying FLX or by specifying FLX and a command string. If you only specify FLX, the utility responds with the prompt:

```
FLX>
```

5.1 FLX COMMAND STRING

Although formats for specifying FLX functions vary, the general format for entering FLX command strings is:

```
devicespec/sw=infile/sw,...,infilen/sw
```

devicename

The device specification for the FLX output device, which takes the form:

```
dev:[ufd]
```

The [ufd] field is optional; if it is not specified, FLX uses the current UIC.

If you explicitly enter the output device specification, you must enter the equal sign.

FLX does not permit output file specifications. The output files take the names of the input files.

infilen

The input file specifications, which are entered as described in Chapter 1.

/sw

One of three types of FLX switches described below in Section 5.2.

FLX now supports 9-character file names for DOS-11-format magnetic tapes.

Wildcards are valid only for input file specifications.

Version numbers are valid only for Files-11 files and cannot be specified as wildcards. The standard rules for updating version numbers apply (see the RSX-11M/M-PLUS MCR Operations Manual).

5.2 FLX SWITCHES

FLX provides three types of switches for file transfers:

- Volume format switches
- Transfer mode switches
- Control switches

Volume format switches specify the format of the volume on which files are stored; that is, Files-11, DOS-11, or RT-11 volumes.

Transfer mode switches provide the means for specifying the format of a file on a non-Files-11 volume. Files can be in formatted ASCII, formatted binary, or file image format.

Control switches provide control functions useful during file transfers. Using file control switches, you can specify, for example, the number of blocks to be allocated to an output file or the UFD for an output file.

5.2.1 Volume Format Switches

FLX has three volume format switches that define the format of the specified volumes. Table 5-1 describes these switches.

Table 5-1
FLX Volume Format Switches

Switch	Description
/DO	Identifies the volume as a DOS-11 formatted volume.
/RS	Identifies the volume as a Files-11 formatted volume.
/RT	Identifies the volume as an RT-11 formatted volume.

Initially, input volumes default to DOS-11 format and output volumes default to Files-11 format. FLX assumes these default volume formats if you do not specify switches in the command string.

You can change the initial default by entering /RS or /DO on a command line by itself. /RS sets the default for input volumes to Files-11 format and output volumes to DOS-11 format. /DO sets the default for input volumes to DOS-11 format and output volumes to Files-11 format.

For example, to specify the default transfer direction from Files-11 to DOS-11, type:

```
FLX>/RS
```

To specify the default transfer direction from DOS-11 to Files-11, type:

```
FLX>/DO
```

If /RT is specified on one side of a command string, the default entry for the other side is /RS.

Examples

1. FLX>DK0:=DT0:SYS1.MAC/RT

   The output is defaulted to /RS.

2. FLX>DK0:/RT=DK0:SYS1.MAC

   The input is defaulted to /RS.

5.2.2 Transfer Mode Switches

FLX has three transfer mode switches, one for each type of file format. Files can be in formatted ASCII, formatted binary, or file image format. Format conversions can be in either direction, and are between DOS-11 files and Files-11 files or between RT-11 files and Files-11 files. Specifying a transfer mode switch determines which format the output file will be in after the conversion of the file. Table 5-2 describes the transfer mode switches.

Table 5-2
FLX Transfer Mode Switches

Switch	Description
/FA:n	Formatted ASCII The DOS-11 or RT-11 output file is to be formatted ASCII. Formatted ASCII is defined as ASCII data records terminated by carriage return/line feed (CR-LF), form feed (FF), or vertical tab (VT). In transfers from DOS-11 or RT-11 files to Files-11 files, CR-LF pairs are removed from the end of records. In transfers from Files-11 files to DOS-11 or RT-11 files, CR-LF pairs are added to the end of each record that does not already end with LF or FF. In both directions, all nulls, rubouts, and vertical tabs are removed from input records. If you specify /FA:n with Files-11 output, fixed-length records of size n are generated. Output records are padded with nulls, if necessary.

(continued on next page)

Table 5-2 (Cont.)
FLX Transfer Mode Switches

Switch	Description
	If you do not specify /FA:n with Files-11 output, FLX generates variable-length records. The output record size equals the input record size. ASCII data is transferred as 7-bit values. Bit 8 of each byte is masked off before transfer. CTRL/Z (ASCII 032 octal) is treated as the logical end of input file for formatted ASCII transfers from DOS-11 cassette or paper tape to Files-11.
/FB:n	Formatted Binary The DOS-11 or RT-11 output file is to be formatted binary. In this mode, formatted binary headers and checksums are added to records output to DOS-11 or RT-11 files, and removed when transferred to Files-11 files. If you specify /FB:n with Files-11 output, fixed-length records of size /FB:n are output (512 decimal bytes is the maximum). FLX pads records with nulls to create the specified length. If you do not specify /FB:n with Files-11 output, FLX generates variable-length records. The output record size equals the input record size.
/IM:n	Image Mode The transfer is to be in image mode. Image mode forces fixed-length records. You can use the value n to indicate the desired record length (in decimal bytes) for Files-11 output (512 decimal bytes maximum). If you do not specify n, FLX assumes a record length of 512(10) bytes.

FLX assumes the following default transfer modes for these file types (with the exception of paper tape transfers; see Section 5.6).

Mode	Switch	File Type
Image	/IM:n	.TSK, .OLB, .MLB, .SYS, .SML, .ULB
Formatted Binary	/FB:n	.OBJ, .STB, .BIN, .LDA
Formatted ASCII	/FA:n	All others

If you specify n with /FA, /FB, or /IM when the output file is not a Files-11 file, FLX ignores n.

See the RSX-11M/M-PLUS MCR Operations Manual for the definitions of the above file types.

5.2.3 Control Switches

FLX provides a number of control switches to control file processing. Table 5-3 describes these switches.

Table 5-3
FLX Control Switches

Switch	Description
/BL:n	Indicates the number of contiguous blocks (n) to be allocated to the output file. This switch is normally used with /CO (described below). If you do not specify /BL, the input file size is used as the output file size. The file allocation scheme used for RT-11 volumes normally allocates the largest available space on the volume for a new file. Using /BL:n with the RT switch for the output file causes the output file to be allocated the first unused space of size n. However, when the RT-11 file is closed, the input file size is used as the output file size. If the input file is not n, an error results. Because all RT-11 files are contiguous, the CO switch is not required with the BL:n switch for RT-11 output.
/BS:n	Specifies the block size (n) in bytes for cassette tape output. If you do not specify /BS, a block size of 128(10) is assumed. /BS is only valid in a cassette tape (CT) output file specification with /RS specified.
/CO	Indicates that the output file is to be contiguous. The /CO switch is used only with disks and DECtapes. If the input file is paper tape, cassette, or DOS-11 magnetic tape, /BL is also required. FLX transfers the file types .TSK, .SYS, and .OLB to Files-11 volumes with /CO implied when the input is a Files-11 volume, or a DOS-11 or RT-11 DECtape or disk.
/DE	Deletes files from a DOS-11 DECtape or disk. It is used also with /RT to delete files from an RT-11 DECtape or disk. When you specify /DE, the FLX command string has no output specification.

(continued on next page)

Table 5-3 (Cont.)
FLX Control Switches

Switch	Description
/DI	Causes a directory listing of cassettes or DOS volumes to be listed on a specified output file. It is used also with /RT to generate a directory listing of RT-11 volumes in a specified output file. You cannot list Files-11 volume directories using FLX. If you do not specify an output device, the directory is sent to TI:. If you do not specify file name and file type on the input file specification, *.* is assumed. See Section 5.3 for information on DOS-volume directory manipulation. See Section 5.4 for information on RT-11-volume directory manipulation.
/DNS:n	Specifies the density of the magnetic tape; n is either 800 or 1600. If n is any other value or not specified at all, FLX prints an error message. If you do not specify /DNS:n, the magnetic tape density defaults to 800 bpi. If you specify /DNS with a non-magnetic-tape device, FLX ignores the switch.
/FC	When using FORTRAN files, indicates that FORTRAN carriage control conventions are to be used, that is, FORTRAN interprets certain characters in the FORTRAN files as carriage control characters. The FC switch applies only to Files-11 output files. (If you have the PDP-11 FORTRAN Language Reference Manual, refer to it for more information on FORTRAN carriage control conventions. Otherwise, refer to the IAS/RSX-11 I/O Operations Reference Manual for a discussion of the file data block and record attributes, of which setting carriage control is a part.)
/ID	Requests the current version number of FLX to be printed. You can specify /ID as part of an output or input specification or type it in response to the FLX prompt (FLX>).
/LI	Same as /DI.
/NU:n	Used with the /ZE and /RT switches to specify the number of directory blocks (n) to allocate when initializing an RT-11 disk or DECtape. If you do not specify /NU:n, four directory blocks are allocated. The maximum number of blocks which can be allocated is 37(8) (31 decimal).

(continued on next page)

Table 5-3 (Cont.)
FLX Control Switches

Switch	Description
/RW	Rewinds the magnetic tape before beginning the file transfer. Specifying /-RW causes FLX to begin the transfer without first rewinding the magnetic tape. If you do not specify either rewind option, the switch defaults to /RW. If you specify the rewind switch with a non-magnetic-tape device, or with /LI, /DI, or /ZE, FLX ignores the switch.
/SP	Indicates that the converted file is to be spooled by means of the old print spooler task or the queue management system. The SP switch applies only to Files-11 output files. The print spooler is described in Appendix C and the queue management system is described in Chapter 7.
/UI	Indicates that the output file is to have the same UFD as the input file. FLX ignores the UI switch if the output specification contains an explicit UFD. /UI is valid only for output files in DOS-11 or Files-11 format.
/VE	Causes each record written to the cassette to be read and verified. The VE switch is only valid with a CT output file specification.
/ZE	Initializes cassettes or DOS-11 volumes. It is also used with /RT (and /NU) to initialize RT-11 volumes. Initializing erases any files already on the device. For DOS-11 DECtape, /ZE creates an entry for the current UIC. The ZE switch does not require a file specification.

5.3 DOS-11 VOLUME DIRECTORY MANIPULATION

This section contains examples that show how to display DOS-11 directory listings, delete DOS-11 files, and initialize DOS-11 volumes using the FLX switches.

On RSX-11M-PLUS only, DOS-11 volumes must be mounted with foreign characteristics before you can use FLX.

5.3.1 Displaying DOS-11 Directory Listings

The LI or the DI switch instructs FLX to send the directory of the cassette or DOS-11 volume specified in the input specification to the Files-11 file specified in the output specification. If you do not enter an output specification, FLX sends the directory to TI:. For example:

```
FLX>DT0:[100,100]*.MAC/LI
```

This command lists on your terminal the directory of all .MAC files under UIC [100,100] on the DOS-11 DECtape on DT0:.

Figure 5-1 shows sample directory listings for a DOS-11 DECtape and a TU60 cassette.

```
DECtape Directory Listing

(1) DIRECTORY  (2) DT:[200,200] (3)
(4) 19-SEP-78

(5) FLX.TSK        (6) 104.  (7) 19-SEP-78  <233> (10)
    UFD.TSK          8.      19-SEP-78  <233>
    TKN.TSK          6.      19-SEP-78  <233>
    MOU.TSK          14.     19-SEP-78  <233>

 (9) TOTAL OF 132. BLOCKS IN 4. FILES

Cassette Directory Listing

(1) DIRECTORY  (2) CT:[200,200] (3)
(4) 19-SEP-78

(5) UFD.TSK-0              (6) 28.  (7) 19-SEP-78 128. (8)
    TKN.TSK-0                20.      19-SEP-78 128.
    MOU.TSK-0                52.      19-SEP-78 128.

 (9) TOTAL OF 100. BLOCKS IN 3. FILES
```

Figure 5-1 DOS-11 Directory Listings

Notes on Figure 5-1:

(1) This line identifies the listing as a directory listing.

(2) The device name and unit number.

(3) The User File Directory.

(4) The date the directory was listed.

(5) The file name, file type, version number, and sequence number (cassettes only).

(6) The file size in decimal blocks.

(7) The file creation date.

(8) The record size in decimal bytes for the file (cassettes only).

(9) A total of the actual file sizes and the total number of files in the directory.

(10) The protection code (disk and DECtape only). A protection code of 233 is the default code provided by the system. This code allows read, run, write, and delete access to the owner, and read and run (but not delete or write) access to the group and world. See the DOS/Batch Handbook for more information on protection codes.

5.3.2 Deleting DOS-11 Files

You can delete files from DOS-11 disks or DECtapes using the Delete switch (/DE). The command string on which you specify /DE requires only the file specification for the file you are deleting. For example:

```
FLX>DK1:[100,100]SYS1.MAC/DE
```

This command deletes SYS1.MAC under UFD [100,100] from the DOS-11 disk on DK1:.

5.3.3 Initializing DOS-11 Volumes

You can initialize cassettes and DOS-11 volumes using the ZE switch. This switch requires only the device specification for the volume you are initializing. For example:

```
FLX>DT1:/ZE
```

This command initializes the DECtape on DT1: in DOS-11 format.

5.4 RT-11 VOLUME DIRECTORY MANIPULATION

You can display RT-11 directory listings, delete RT-11 files, and initialize RT-11 volumes using the FLX switches described in this section.

On RSX-11M-PLUS only, RT-11 volumes must be mounted with foreign characteristics before you can use FLX.

5.4.1 Displaying RT-11 Directory Listings

The LI or the DI switch, when combined with the RT switch, instructs FLX to send the directory of the RT-11 volume in the input specifications to the Files-11 file in the output specification. If you do not enter an output specification, FLX sends the directory to TI:. For example:

```
FLX>DT0:*.MAC/LI/RT
```

This command lists on your terminal all .MAC files on the RT-11 volume on DT0:.

Figure 5-2 shows a sample directory listing of an RT-11 disk.

```
(1)DIRECTORY (2)DK:
(3) 4-JUN-78

(4)SIPBOO.MAC (5) 49. (6)4-JUN-78
< UNUSED >     6.
SIP    .MAC    10.    4-JUN-78
SIPCD .MAC      7.    4-JUN-78
< UNUSED >     21.
SIPQIO.MAC      7.    4-JUN-78
< UNUSED > 4686.

(7) 4713. FREE BLOCKS

(8) TOTAL OF 73. BLOCKS IN 4. FILES
```

Figure 5-2 RT-11 RK05 Cartridge Disk Directory Listing

Notes on Figure 5-2:

(1) This line identifies the listing as a directory listing.

(2) The device name and unit number.

(3) The date the directory was listed.

(4) The file name and file type; <UNUSED> indicates free space.

(5) The number of blocks in the file or free space.

(6) The file creation date, or blank for free space.

(7) The total number of free blocks on the volume.

(8) The total number of blocks allocated to files on the volume.

5.4.2 Deleting RT-11 Files

You can delete files from RT-11 disks or DECtapes using the Delete switch (/DE) with the RT-11 switch (/RT). The command string on which you specify /DE/RT requires only the file specification for the file you are deleting. For example:

```
FLX>DK1:SYS1.MAC/DE/RT
```

This command deletes SYS1.MAC from the RT-11 volume on DK1:.

5.4.3 Initializing RT-11 Volumes

You can initialize RT-11 volumes using the ZE switch with the RT switch. /ZE requires only the device specification for the volume you are initializing. For example:

```
FLX>DT1:/ZE/RT
```

This command initializes the DECtape on DT1: in RT-11 format.

When you initialize RT-11 volumes, /ZE takes an optional argument in the form:

```
/ZE:n
```

The value n specifies the number of extra words per dictionary entry. A directory segment consists of two disk blocks with a total of 512(10) words. The directory header uses five words, leaving 507(10) words for directory entries.

Normally, each directory entry is seven words long; two directory entries within each directory segment are allocated to the file system. Therefore, the number of entries in each segment (when no extra words are specified) is determined as follows:

```
Directory entries = (507/7)-2

                  = 72-2=70 entries
```

Some RT-11 applications require extra words in the directory entries. (The default is no extra words.) When you specify extra words for directory entries (by means of /ZE:n), the number of directory entries is determined as follows:

```
Directory entries = [507/(n+7)]-2
```

For example, 61(10) entries can be made per directory segment if you specify /ZE:1.

Use of /NU with /ZE and /RT specifies the number of directory segments to allocate to the RT-11 volume. The NU switch has the following form:

```
/NU:n
```

The value n specifies the number of directory segments to allocate. Four directory segments are allocated by default. The maximum number of segments that can be allocated is 37(8) or 31(10). For example:

```
FLX>DT0:/ZE:2/NU:6/RT
```

This command initializes the DECtape on DT0: in RT-11 format, allocates two extra words per directory entry, and allocates six directory segments. This results in a total of 54(10) directory entries, each of which is 9 words long.

5.5 FLX TA11/TU60 CASSETTE SUPPORT

FLX supports the DIGITAL standard cassette file structure. Files can be transferred to and from cassettes in either Files-11 format (/RS) or DOS-11 format (/DO). The transfer mode selected depends on the file format requirements.

The file formats for Files-11 or DOS-11 cassette files are almost the same; that is, they both conform to the DIGITAL standard cassette file format. The differences between the Files-11 and DOS-11 cassette file formats are as follows:

Files-11 Format	DOS-11 Format
Standard level 2	Standard level 0
12-character file name (9-character file name and 3-character extension)	9-character file name (6-character file name and 3-character extension)
Blocks of any size up to 512(10) bytes (128 decimal bytes default)	128(10)-byte blocks
Version numbers	No version numbers

Files-11 cassette file format (level 2) is a superset of the DOS-11 cassette file format (level 0). Therefore, any cassette written in DOS-11 format can be read in Files-11 format. The reverse of this, however, is true only when:

- The Files-11 file is written with 128(10)-byte blocks
- The extra file header data (such as version number), which does not appear in DOS-11 files, can be ignored

Files-11 files and DOS-11 files can be mixed on a given cassette as long as you use a proper retrieval mode when you access the file. Files of various blocksizes can also share a given cassette. FLX uses the block size contained in the file label data when reading a file.

5.5.1 Multivolume Cassette Support

FLX supports multivolume cassettes in both Files-11 and DOS-11 formats. No special switches are required to notify FLX that a multivolume file is being accessed.

5.5.2 FLX Cassette Output Files

When FLX detects the physical end-of-tape for an output cassette, the following sequence of events occurs.

1. FLX issues the following message:

   ```
   FLX -- END OF VOLUME ON CASSETTE
   CTn:[g,m]
   ```

   where n, g, and m specify the unit number, group number, and member number.

2. The cassette is rewound.

3. FLX issues an additional message:

   ```
   MOUNT NEW CASSETTE?  (Y, Z (OUTPUT ONLY) OR CR)
   FLX>
   ```

4. At this point, you have three alternatives:

   a. Mount the next output cassette volume and type Y, followed by a carriage return. If you select this alternative, the new cassette is rewound, FLX searches for the logical end-of-tape (end of the last file), and then continues transferring data onto the tape. If FLX, while searching for logical end-of-tape, encounters a file with the same file name as the current input file, it displays the following message:

      FLX -- FILE ALREADY EXISTS

      FLX then returns to step 3 above.

   b. Mount the next output cassette volume and type Z, followed by a carriage return. The new cassette is rewound, and FLX continues by transferring data onto it. Thus, the tape is effectively zeroed before data is transferred to it.

   c. Enter a carriage return to terminate the transfer.

If you choose alternative 4c, FLX assumes that EOF is desired, and issues the following message:

FLX -- REQUEST TERMINATED -- LAST BLOCK NOT WRITTEN

This message indicates that the last input file block processed was not written onto the tape.

5.5.3 FLX Cassette Input Files

When FLX detects the physical end-of-tape for an input cassette, the following sequence of events occurs:

1. FLX issues the following message, including the input file specification on which the end-of-tape was detected:

   ```
   FLX -- END OF VOLUME ON CASSETTE
   CTn:[g,m]filename.type
   ```

   where n, g, and m specify the unit number, group number and member number.

2. The cassette is rewound.

3. FLX issues an additional message:

   ```
   MOUNT NEW CASSETTE:  (Y, Z (OUTPUT ONLY) OR CR)
   FLX>
   ```

4. At this point you have two alternatives:

   a. Mount the next input cassette volume and type Y, followed by a carriage return to continue.

   b. Type a carriage return to terminate the transfer.

If you choose alternative 4a, the new input cassette is rewound, and a validity check is performed on the file label and sequence number. If the file label and sequence number are correct, FLX begins processing data from the volume. If, however, the file label and sequence number are not correct, FLX issues the following message:

```
FLX -- FILE NOT FOUND
```

The process then returns to step 3 above.

If you choose alternative 4b, FLX assumes that EOF is desired, and the transfer is terminated. If the input file is being processed as a formatted binary or an ASCII file, a format error may occur.

If you enter Z, FLX prints the message:

```
FLX -- BAD RESPONSE
```

The process then returns to step 3 above.

5.6 FLX PAPER TAPE SUPPORT

FLX supports the standard DIGITAL paper tape devices, such as the PC-11 Paper Tape Reader/Punch and the PR-11 Paper Tape Reader, as DOS-11 devices.

FLX provides the ability to delimit records on paper tape for files in formatted binary mode or in formatted ASCII mode. Formatted binary records are delimited by standard DOS-11 4-byte headers and a trailing checksum. Formatted ASCII records that do not already end with line feeds or form feeds are delimited by carriage return-line feed pairs.

FLX gives special treatment to files that normally default to image mode transfers, that is, .TSK, .OLB, .MLB, .SYS, .SML, and .ULB files. On output to paper tape, these files are written, by default, in formatted binary. When read back from paper tape to a Files-11 volume, the file is written with fixed-length, 512(10)-byte records as the default.

These defaults ensure that when the files are read back from paper tape they are in the same format as they were before being punched. However, the new files are not contiguous unless you specify /CO/BL:n with the output file specification. You must know an appropriate value for n (the number of contiguous blocks to allocate) before issuing the command. You can use PIP to create a contiguous file (see Chapter 4).

The use of explicit transfer mode switches when transferring .TSK, .OLB, .MLB, .SYS, .SML and .ULB files between paper tape and Files-11 volumes can cause files read back in from paper tape to be different from the files that were originally written out.

For FLX paper tape transfer commands, you cannot specify file names in the output specification. The file name entered for the input file specification is used as the file name for the output file. For example:

```
FLX>DK1:/RS=PR:CRTMAC.DAT/DO
```

This command writes a file whose file name is DK1:CRTMAC.DAT.

If you do not specify a file name on the input file specification, the default file name is .;n where n represents the latest version number.

RSX-11M and RSX-11M-PLUS systems support paper tapes only as DOS-11 devices; you should specify /DO with paper tape file specifications. RSX-11D supports all formats. The examples below show paper tape specifications for input and output file specifications:

```
FLX>PP:/DO=CRTMAC.DAT/RS
FLX>DK:/RS=PR:CRTMAC.DAT/DO
```

To copy paper tapes from one paper tape to another, use the image-mode switch (/IM) regardless of the format of the paper tapes. For example:

```
FLX>PP:/DO/IM=PR:/DO
```

5.7 FLX ERROR MESSAGES

Errors encountered by FLX during processing are reported on the initiating terminal. The FLX error messages, their descriptions, and suggested user actions are described below.

FLX -- BAD LIST FILE SPEC

Explanation: One of the following was specified:

1. More than one output file for an /LI or /DI operation
2. Wildcards in the output file for an /LI or /DI operation

User Action: Reenter the command line correctly.

FLX -- BAD RESPONSE

Explanation: Z was entered in response to the message:

```
MOUNT NEW CASSETTE (Y, Z (OUTPUT ONLY) OR CR)
FLX>
```

The cassette in guestion is an input volume.

User Action: Respond with Y or CR after the message is redisplayed.

FLX -- CAN'T OPEN @ FILE

Explanation: The specified indirect command file could not be opened for one of the following reasons:

- The file is protected against access.
- A problem exists on the physical device (for example, disk is not spinning).
- The volume is not mounted or is allocated to another user.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.
- The volume is not online.

User Action: Correct the condition and reenter the command line.

FLX -- CO FILES TO OUTPUT DEVICE NOT ALLOWED

Explanation: An illegal output device (for example, MT, CT, or PP) was entered with the CO switch.

User Action: Reenter the command line without /CO specified.

FLX -- CASSETTE ERROR I/O TERMINATED

Explanation: An unexpected hardware error occurred during the end-of-volume sequence on a cassette volume. The transfer was aborted.

User Action: Reenter the command line using a new cassette.

FLX -- COMMAND SYNTAX ERROR

Explanation: The command was entered in a format that does not conform to syntax rules.

User Action: Reenter the command line with the correct syntax.

FLX -- CONFLICTING TRANSFER MODES SPECIFIED

Explanation: Conflicting transfer mode switches were entered. For example:

```
SY:=DT:FOO.OBJ/IM/FB
```

User Action: Reenter the command line with only one transfer mode switch specified.

FLX -- DOS-11 OR RT-11 DEVICE NOT VALID FORMAT

Explanation: The device specified with /DO has an incorrect DOS-11 file structure, or the device specified with /RT has an incorrect RT-11 file structure.

User Action: Correctly identify the file structure on each volume, and reenter the command line.

FLX -- DT: UFD FULL

Explanation: The DECtape directory is full.

User Action: Clean up the directory by deleting all unnecessary files.

FLX -- END OF VOLUME ON CASSETTE
MOUNT NEW CASSETTE? (Y, Z (OUTPUT ONLY) OR CR)

Explanation: Physical end-of-tape was encountered during a cassette transfer. The tape is rewound, and you are asked to mount the next cassette.

User Action: See Section 5.5.2 if an output transfer is being performed; or Section 5.5.3 if an input transfer is being performed.

FLX -- ERROR DURING DIRECTORY I/O

Explanation: One of the following conditions may exist:

1. The volume is not write-enabled.
2. The DO, RT or RS switches were incorrectly specified.
3. The volume is not of the proper format.
4. A hardware error occurred during a directory I/O operation (that is, bad tape).

User Action: The following responses correspond (by number) to the conditions listed above.

1. Write-enable the volume.
2. Respecify /DO, /RT, or /RS correctly.
3. No recovery is possible with the volume currently mounted. Mount a volume that is in the proper format, and retry the operation.
4. Retry the operation.

FLX -- FILE ALREADY EXISTS

Explanation: The specified output file already exists on the output device.

User Action: Reenter the file specification using a new or corrected file name.

FLX -- FILE NOT FOUND

Explanation: The named file does not appear, as specified, in the requested directory.

User Action: Retry the operation with the file name and directory correctly specified.

FLX -- WARNING -- INPUT FILE OUT OF SEQUENCE

Explanation: A cassette multivolume file is being accessed out of sequence.

User Action: This is a warning message. The transfer will continue unless you terminate it by means of the ABORT command.

FLX -- @ FILE NESTING EXCEEDED

Explanation: More than one level of indirect command file was specified.

User Action: Retry the operation with only one level of indirect command file specified.

FLX -- @ FILE SYNTAX ERROR

Explanation: A syntax error occurred in the indirect command file specification.

User Action: Edit the indirect command file. Rerun FLX using the corrected indirect command file.

FLX -- FMTD ASCII RECORD FORMAT BAD

or

FLX -- FMTD BINARY RECORD FORMAT BAD

Explanation: Either the file is corrupted, or the file is not of the specified type.

User Action: If the file is corrupted, no recovery is possible. If the file type is incorrect, retry the operation specifying the correct transfer mode switch.

FLX -- ILLEGAL /BS SIZE -- USE 0<N<=512. AND EVEN

Explanation: An illegal block size was specified with /BS on cassette output.

User Action: Reenter the command line with a legal block size.

FLX -- INCORRECT # IN/OUT SPECS

Explanation: More than one input or output specification in a command was entered where only one is allowed.

User Action: Reenter the command line with the proper syntax.

FLX -- INVALID DEVICE

Explanation: A device was specified that cannot be used as an input or output device, for example, trying to read from a line printer.

User Action: Reenter the command line with a legal device specified.

FLX -- INVALID DOS OR RT-11 FILE SPEC

or

FLX -- INVALID RSX FILE SPEC

Explanation: The file specification does not conform to proper syntax, or the specified operation could not be performed on the specified device.

User Action: Reenter the file specification with the proper syntax.

FLX -- INVALID SYNTAX

Explanation: A switch was entered that is not a valid FLX switch or does not conform to proper syntax.

User Action: Reenter the command line with a correct switch specification.

FLX -- I/O ERROR

Explanation: One of the following conditions may exist:

- The specified device is offline.
- A hardware error occurred (for example, bad tape).

User Action: Ensure that the device is online. Reenter the command line. If a hardware error occurred, recovery may not be possible.

FLX -- I/O ERROR DELETING LINKED FILE

Explanation: An uncorrectable error occurred while a DOS linked file was being deleted.

User Action: No action required; the file is effectively deleted, but the volume may be corrupted.

FLX -- I/O ERROR INITIALIZING DIRECTORY

Explanation: One of the following conditions may exist:

- The specified directory is not online.
- The specified volume is not mounted.
- A hardware error occurred (for example, bad tape).

User Action: Ensure that the device is online and is operable. Reenter the command line with the required switch specified.

FLX -- I/O ERROR ON COMMAND INPUT

Explanation: An unexpected error in command input was encountered from either an indirect command file, or TI:; FLX exits.

User Action: Restart FLX.

FLX -- I/O ERROR ON FLX TEMPORARY FILE

Explanation: FLX encountered an error condition with its temporary file. FLX creates a temporary file on SY: for operations involving DOS-11 CT, DT, or MT. This error occurs when one of the following conditions exists:

- SY: is not online and mounted.
- SY: is write-locked.
- A protection violation occurred.
- A hardware error was encountered.

User Action: Correct the error condition and reenter the command line.

FLX -- I/O ERROR ON LIST FILE

Explanation: An error occurred on the output device during a /DI or /LI sequence. There is a hardware problem with the output device (for example, device powered down).

User Action: Correct the condition. Reenter the command line.

FLX -- OUTPUT DEVICE FULL

Explanation: The DOS-11 or RT-11 output volume does not contain enough space for the output file.

User Action: Delete all unnecessary files and reenter the command line.

FLX -- OUTPUT FILE SPEC NOT ALLOWED

Explanation: An output file specification was entered for a command that does not allow one.

User Action: Reenter the command without an output file specification.

FLX -- RECORD TOO LARGE

Explanation: FLX detected an input record in a Files-11 transfer that is larger than the specified or implied record size for the file, that is, the file is corrupted.

User Action: The file in question is unusable.

FLX -- REQUEST TERMINATED -- LAST BLOCK NOT WRITTEN

Explanation: The <CR> reply was given to indicate that no new volume would be mounted when an end-of-volume was encountered on cassette output. The block that FLX was attempting to write when it encountered the end of the cassette has not been written; the output file is incomplete.

User Action: This message is informational; no action is required.

FLX -- WARNING -- SPECIFIED RECORD SIZE BAD, 512. USED

Explanation: The record size specified with /FA, /FB, or /IM is not acceptable. A record size of 512(10) bytes is assumed.

User Action: This is a warning message. No action is required.

FLX --UNABLE TO ALLOCATE FILE

Explanation: No space is available on the DOS-11 or Files-11 volume for the specified file.

User Action: Delete all unnecessary files and reenter the command line.

FLX -- UNABLE TO OPEN FILE

Explanation: A specified input or output Files-11 file could not be opened. Possible reasons are:

- The input file does not exist.
- The volume is not mounted.
- A protection violation occurred.

User Action: Correct the condition and reenter the command line.

FLX -- UNABLE TO OPEN LIST FILE

Explanation: The list file cannot be opened under the specified file name and UFD, or the specified device may not be a valid Files-11 volume.

User Action: Reenter the command line specifying the correct file name and UFD.

FLX -- UNDIAGNOSABLE REQUEST

Explanation: FLX does not recognize the command line syntax.

User Action: Reenter the command line with the proper syntax.

FLX -- /CO FILES FROM INPUT DEVICE NOT ALLOWED UNLESS BL: SPEC

Explanation: When transferring files from MT, PR, or CT, /CO can only be specified when /BL is also specified.

User Action: Reenter the command line, specifying /BL.

FLX -- * IN VERSION NUMBER NOT ALLOWED

Explanation: A wildcard was detected in the version number field of a file specification.

User Action: Reenter the command line with all version numbers explicitly specified.

PART 4
FILE SPOOLING UTILITIES

CHAPTER 6

PRINT AND QUEUE UTILITY

This chapter is for RSX-11M only. Users of RSX-11M-PLUS should see the RSX-11M-PLUS Batch and Queue Operations Manual for corresponding information.

6.1 INTRODUCTION TO THE PRINT COMMAND AND QUEUE MANAGER

The Print command spools print jobs and places them in a queue controlled by the Queue Manager for despooling.

The most common use of this command is printing files on the system's line printer. Some systems may use a high-speed terminal as a line printer. There may be more than one output device receiving jobs from the Queue Manager.

Other likely output devices include paper-tape punches, electrostatic plotters, or magtapes. Jobs sent to such devices are still called print jobs because they are sent by the Print command.

In general, in this chapter, "to print" and "to despool," and "printer" and "output device," are synonyms.

"Spool" is an acronym for Shared Peripheral Operations On-Line. Spooling on RSX-11M is gathering output sent to an output device on a mass-storage device - usually a disk - to be passed in an orderly fashion to the output device. Despooling is the orderly transfer of this output from the mass-storage device to the output device.

The Print command requires the presence of the Queue Manager. Not all installations have the Queue Manager.

To be sure, check the list of installed tasks (issue the TAL command) to see if both QMG... and PRT... are installed. If both are installed, your system includes the Queue Manager. If only PRT... is installed, your system has the serial print despooler. In this case, you can only use the PIP /SP switch to send files to the line printer. If neither is installed, you must use PIP to send files directly to the line printer with a command in the form PIP LP0:=filespec. If you use this command, you have no protection against another user issuing the same command at the same time, in which case your output and the other user's may be interspersed on the line printer.

If you have any doubts about the situation at your installation, see your system management for information. Documentation on the serial despooler appears in Appendix C of this manual. For further information on PIP, see Chapter 4 of this manual.

The Print command, in its simplest form,

```
>PRI filespec
```

places an entry in the file SY:[1,7]QUEUE.SYS. This queue entry has a unique number and a name based on the first six characters of the file name. The entry is placed in the default queue PRINT. When the line printer is free, the file is despooled by the print processor and passed by the processor to the line printer and printed. After printing, the entry disappears from the queue file in [1,7]. Some or all of the contents of the queue file can be displayed by using QUE commands discussed later in this chapter.

The Print command accepts multiple file specifications and wildcards. Files are despooled in the order given, without interruption, on the same print processor. With wildcards, the files are despooled in the order they appear in the UFD, as displayed by a PIP /LI command.

The Print command defines a print job for the Queue Manager. A print job can have one or more files in it.

Switches on the Print command can specify many attributes of the print job. These attributes include the following:

- Time the spooling is to be done
- Device to accept spooled output
- Queue priority of the job
- Restartability of the job
- Forms the job is to be printed on
- Number of lines per page
- Number of copies of each file to be printed
- Choice of deletion after spooling

Commands to the Queue Manager permit the user to display the queues or an individual job in a queue or to modify most attributes of a job in a queue. Users can also hold a job in a queue, release it, or remove it from a queue.

You do not have to invoke the Queue Manager to get access to it. The Print command does this automatically. Commands directly to the Queue Manager are on one line and start with QUE. Unlike other RSX-11M utilities, the Queue Manager does not issue a QUE> prompt. The Queue Manager can only be accessed through the one-line format. User commands to the Queue Manager are described in this chapter. For full details see Chapter 7 of this manual.

A number of system tasks can create printable output. The output includes Task Builder maps, MACRO-11 assembly listings, and listings from the high-level language compilers. These tasks are accessible directly from MCR. These tasks follow their own rules for spooling output, but when you request that such output be spooled, it will be handled by the Queue Manager.

You cannot control the despooling of such output directly, as you can through the Print command, but once the jobs appear in a queue, you can alter their attributes through the QUE /MOD commands.

If the Queue Manager is not installed and the serial despooler is installed, system tasks that create printable output work as usual. If neither the Queue Manager nor the serial despooler is installed, the map or listing file is created and written on the disk, but is not printed on the line printer.

You can also use PIP, the Peripheral Interchange Program, to queue print jobs.

The following PIP command:

```
PIP> NAME.TYP /SP
```

causes the file NAME.TYP to be queued for despooling. The print job has a name derived from the name of the first file in the job.

You cannot control the despooling of such jobs directly, as you can through the Print command, but once the jobs appear in a queue, you can alter their attributes through the QUE /MOD command.

For information on writing user tasks to spool output, see Section 6.5.

6.2 PRINT COMMAND FORMAT

The format of the Print command is:

```
>PRINT [[ddnn:]jobname [/jsw]=filespec [/fsw [/fsw]][,filespec [/fsw]]
```

ddnn:

Specifies the despool device to be used.

jobname

Specifies the name to be given to the print job.

/jsw

Specifies one or more job-related switches

filespec

Specifies the file to be spooled, in standard RSX-11M format.

fsw

Specifies one or more file-related switches.

There are defaults for most of these fields:

- ddnn: defaults to the PRINT queue.
- jobname defaults to a name derived from the first six characters of the file name.
- filespec defaults are SY: for the device, current UFD, and .LST for the file type.
- Switch defaults are included in the command descriptions in Section 6.3.

6.3 PRINT COMMAND DESCRIPTION

The Print command passes print jobs to the Queue Manager for despooling, as described in Section 6.1.

Multiple filespecs, separated by commas, are acceptable. Wildcards (*) are also acceptable.

A detailed description of each field of the Print command is provided below.

ddnn:

Each output device is served by a device-specific queue. This queue is named after the device it serves. Thus, LP0: has the device-specific queue LPQ0, TT21: (if set spooled) has the device-specific queue TTQ21. To send a print job to a specific device, include the device name in this field. The Queue Manager directs the job to the proper queue for that device. If the file does not exist, the command is rejected.

The device named must be set spooled or the job will not run.

If no device is named, jobs are sent to the default queue PRINT.

jobname

Job names can be from one to six alphanumeric characters. The job is displayed in the queue under this name. The name also appears on the job flag page. See Figure 6-1 for the layout of the job and file flag pages.

If no job name is supplied, the name is derived from the first six characters of the first file name in the job. If the first file name in the job is *, then the name PRINT is given to the job.

Each job also has a queue entry number that is displayed in the QUEUE /LI display.

/jsw

Job-related switches must be placed to the left of the equals (=). The = is required syntax if you include any job-related switches.

All switches can be abbreviated to two characters, not counting any preceding NO or minus (-) if permitted. Switches can be listed in any order.

A blank before the slash (/) is accepted but not required.

A description of each of the job-related switches is provided below.

/AFter:hh:mm

The job is time-blocked in its queue until after the stated time; hh can be from 0 to 23; mm can be from 0 to 59.

If this switch is not specified, the job is enqueued after 0:00 on the present day, that is, immediately.

This switch does not set the time the job will be printed. It sets the time after which the job is eligible to be printed. If there are other jobs in the queue ahead of it when the specified time arrives, these jobs are printed before your job.

This switch setting can be changed after the job is placed in its queue by the QUE /MOD command.

/[NO]FLag

If the /FL switch is applied, each file in the job is preceded by one or more file flag pages (depending on how the printer is set up) in addition to the one or more job flag pages that precede the entire job.

The default for files is /NOFLAG. Only job flag pages are printed. Files are printed sequentially, separated only by form feeds.

A SYSGEN option permits overriding the default and forcing flag pages. If this option has been applied, it can be overridden by setting /NOFLAG.

The job flag pages have the UIC the Print command was issued under and the job name printed on them in large banner characters.

The file flag pages have the file name and type printed on them in large banner characters. See Figure 6-1 for the layout of the job and file flag pages.

This switch setting can be changed after the job is placed in its queue by the QUE /MOD command.

/PRiority:n

This switch sets the priority of the job in its queue. Priority can be from 0 through 250.

The default priority is 50.

The Queue Manager places jobs in the queue according to priority. Jobs of equal priority appear in the queue in the order they were submitted. The highest priority job is the first job processed when the printer becomes available. Within a group of jobs of the same priority, the one that has been waiting the longest is processed first.

This switch setting can be changed after the job is placed in its queue by the QUE /MOD command.

/HOld

This switch holds the job in its queue until it is released by the QUE jobname /RElease command. See the examples at the end of this section.

```
0123456789@123456789    ** RSX-11M V3.2 **    [303,5]FISH   - NO PAGE LIMIT        25-MAY-79    13:41:32    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    FORM #0 - NORMAL HARDWARE FORMS       25-MAY-79    13:41:32    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    NO IMPLIED FORM FEED                  25-MAY-79    13:41:32    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    DB0:[303,5]FISH.SWM;1                 25-MAY-79    13:41:32    0123456789@123456789
```

```
  [[[[[[      333333       @@@@@@       333333                5555555555     ]]]]]]
  [[[[[[      333333       @@@@@@       333333                5555555555     ]]]]]]
  [[        33      33    @@      @@    33      33              55               ]]
  [[        33      33    @@      @@    33      33              55               ]]
  [[                33    @@    @@@@      33      33            555555           ]]
  [[                33    @@    @@@@      33      33            555555           ]]
  [[              33      @@  @@  @@        33        ''''          55           ]]
  [[              33      @@  @@  @@        33        ''''          55           ]]
  [[                33    @@@@    @@        33        ''''            55         ]]
  [[                33    @@@@    @@        33        ''''            55         ]]
  [[        33      33    @@      @@    33      33      ''      55      55       ]]
  [[        33      33    @@      @@    33      33      ''      55      55       ]]
  [[[[[[      333333       @@@@@@       333333        ''          555555       ]]]]]]
  [[[[[[      333333       @@@@@@       333333        ''          555555       ]]]]]]
```

```
FFFFFFFFFF     IIIIII       SSSSSSSS    MM      MM
FFFFFFFFFF     IIIIII       SSSSSSSS    MM      MM
FF               II       SS            MM      MM
FF               II       SS            MM      MM
FF               II       SS            MM      MM
FF               II       SS            MM      MM
FFFFFFFF         II         SSSSSS      MMMMMMMMMM
FFFFFFFF         II         SSSSSS      MMMMMMMMMM
FF               II                SS   MM      MM
FF               II                SS   MM      MM
FF               II                SS   MM      MM
FF               II                SS   MM      MM
FF             IIIIII     SSSSSSSS      MM      MM
FF             IIIIII     SSSSSSSS      MM      MM
```

```
0123456789@123456789    ** RSX-11M V3.2 **    [303,5]FISH   - NO PAGE LIMIT        25-MAY-79    13:41:32    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    FORM #0 - NORMAL HARDWARE FORMS       25-MAY-79    13:41:32    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    NO IMPLIED FORM FEED                  25-MAY-79    13:41:32    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    DB0:[303,5]FISH.SWM;1                 25-MAY-79    13:41:32    0123456789@123456789
```

```
0123456789@123456789    ** RSX-11M V3.2 **    DB0:[303,5]FISH.SWM;1                 25-MAY-79    13:41:35    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    COPY 1 OF 1                           25-MAY-79    13:41:35    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    DELETION NOT SPECIFIED                25-MAY-79    13:41:35    0123456789@123456789
```

```
FFFFFFFFFF     IIIIII       SSSSSSSS    MM      MM
FFFFFFFFFF     IIIIII       SSSSSSSS    MM      MM
FF               II       SS            MM      MM
FF               II       SS            MM      MM
FF               II       SS            MM      MM
FF               II       SS            MM      MM
FFFFFFFF         II         SSSSSS      MMMMMMMMMM
FFFFFFFF         II         SSSSSS      MMMMMMMMMM
FF               II                SS   MM      MM
FF               II                SS   MM      MM      ****
FF               II                SS   MM      MM      ****
FF             IIIIII     SSSSSSSS      MM      MM      ****
FF             IIIIII     SSSSSSSS      MM      MM      ****
```

```
  SSSSSSSS   WW      WW   MM      MM       ;;;;         11
  SSSSSSSS   WW      WW   MM      MM       ;;;;         11
SS           WW      WW   MMMM  MMMM       ;;;;       1111
SS           WW      WW   MMMM  MMMM       ;;;;       1111
SS           WW      WW   MM  MM  MM                    11
SS           WW      WW   MM  MM  MM                    11
  SSSSSS     WW      WW   MM      MM       ;;;;         11
  SSSSSS     WW      WW   MM      MM       ;;;;         11
       SS    WW  WW  WW   MM      MM       ;;;;         11
       SS    WW  WW  WW   MM      MM       ;;;;         11
       SS    WWWW  WWWW   MM      MM         ;;         11
       SS    WWWW  WWWW   MM      MM         ;;         11
SSSSSSSS     WW      WW   MM      MM       ;;         111111
SSSSSSSS     WW      WW   MM      MM       ;;         111111
```

```
0123456789@123456789    ** RSX-11M V3.2 **    DB0:[303,5]FISH.SWM;1                 25-MAY-79    13:41:35    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    COPY 1 OF 1                           25-MAY-79    13:41:35    0123456789@123456789
0123456789@123456789    ** RSX-11M V3.2 **    DELETION NOT SPECIFIED                25-MAY-79    13:41:35    0123456789@123456789
```

Figure 6-1 Job Flag Page and File Flag Page

/LEngth:n

This switch sets the number of lines per page. Length can be set from 1 through 65535. When this switch is applied, the print processor issues a form feed if one is not encountered within n lines. A form feed causes the line printer to move the paper down to the beginning of the next form. Form length is determined by the hardware and the print processor software.

The default is a page of infinite length.

This switch setting can be changed after the job is placed in its queue by the QUE /MOD command.

/FOrm:n

This switch sets the forms setting that an output device must have to accept the job. Your system manager has information on which forms used in your installation take which numbers. If this switch is set, the job waits until a printer with the matching forms setting is available.

The default n is 0, which is the standard line-printer setup in most installations.

This switch is used to print jobs on special forms, such as checks, invoices, or other special forms used in your installation.

This switch setting can be changed after the job is placed in its queue by the QUE /MOD command.

/[NO]REstart

This switch makes the job restartable [or not]. If /RE is specified and the job does not complete its run for some reason, when the system, or print processor, or Queue Manager is restarted, the job is restarted from the beginning of the first file in the job.

/NORESTART is the default. If the job's run is interrupted, it is restarted at the top of the file that was being despooled at the time the job was interrupted.

In no case is any job or part of a job lost, but if you set the switch, the job will be all in one piece.

This switch setting can be changed after the job is placed in its queue by the QUE /MOD command.

filespec[s]

Users can print any file to which they have Read access.

The usual rules on wildcards and defaults apply, except that the default file type is .LST.

Multiple filespecs must be separated by commas.

Files cannot be renamed or deleted before they are printed. Files must have the same name and type, be on the same volume, and reside in the same directory from the time they are placed in the queue until they are despooled.

If the files are on a volume on a medium mounted on a private device, the volume cannot be removed or dismounted before the files are despooled. If the volume is dismounted, the print job remains in the queue and the flag pages are printed, with an error message noting a failure to open the file.

/fsw

File-related switches can follow any filespec in the command line.

File-related switches are applied to all filespecs that follow the filespec to which they are attached unless overridden. See the examples at the end of this section.

All switches can be abbreviated as indicated, not counting any NO or minus (-) if permitted. Switches can be listed in any order.

/COpies:n

This switch sets the number of copies of the file to be despooled.

If applied to multiple filespecs, each file is printed n times before any copies of the next file are printed. That is, this switch does not queue up two copies of the entire job, it only determines how many copies of each file are to be made.

If you do not specify this switch, the default n is 1.

/[NO]DELete

This switch directs the Queue Manager to delete the file from its UFD after it is despooled.

NODELete is the default.

Examples

```
>PRINT JAM
```

The most recent version of JAM.LST from the default device and UFD goes in the default queue PRINT taking all defaults.

```
>PRINT /FLAG=PAUL.MAC,ALAN.MAC,TOM.MAC
```

The files named go to the default queue PRINT as one print job. They are printed in the order listed. The job name is PAUL. The name appears on the first flag page along with the UIC the Print command was issued under. Each individual file is preceded by file flag pages with the file name, file type, and version number. Any command including a job-related switch, such as FLAG, must include the equals (=).

```
>PRI /FL=PAUL.MAC,ALAN.MAC /CO:2,TOM.MAC
```

The Print command defines a single print job consisting of one copy of PAUL.MAC, followed by two copies of ALAN.MAC, and two copies of TOM.MAC. Each copy of each file is preceded by flag pages.

```
>PRI /FORM:2 /PRIORITY:75=PAYROLL
```

The job name is PAYROL. It has a priority of 75, but it will not be despooled until an output device with the proper forms - paychecks in this case - is available. The file is PAYROLL.LST.

```
>PRI INDEX /AFTER:16:00=CHAP1.NDX,CHAP2.NDX,CHAP3.NDX,CHAP4.NDX
```

The job is named INDEX. It is time-blocked in the PRINT queue until after 4:00 pm. At that time, it is placed in contention for the printer, but the time it will be printed depends on what else is in the queue when INDEX is released. This is the equivalent of issuing the same Print command at exactly 16:00.

6.4 QUEUE MANAGER COMMAND FORMAT

The format of commands to the Queue Manager is

```
>QUE job-id /switch[:option]
```

The job identification (job-id) can be either the name assigned to the job at the time of submission or the unique entry number assigned the job by the Queue Manager and displayed by the QUE /LI command.

The entry number is displayed in the form (n1,n2). It is used in commands in the form JOB:n1:n2.

Queue Manager commands allow the nonprivileged user to

- List queue entries in various formats
- Modify attributes of jobs after submission
- Hold jobs in a queue
- Release held jobs
- Delete jobs from a queue

All Queue Manager commands are in single-line format.

6.4.1 QUEUE /LIST Command

This command is used to list the queue entries. Unless you specify options, or name an output device, only jobs in the default queue PRINT are displayed.

The display also shows which output devices queues are assigned to.

Format

```
>QUE job-id /sw[:option]
```

Switches	Options
/BRief	:QUE
/LIst	:ALLQUE
/FUll	:DEV
	:ALLDEV

Job Identification Field

There are three ways you can specify the contents of this field. Details on each are provided below.

[ddnn:[uic]]jobname

The job name alone is sufficient identification if the job is from the same UIC you are logged in to and if you have not specified a device in your Print command. Although the UIC is optional, the brackets around the UIC must be included if you include the UIC in the command line. If you wish to display only entries queued to a particular output device, name the output device in the command. Each output device has a queue named after it (device LP0: has queue LPQ0). When you name the device, you are specifying the queue whose contents you wish to display.

/JOB:n1:n2

Note that the display format and the command format for this entry number are different. Display: (n1,n2). Command: /JOB:n1:n2.

[uic]

This form of job identification displays all jobs in queues from a particular UIC.

Switches

/BRief

The brief format displays only the job names, the UIC from which the Print command was issued, the queue entry number, and the status of the job. Jobs can be ACTIVE, HELD, PRINT AFTER (time-blocked), or waiting. A blank status field indicates a waiting job.

/LIst

The standard format displays job names, the UIC from which the Print command was issued, the queue entry number, the status of the job, and the filespecs of all files necessary to complete the job.

If you have not specified restartability in the Print command, only the file being printed and any remaining files are displayed.

If you have specified restartability, then all files in the job are displayed.

/FUll

The full format displays job names, the UIC from which the Print command was issued, the queue entry number, the status of the job, the filespecs of all files necessary to complete the job, and all attributes of the job, whether set explicitly or by default.

Options

:QUE

This is the default. Displays entries in the default queue PRINT.

:ALLQUE

Displays all entries in all queues.

:DEV

Displays characteristics of a device. If you have not specified a device in the job-identification field of the command line, you will get an error message.

:ALLDEV

Displays characteristics of all spooled devices.

Examples

```
>QUE /LIST

** PRINT QUEUES **
PRINT => LP0:
  [304,303] SCNDIR (20000,126)                ACTIVE ON LP0:
        DB0:[304,303]SCNDIR.LST;174
  [7,372]   RELABE (2100,130)
        DB0:[7,372]RELABEL.FLO;2
```

This example displays all the jobs in the default queue PRINT.

The display shows the line printer to which the queue is assigned.

For each job in the queue, the display shows the UIC from which the Print command was issued, the job name (based on the name of the first file in the job), a unique queue entry number, the status of the job, and the filespecs for all files in the job.

The job named SCNDIR is active, as shown. The job named RELABE is waiting, as indicated by a blank status field. Status field may also show HELD jobs and PRINT AFTER (time-blocked) jobs.

```
>QUE /LIST:QUE

** PRINT QUEUES **
PRINT  => LP0:
  [304,303] SCNDIR (2000,126)                 ACTIVE ON LP0:
        DB0:[304,303]SCNDIR.LST;174
  [7,372]   RELABE (2100,130)
        DB0:[7,372]RELABEL.FLO;2
```

This example is the equivalent of the previous example. The option :QUE is the default option. Only jobs in the default queue PRINT are displayed.

```
>QUE LP1: /LIST

** PRINT QUEUES **
LPQ1  => LP1:                                   ACTIVE ON LP1:
  [303,5] IZZY (2400,132)
        DB0:[303,5]IZZY.TXT;1
```

This example displays queue LPQ1, which is assigned to printer LP1:. The user specified this queue by naming the printer in the QUE /LI command.

```
>QUE PRTJOB /LIST /FU

** PRINT QUEUES **
PRINT  => (LP1:)
  [301,370]PRTJOB (2140,162) FORM:0
      PRI:50 LEN:0 PAGE:0 NORESTART NOFLAG
        DB0:[301,370]PRTJOB.TXT;22     COP:1          NODELETE
```

This example displays in full mode information about the job named PRTJOB from the default queue PRINT.

The display shows all attributes of the job, either explicitly set or set by default.

NOTE

Parentheses around the line printer name mean that the printer has been stopped by the Queue Manager at the operator's request.

```
>QUE [301,370 /BR

** PRINT QUEUES **
PRINT => LP0:
  [301,370]DUNGEO   (2200,116)              ACTIVE ON LP0:
  [301,370]RUNOFF   (1320,660)              HELD
```

This example displays in brief mode all the jobs created under [301,370] in the default queue PRINT. No filespecs are displayed.

```
>QUE /LI:ALL

** PRINT QUEUES **
PRINT => LP0:
  [7,372] RELABE (2100,130)                 ACTIVE ON LP0:
        DB0:[7,372]RELABEL.FLO;2
  [7,372] LPDRV (2000,134)                  HELD
        DB0:[7,372]LPDRV.MAC;2
LPQ0 => LP0:
  [222,333] CORFIL (2200,136)               PRINT AFTER 18-DEC-78 17:00
        DB0:[222,333]CORFIL.COR;4
```

This example displays all the jobs in all the queues. All jobs are displayed because there is no job identification given. All queues are displayed because :ALL is specified.

```
>QUE /FU:ALL

** PRINT QUEUES **
PRINT => LP0:
  [7,372] RELABE (2100,130) FORM:0          ACTIVE ON LP0:
        PRI:50 LEN:0 NORESTART FLAG
          DB0:[7,372]RELABEL.FLO;2 COP:1 NODELETE
  [7,372] LPDRV (2000,134) FORM:0 HELD
        PRI:50 LEN:55 PAGE:0 NORESTART NOFLAG
          DB0:[7,372]LPDRV.MAC;2 COP:50 NODELETE
LPQ0 => LP0:
  [222,333] CORFIL (2200,136) FORM:1 PRINT AFTER 18-DEC-78 17:00
        PRI:100 LEN:0 PAGE:0 RESTART NOFLAG
          DB0:[222,333]CORFIL.COR;4 COP:1 DELETE
```

This example displays in full format all the jobs in all the queues. All jobs are displayed because there is no job identification given. All queues are displayed because :ALL is specified.

```
>QUE LP: /LI:DEV

** SPOOLED DEVICES **
LP0:  <= PRINT LPQO
         FORM:0 FLAG:1
         CURRENT JOB:  [7,2]SYSXTN          (2000,127)
```

This example displays all spooled devices, shows queue assignments to them, and shows the attributes of the devices.

If there is no current job on a device, the current-job line does not appear.

6.4.2 QUE /MOD Command

This command is used to modify the attributes given to queue entries by the Print command. Only the attributes listed here can be modified.

Format

```
>QUE job-id /MODify /sw
```

Switches

```
/AFTER:hh:mm
/[NO]FLAG
/LEngth:n
/PRiority:n
/FOrm:n
/[NO]REstart
```

Job Identification Field

There are two ways you can specify the contents of this field. Details on each are provided below.

[ddnn:[uic]]jobname

The job name alone is sufficient identification if the job is from the same UIC you are logged in to and if you have not specified a device in your Print command. Although the UIC is optional, the brackets around the UIC must be included if you include the UIC in the command line.

/JOB:n1:n2

Note that the display format and the command format for this entry number are different. Display: (n1,n2). Command: /JOB:n1:n2.

Switches

Switches name the attribute to be modified. All switches can be abbreviated as indicated, not counting any NO or - if permitted.

The QUE /FU commands display attribute information. The Print command description includes a discussion of the meaning of each attribute.

/AFter:hh:mm

Modifies the time after which the job is to be despooled.

/[NO]FLag

Modifies the FLAG attributes of the job.

/LEngth:n

Modifies the LENGTH attribute of the job, which is the number of lines to the page.

/PRiority:n

Modifies the queue priority of the job.

/FOrm:n

Modifies the FORM attribute of the job.

/[NO]RESTart

Modifies the RESTART attribute of the job.

6.4.3 QUEUE /HOLD Command

This command holds a job in its queue. The job will not be despooled.

Format

>QUE job-id /HOLD

Job Identification Field

There are two ways you can specify the contents of this field. Details on each are provided below.

[ddnn:[uic]]jobname

The job name alone is sufficient identification if the job is from the same UIC you are logged in to and if you have not specified a device in your Print command. Although the UIC is optional, the brackets around the UIC must be included if you include the UIC in the commmand line.

/JOB:n1:n2

Note that the display format and the command format for this entry number are different. Display: (n1,n2). Command: /JOB:n1:n2.

Switch

/HOLD

The job is held in its queue. The job remains in its queue, but is not eligible for processing. The effect is the same as the /HOLD qualifier to the Print command.

You cannot hold an active job. An active job is a job currently being printed.

Examples

```
>QUE BIGJOB /HOLD
```

The job named BIGJOB from the default device and UIC is held in the PRINT queue.

```
>QUE LP1:[303,5]LILJOB /HOLD
```

The job named LILJOB from UIC [303,5] is held in the queue for LP1:. This queue is named LPQ1.

6.4.4 QUEUE /RELEASE Command

This command releases held jobs from their queues. The job to be released must be marked HELD. You cannot release PRINT AFTER jobs with this command. You must use QUE /MOD /AFTER to release such time-blocked jobs.

Format

```
>QUE job-id /RELease
```

Job Identification Field

There are two ways you can specify the contents of this field. Details on each are provided below.

[ddnn:[uic]]jobname

The job name alone is sufficient identification if the job is from the same UIC you are logged in to and if you have not specified a device in your Print command. Although the UIC is optional, the brackets around the UIC must be included if you include the UIC in the command line.

/JOB:n1:n2

Note that the display format and the command format for this entry number are different. Display: (n1,n2). Command: /JOB:n1:n2.

Switch

/RELease

Releases a held job making it eligible for processing. This is equivalent to placing the job in its queue. The job must still wait its turn to be despooled.

Example

```
>QUE READY /REL
```

The job named READY from the default device and UIC is released from the held state and is eligible for processing.

```
>QUE LATER /REL
QUE -- JOB DOES NOT EXIST
```

The user attempted to release a PRINT AFTER (time-blocked) job. The job exists, but no HELD job of that name exists.

6.4.5 QUEUE /DELETE Command

This command deletes print jobs from queues.

It does not delete files from directories. It has no connection to the PIP /DE or PRINT /DE command switches.

Format

```
>QUE job-id /DELETE
```

Job Identification Field

There are two ways you can specify the contents of this field. Details on each are provided below.

[ddnn:[uic]]jobname

The job name alone is sufficient identification if the job is from the same UIC you are logged in to and if you have not specified a device in your Print command. Although the UIC is optional, the brackets around the UIC must be included if you include the UIC in the command line.

/JOB:n1:n2

Note that the display format and the command format for this entry number are different. Display: (n1,n2). Command: /JOB:n1:n2.

Switch

/DELETE

Deletes the specified queue entry. If the job is active at the time the command is issued, the output ceases and the queue entry is deleted.

Example

```
>QUE BADJOB /DELETE
```

The job named BADJOB from the default device and UIC is deleted from the default queue PRINT.

6.5 PRINT JOBS QUEUED BY USER TASKS - OUTPUT SPOOLING

There are two means of spooling output from user-written tasks. The first is from within the task:

- Open a file on disk and use PRINT$ macro or .PRINT subroutine using FCS.

Because the line printer (or other output device) is attached and owned by its despooling task (print processor), you cannot attach and write logical blocks to the device. The Write Logical Blocks operation fails and an Attach operation does not complete. Thus, you cannot in your user program issue an ALUN$ directive and expect QIOs to that LUN to work. If you wish to use this procedure, you must set the device unspooled by deleting the processor. An unspooled device is not accessible by the Queue Manager.

You cannot control the despooling of jobs queued with the PRINT$ macro or .PRINT subroutine directly, as you can through the Print command. Once the jobs appear in a queue, however, you can alter their attributes through the QUE /MOD command.

The alternate method of spooling output is from outside the task:

- Open a file on disk and use the Print command on task exit.

This is the only means of spooling output that gives you full control over how, when and where your output is despooled.

6.5.1 Opening A File On Disk and Using FCS

Your task should open a file on disk and output to it. The task can then issue the PRINT$ macro call or call the .PRINT subroutine to close the file and spool the output.

The file is placed in the default queue PRINT.

In FORTRAN, the equivalent is to use DISP= 'PRINT' in your OPEN or CLOSE statement.

You cannot control the despooling of such output directly, as you can through the Print command, but once the jobs appear in a queue, you can alter their attributes through the QUE /MOD commands.

See Chapter 8 of the IAS/RSX-11 I/O Operations Reference Manual for more information on PRINT$ and .PRINT.

6.5.2 Opening A File On Disk and Using the Print Command

Your task should open a file on disk, output to it, and close it. Once your task exits, despool the file with the Print command.

While this is not the simplest method of spooling, it is the only method that gives you access to the qualifiers on the Print command. You can accomplish much the same thing using the other method by waiting until the job is in the queue and then altering its attributes with the QUE /MOD command.

If you run your task from an indirect command file that includes a Print command after task exit, the difference between spooling from within the task or from outside it will probably not be noticeable.

6.6 ERROR MESSAGES

Error messages from the Queue Manager may be preceded by the letters PRI or QUE, depending on where within the software the error detection takes place. Here is how they appear on your terminal:

```
PRI -- NO SUCH FILE

QUE -- NO FUNCTION SPECIFIED
```

The PRI and QUE identifiers are not used in this section. When you receive an error message such as those above, look for the message portion (NO SUCH FILE or NO FUNCTION SPECIFIED in the examples above). The message portion of each error message appears in alphabetical order in this section along with a description of the error that produced the message and a suggested correction.

If you issue a QUE /LI (or /BR or /FU) and only a prompt is returned, this means no queue file exists on your system.

In most cases, the error will be a typing mistake or other syntax error. In a few cases, you will be directed to inform your system manager of the error.

BAD COMMAND LINE

Explanation: Invalid MCR line, or invalid syntax in Print or QUE command.

User Action: Retype command line.

BAD FILE SPEC SWITCH VALUE

Explanation: Bad value for file switch (/CO).

User Action: Retype command line after checking switch value.

BAD JOB SWITCH VALUE

Explanation: Bad value for job switch (/AF, /FO, /LE, /PR).

User Action: Retype command line after checking switch value.

BAD MODIFY VALUE

Explanation: Bad Modify switch value.

User Action: Retype command line after checking switch value.

BAD SWITCH TYPE

Explanation: Bad ASCII switch value (QMG, EOF, etc.).

User Action: Retype command line after checking switch type.

BAD SWITCH VALUE

Explanation: Bad numeric switch value.

User Action: Retype command line after checking switch value.

BAD SWITCH FOR FUNCTION

Explanation: The switch specified is not allowed for this function.

User Action: Retype command line after checking switch type.

DEVICE/QUEUE DOES NOT EXIST

Explanation: Command directed to device or queue not controlled by QMG.

User Action: QUE /LI:ALLDEV lists all devices. QUE /LI:ALLQUE lists all queues.

DIRECTORY NOT ALLOWED

Explanation: UIC not allowed on this command.

User Action: Retype command line without UIC.

ILLEGAL ARGUMENT VALUE

Explanation: The argument value is not valid.

User Action: Retype command after checking syntax.

QMG ERROR (n)

Explanation: Command in error or QMG task in error.

User Action: Look up n on list. If QMG task is in error, n will help pinpoint error. See your system manager. Retype command.

n	Code	Meaning
63.	QE.BSN	BAD PROCESSOR NAME
68.	QE.QMD	QUEUE MARKED FOR DELETE
69.	QE.SMD	PROCESSOR MARKED FOR DELETE
71.	QE.SDF	PROCESSOR DIRECTORY FULL
72.	QE.TYP	QUEUE AND PROCESSOR NOT SAME TYPE
73.	QE.JEA	JOB EXISTS ALREADY
84.	QE.RED	SPOOLED DEVICE REDIRECTED
91.	QE.IDB	INTERMEDIATE DEVICE BUSY - TRY AGAIN
92.	QE.BQN	BAD QUEUE NAME
100.	QE.NVT	VIRTUAL TERMINALS NOT SUPPORTED
104.	QE.IID	ILLEGAL OR NON-EXISTENT INTERMEDIATE DEVICE
105.	QE.INM	INTERMEDIATE DEVICE NOT MOUNTED

INVALID FILE SPEC SYNTAX

Explanation: Invalid syntax in a filespec.

User Action: Retype command line after correcting filespec.

INVALID ENTRY NUMBER

Explanation: Invalid syntax for entry number.

User Action: Retype command after correcting entry number. The display format and the command format for entry number are different. Display: (n1,n2). Command: /JOB:n1:n2.

INVALID JOB SYNTAX
Preceded by PRI --

Explanation:

- Invalid syntax in the job field
- Wildcards in job name
- Multiple job fields
- UIC present
- Bad terminator for job name (= or / are the only legal terminators)

User Action: Retype command line after checking proper syntax.

Preceded by QUE --

Explanation:

- Invalid syntax in the command
- Invalid job name
- Both job name and entry number specified
- Neither job name nor entry number specified

User Action: Retype command line after checking proper syntax.

I/O ERROR ON SY:QUEUE.SYS

Explanation: Error encountered reading the queue file.

User Action: See your system manager.

JOB DOES NOT EXIST

Explanation: Specified job does not exist in queue under the current UIC and the status (example, HELD) that you supplied in command.

User Action: Retype command after checking job status and job identification with QUE /LI.

JOB NAME NOT ALLOWED

Explanation: Job name or other form of job identification not allowed on this command.

User Action: Retype command line after checking proper syntax.

MULTIPLE FUNCTIONS DETECTED

Explanation: More than one function specified.

User Action: Retype command line after checking proper syntax.

MUTUALLY EXCLUSIVE SWITCH/VALUE

Explanation: Function value and switch mutually exclusive.

User Action: Retype command line after checking proper syntax.

NO FILE NAME GIVEN

Explanation: No file name given in a file spec.

User Action: Retype command including file name.

NO JOB FIELD GIVEN

Explanation: The command syntax requires a job field.

User Action: Retype command line after checking proper syntax.

NO FUNCTION SPECIFIED

Explanation: No function specified in command line.

User Action: Retype command line after checking proper syntax.

NO POOL SPACE

Explanation: Dynamic storage region has been exhausted; the system is either heavily loaded or there is a system problem.

User Action: See your system manager.

NO SWITCH(ES) ALLOWED

Explanation: The command syntax does not allow switches.

User Action: Retype command line after checking proper syntax.

NO SUCH COMMAND

Explanation: QMGCLI installed and run under wrong name.

User Action: See your system manager.

NO SUCH FILE filespec

Explanation: Print command named nonexistent file. Error message displays filespec given.

User Action: Retype command line after correcting filespec. Default file type for Print command is .LST.

NO SWITCHES ALLOWED

Explanation: Switches not allowed with this command.

User Action: Retype command line after checking proper syntax.

PRIVILEGED COMMAND

Explanation: Command reserved for privileged users: /START, /STOP, /SPOOL, /PUBLIC, /NOSPOOL, /NOPUBLIC, /MODIFY, /HOLD, /RELEASE, /DELETE on another UIC.

User Action: See your system manager.

QUEUE DIRECTORY FULL

Explanation: All devices and queue slots full in queue.

User Action: See your system manager.

QUEUE FULL

Explanation: The queue file is saturated; your job cannot be queued.

User Action: Enter the command again later, check the queue with QUE /LI, and see your system manager.

QUEUE/DEVICE NOT ALLOWED

Explanation: Command syntax does not allow a queue or device to be named.

User Action: Retype command after checking proper syntax.

QUEUE/DEVICE NOT SPECIFIED

Explanation: Command syntax requires a queue or device to be named.

User Action: Retype command after checking proper syntax.

QMG MARKED FOR EXIT

Explanation: Attempt to spool after /STOP:QMG issued.

User Action: Wait or see your system manager.

REQUEST FAILURE ON PROCESSOR TASK

Explanation: Device despool task failed to start.

User Action: See your system manager.

UIC NOT ALLOWED

Explanation: The command syntax does not include a UIC.

User Action: Retype after checking proper syntax.

CHAPTER 7

THE QUEUE MANAGER

This chapter is directed to the system manager or operator who is setting up and running the Queue Manager on an RSX-11M system. (For RSX-11M-PLUS systems, see the RSX-11M-PLUS Batch and Queue Operations Manual.) Except where noted, all commands in this chapter are privileged.

Most systems include the Queue Manager for the convenience of multistream line printer spooling. If not, the serial despooler, PRT..., can be installed instead. Note that the Queue Manager includes QMGPRT, which installs as PRT..., to intercept output from the PIP /SP switch and other utilities and system tasks.

The RSX-11M V3.2 Queue Manager consists of three components:

1. A command line interpreter (CLI) which processes both print requests and queue access commands as described in Chapter 6 of this manual. CLI communicates with the Queue Manager by sending data packets to it describing the print job.

2. The Queue Manager (QMG), which controls the queueing and dequeueing of print jobs. QMG communicates with CLI and the despooler tasks to process print jobs.

3. The despool prototype task (LPP). This task attaches a device and despools print jobs to it. A task must be installed for each device to be used as a despool device with the task name ddPn, where dd is the generic device name and n is the unit number. A despool device is a device under the control of QMG.

"Spool" is an acronym of Shared Peripheral Operations On-Line. "Spooling" on RSX-11M is gathering output sent to the line printer on a mass storage device - usually a disk - to be passed in an orderly fashion to the line printer. "Despooling" is the orderly transfer from the mass storage device to the line printer.

CLI takes a Print command and interprets the print request it specifies. The information is sent to QMG which creates a print job consisting of the names of the files to be printed and the attributes of the job. The print job entry is placed in a print queue kept in the disk file SY0:[1,7]QUEUE.SYS. Queues are assigned to despool devices which are controlled by despool tasks. When a despool device is idle the Queue Manager selects the next job in the queue(s) assigned to its device and processes it. Finally the queue entry is deleted.

This chapter consists of a narrative introduction and review of the process of setting up the Queue Manager, detailed descriptions of all commands to the Queue Manager, and a listing of an indirect command file to be included in the system STARTUP procedure as a reference example.

7.1 NARRATIVE INTRODUCTION AND REVIEW

Here is the procedure for bringing the Queue Manager into the system.

This can be done interactively, but in most cases it is more convenient to include it in an indirect command file invoked as part of the system's STARTUP procedures.

Note that this procedure must be adapted for each installation. The example is based on the procedure used for the RSX-11M development system. QUE setup commands allow considerable flexibility in setting up the Queue Manager system. Dynamic alteration of the Queue Manager is also permitted.

1. Install the Queue Manager. It is found with the privileged utility tasks.

```
>INS DB0:QMG /PRI=70
```

2. Install the output despooler task.

```
>INS DS0:LPP /PRI=70
```

   This task installs automatically as LPP0. You must install this task for each physical device under the direction of the Queue Manager. This task "owns" the output device and must have a name based on the name of the device. Thus, the output despooler for LP1: must be installed under the name LPP1, using the /TASK= option. There can be up to 15 output devices controlled by the Queue Manager.

3. Install PRT...

```
>INS QMGPRT
```

   This task installs automatically as PRT... and is the interface to PIP and other tasks that use PRINT$ or .PRINT. The Queue Manager can then accept jobs from the PIP /SP switch. (Although this task has the same name as the Serial Despooler used on previous versions of RSX-11M (and available on RSX-11M V3.2), it is not the same task. It replaces this task.)

4. Install the Print Command Line Interpreter and the Queue Command Line Interpreter.

```
>INS QMGCLI /TASK=...PRI

>INS QMGCLI /TASK=...QUE
```

   The same task parses both the PRI and QUE commands and therefore must be installed twice, once under each name.

5. Start the Queue Manager.

```
>QUE /START:QMG
```

This command starts the Queue Manager and initializes the default queue PRINT. It creates the queue file QUEUE.SYS on SY:[1,7] if it is not there. The command also clears all assignments previously recorded. If the queue file was intact, all queues will still contain their jobs, but are not assigned to processors. (Processors will have to be spooled again.)

6. Spool the processor.

```
>OUE LP: /SPOOL /FLAG:1 /FORM:0
```

This command creates a queue LPQ0 and starts the previously installed despool task LPP0. This task attaches device LP0: for despooling. Queue LPQ0 is assigned to processor LPP0. Repeat this step for each output device under control of the Queue Manager.

The /FORM switch establishes this attribute of the despooler. This means the despooler accepts jobs initiated by Print commands with the same FORM attribute given. Jobs with other FORM attributes remain in the queue until a despooler with the proper FORM attribute is available.

The /FLAG switch sets the number of flag pages that precede each job. If users specify /FLag in their Print commands, each file in the job will be preceded by this many file flag pages as well.

7. Assign queues to processors.

```
>QUE LP: /ASSIGN:PRINT
```

This command assigns the default queue PRINT to processor LPP0. Normally, only the PRINT queue has to be assigned. Device-specific queues are assigned to their device despoolers by /SPOOL. This assignment of device-specific queues is a default, however, and not a requirement. A device-specific queue must exist for every spooled device, but these queues can be assigned to other devices.

You can assign a queue to more than one processor and you can assign more than one queue to a processor. You can assign a device-specific queue to another device if necessary. For example, if printer LP0: is temporarily unavailable, you could assign LPQ0 to LP1:.

7.2 REFERENCE EXAMPLE

Here is a listing of SY:[1,2]QMGSTART.CMD from the RSX-11M development system:

```
;
;
INS DB0:QMG /PRI=70
INS DS0:LPP /PRI=70
INS DS0:QMGPRT
INS DS0:QMGCLI /TASK=...PRI
INS DS0:QMGCLI /TASK=...QUE
QUE  /START:QMG
QUE LP: /SPOOL /FLAG:1 /FORM:0
QUE LP: /ASSIGN:PRINT
;
;
```

7.3 QUEUE MANAGER COMMAND DESCRIPTIONS

The configuration of the Queue Manager can be controlled through a number of special functions applied to the QUE command.

The format of the Queue Manager setup commands is:

```
>QUE ddnn:  /function[:option[s]][/sw[s]]
```

ddnn:

Specifies the name of the device affected by the command.

/function

Specifies the function performed by the command.

:option[s]

Specifies one or more options for a function.

/sw[s]

Specifies one or more switches for a function or option.

These commands allow a privileged user to attach or detach devices for spooling, start and stop devices, assign the default queue to devices and start and stop the Queue Manager.

7.3.1 STOP

This function suspends a device or stops the Queue Manager.

Format

```
>QUE ddnn: /STOP[:option]
```

Options

```
:EOF
:EOJ

>QUE /STOP:QMG
```

ddnn:

The device name is required with all options except /STO:QMG. You cannot include a device when stopping the Queue Manager.

Options

If you do not enter an option (null option) the device stops immediately. This is a pause.

:EOF

The device stops at the end of the current file.

:EOJ

The device stops at the end of the current job.

/STO:QMG

The Queue Manager stops at the end of the current job. All assignments of queues to devices are broken (deassigned). All despooler tasks are detached from their devices. The Queue Manager stops.

Examples

```
>QUE LP0: /STOP
```

This example stops printing on LP0:. No jobs will be taken from queues assigned to the device, but jobs can still be added to queues.

```
>QUE LP2: /STO:EOJ
```

This example stops printing on LP2: at the end of the active job.

```
>QUE /STOP:QMG
```

This example stops the Queue Manager. QMG is marked for stop and the current job is allowed to finish.

7.3.2 START

This command starts printing on a previously stopped device or starts the Queue Manager after a STOP. This command counteracts STOP.

Format

```
>QUE ddnn: /STArt[:option] [/sw]
```

Options	Switches
:NExt	/FOrm:n
:TOP	/FLag:n
:PAge:n	
:BAck:n	
:FOrward:n	

```
>QUE /STArt:QMG
```

ddnn:

A device name is required with all options except /STA:QMG. You cannot include a device when starting the Queue Manager.

Options

Option and switch are mutually exclusive fields. Use one or the other.

If you enter nothing (null option), the device starts printing from the point at which printing stopped. This, in effect, resumes printing after a pause.

:NExt

Printing starts at the top of the next job.

:TOP

Printing starts at the top of the current file.

:PAge:n

Printing starts at page n of the current file.

:BAck:n

Printing starts n pages back from the point at which it was stopped.

:FOrward:n

Printing starts n pages forward from the point at which it was stopped.

/STA:QMG

This option is entered without ddnn: as it starts Queue Manager and initializes the default queue PRINT. It creates the file SY:[1,7]QUEUE.SYS if it is not there. The command also clears all assignments previously recorded. If the queue file was intact, all queues still contain their jobs, but are not assigned to processors. (Processors will have to be spooled again.)

Switches

The switch and option fields are mutually exclusive. Use one or the other. You can enter more than one switch, however.

/FOrm:n

This switch resets the forms type of the despooler task from the type set with the SPOOL function

/FLag:n

This switch resets the flag setting of the despooler task from the number of flag pages set with the SPOOL function. Print jobs have n job flag pages preceding the job. If users specify /FLA in their Print command, each file in the job will also be preceded by n file flag pages in addition to the job flag pages. n can be 0, 1, 2, or 3.

If you are using special forms in your printer, you should set n=0 so that no flag pages are printed on your special forms.

Examples

```
>QUE LP0: /STA
```

This example starts LP0:; printing proceeds from where it was stopped.

```
>QUE LP2: /START:NE
```

This example starts printing with the next job queued to LP2:.

```
>QUE LP: /STA /FO:11
```

This example starts printing on LP0: with the FORMS attribute reset to 11.

Notes

Page-related options have no effect unless the user specified a /LEngth:n quantity in the Print command.

If you are restarting a processor to handle special forms types, you may want to set /FL to 0 to avoid printing banner pages on your special forms.

7.3.3 SPOOL

This function attaches a device for despooling. The command names and starts a despool task and a queue named after it, and assigns the queue to the despool task.

Format

>QUE ddnn: /SPool [/sw]

Switches

/FOrm:n
/FLag:n
/EXternal

ddnn: /SPOOL

ddnn: /SPOOL creates queue ddQn and starts the despool task ddPn which must be installed previously for device ddn:. This attaches device ddnn: for despooling by the despool task ddPn open to jobs from queue ddQn.

Switches

More than one switch can be used with this function.

/FOrm:n

Sets the device despooler to accept jobs with forms type n.

You can set a despooler to accept more than one forms type, but the forms definition must be duplicated.

The number n can be anything from 0 to 255. The default is 0.

These numbers are to be specified by your users in their Print commands. If the right forms are not in an available printer, then their jobs will be waiting until the printer has been stopped, the proper forms put in place, and the printer restarted using the /FORMS: qualifier and the proper value for n in the START command.

Forms differ by their length and width. Both values can be set in the forms table.

Setting width is simply a matter of how many columns are needed to fill in the form. There are no standard widths for forms, but 80. and 132. are the most common.

Setting length requires you to determine whether your forms are of a standard length.

"Standard length" is any form length that can be set on your printing hardware. If the hardware can handle the form by setting a response to a form-feed character <FF>, then it is a standard length form, and the form feed is a "real," that is, hardware, form feed.

If the form length cannot be set on the hardware, then it is a non-standard length. In this case, the form feed <FF> must be replaced by an appropriate number of line feeds <LF>. This is called a "simulated," that is, software, form feed.

You should determine which forms are used on your system and assign each a number. This number points to a table of form definitions. The form definitions consist of a single 16-bit word. The length is the low-order byte and the width is the high-order byte. The high bit of the length byte is set when simulated forms (forms whose length cannot be set on your hardware) are needed. This form definition word can be changed by editing the Task Builder command file for the despooler task. You must add a global patch statement of the following form:

```
TKB>GBLPAT=INIT+FRMTBL+n*2:value
```

where n is the assigned form number and value is the form definition word.

Table 7-1 shows the forms requirements for a hypothetical installation. The form names describe different uses that special forms might be put to, but have no significance otherwise.

Table 7-1
Demonstration Forms Settings

Form	Assigned n	Width	Length	Standard?
Accordion	0	132	66	Y
Paychecks	1	40	10	Y
Invoices	2	80	66	Y
Stationery	3	80	66	Y
Moon Diary	4	22	43	N
Star Log	5	112	113	N

NOTE

As shipped, FORMS:0 is 66 lines long by 132 columns with real form feeds. FORMS:1 is 1 line long by 132 columns with simulated form feeds; FORMS:2 is 2 lines long by 132 columns with simulated form feeds; and so on through FORMS:66 which is 66 lines long by 132 columns with simulated form feeds. The remaining FORMS:n entries are set to 1 line long by 132 columns with simulated form feeds. Thus, if you specify FORMS:43, when the printer encounters a form feed, it counts the number of lines on the page and subtracts that number from 43. The printer then outputs a number of line feeds equal to the difference. You may find this arrangement satisfactory if your installation rarely uses forms other than standard line-printer paper, but if you have more than one standard form, you will probably want to make the changes in the forms table entries.

/FLag:n

Sets the number of flag pages to precede all jobs printed on the device. Note that if you are using special forms, you may want to set this attribute to 0 to avoid printing banner pages on your special forms. This also sets the number of file flag pages to be printed when users specify the /FLAGS switch in their Printer commands. The default value of n is 0.

/EXternal

Specifies that a device is being set spooled that is not currently part of the local system configuration. Use this switch when you wish to despool to an applications task, or a network, or in some other instance of using the Queue Manager for down-line loading.

Examples

```
>QUE LP0: /SPOOL
```

This example attaches LP0: for despooling.

```
>QUE LP3: /SP /FO:2
```

This example attaches LP3: for despooling with forms set to 2.

7.3.4 UNSPOOL

This function removes a device from despooling. The device is no longer assigned to the queue. The despooler task exits. The associated device-specific queue is deleted.

Format

```
>QUE ddnn: /UNSPool
```

ddnn: /UNSPool

ddnn:/ UNSPOOL frees a device used in despooling for general use. The despool task for the device exits but its entries in queues assigned to it remain in the queue file.

Example

```
>QUE LP0: /UNSP
```

This example removes the device LP0: from active use by the despooler LPP0 and the Queue Manager.

7.3.5 ASSIGN

This function creates a path between a queue and despooler.

Format

```
>QUE ddnn: /ASsign:queuename
```

ASSIGN is normally used to assign the default queue PRINT to one or more despooler tasks, but you can assign any queue to any despooler.

Example

```
>QUE LP0: /AS:PRINT
```

This example assigns the PRINT queue to LP0:.

Note

Remember that SPOOL automatically assigns a device-specific queue to its related device. For example, LPQ1 is automatically assigned to LP1:.

7.3.6 DEASSIGN

This function breaks the assignment of a queue to a despooler.

Format

```
>QUE ddnn: /DEAssign:queuename
```

DEASSIGN eliminates the path between a queue and a despooler, but the queue still exists with all its jobs. Jobs can be added to the queue, but none will be dequeued while the queue is not assigned.

7.4 ERROR MESSAGES

Error messages from the Queue Manager may be preceded by the letters PRI or QUE, depending on where within the software the error detection takes place. Here is how they appear on your terminal:

```
PRI -- NO SUCH COMMAND

QUE -- DIRECTORY NOT ALLOWED
```

The PRI and QUE identifiers are not used in this section. When you get an error message such as those above, look for the message portion (NO SUCH COMMAND or DIRECTORY NOT ALLOWED in the examples above). The message portion of each error message appears in alphabetical order in this section, along with a description of the error that produced the message and a suggested correction.

If you issue a QUE /LI (or /BR or /FU) and only a prompt is returned, this means no queue file exists on your system.

In most cases, the error will be a typing mistake or other syntax error.

BAD COMMAND LINE

Explanation: Invalid MCR line or invalid syntax.

User Action: Retype command line after checking proper syntax.

BAD SPOOL DEVICE TYPE

Explanation: Only record-oriented devices can be spooled.

User Action: Check device type.

BAD SWITCH TYPE

Explanation: Bad ASCII switch value.

User Action: Retype command line after checking proper switch value.

DEVICE ALREADY SPOOLED

Explanation: Specified device already spooled by a previous command.

User Action: Issue QUE /LI:ALLDEV to check list of spooled devices.

DIRECTIVE FAILURE

Description: Program error. Executive directive failed.

User Action: Detect and correct error.

DIRECTORY NOT ALLOWED

Explanation: UIC included in command that does not accept it.

User Action: Retype command line after checking proper syntax.

ILLEGAL ARGUMENT VALUE

Explanation: Bad switch value.

User Action: Retype command line after checking proper switch value.

INVALID JOB SYNTAX

Explanation:

- Invalid syntax in the command
- Invalid job name
- Included both job name and job identification
- Omitted both job name and job identification

User Action: Retype command line after checking proper syntax.

I/O ERROR ON SY0:QUEUE.SYS

Explanation: Error encountered in reading queue file.

User Action: Check SY0:[1,7] for presence of QUEUE.SYS.

JOB DOES NOT EXIST

Explanation: Specified job not found in queue under current UIC.

User Action: Retype command line, including correct UIC in job identification.

JOB NAME NOT ALLOWED

Explanation: Job name or other form of job identification not accepted in this command.

User Action: Retype command line after checking proper syntax.

MULTIPLE FUNCTIONS DETECTED

Explanation: More than one function specified.

User Action: Retype command line after checking proper syntax.

MUTUALLY EXCLUSIVE SWITCH/VALUE

Explanation: Command line specified mutually exclusive function value and switch.

User Action: Retype command line after checking proper syntax.

NO FUNCTION SPECIFIED

Explanation: Command line did not include required function.

User Action: Retype command line after checking proper syntax.

NO SUCH COMMAND

Explanation: QMGCLI installed and run under wrong name.

User Action: Install and run QMGCLI under proper name: ...PRI, or ...QUE.

NO SWITCHES ALLOWED

Explanation: No switches allowed with this command.

User Action: Retype command line after checking proper syntax.

PRIVILEGED COMMAND

Explanation: The following functions are privileged: /START, /STOP, /SPOOL, /UNSPOOL, /ASSIGN, /DEASSIGN, /MODIFY, /HOLD, RELEASE, /DELETE jobs from another UIC.

User Action: Enter command from a privileged terminal.

QUEUE/DEVICE NOT ALLOWED

Explanation: /STOP:QMG or START:QMG command issued with device name.

User Action: Retype command without device name.

QUEUE/DEVICE NOT SPECIFIED

Explanation: /START, /STOP, /UNSPOOL, /ASSIGN, and /DEASSIGN require device name.

User Action: Retype command with device name.

QMG DID NOT START

Explanation: /START:QMG did not work.

User Action:

- Check to see if QMG is installed under name QMG...
- Check to see if pool is low.

PART 5
VOLUME MAINTENANCE UTILITIES

CHAPTER 8

DISK VOLUME FORMATTER (FMT)

The Disk Volume Formatter (FMT) utility formats and verifies DB:, DK:, DM:, DP:, DR: and DY: disk volumes under any RSX-11M, RSX-11M-PLUS, or IAS operating system which includes diagnostic support in the Executive. (Check with your system manager to determine whether your system includes this feature.)

The disks can be completely formatted (in normal operating mode) or formatted on an individual sector (or track for DM: volumes) basis (in manual operating mode).

Formatting in manual operating mode sometimes allows you to recover use of a bad sector or track.

In general, FMT performs the following tasks:

- Writes a complete header for each sector of the disk it is formatting.
- Verifies the address contents of each sector header.
- Sets the density for DY: floppy diskettes.
- Lets you specify a maximum pack error limit for the disk volume being formatted. FMT terminates processing when the error limit is reached.
- Allows the Bad Block Locator (BAD) task to be run (spawned) if your system permits spawned tasks.

8.1 INITIATING AND TERMINATING FMT

To initiate FMT, enter the appropriate command following the system monitor prompt, as explained in Chapter 1 of this manual.

The general form of the FMT command line is:

```
>FMT ddn:/[switch 1...switch m]
```

where dd: is the abbreviation for the volume you are formatting, n is the unit number of the volume, and the possible switches are:

/BAD Runs the Bad Block Locator task (BAD) if it is installed on the system.

Note that this switch can only be used with operating systems that allow spawning of tasks. RSX-11M and RSX-11M-PLUS provide spawned tasks as a System Generation option.

/DENS Selects high (double) or low (single) density for RX02 floppy diskettes.

/ERL Determines the maximum number of errors FMT will allow on the volume.

/MAN Enters manual operating mode and formats the sector or track you specify.

/OVR Overrides or ignores the Manufacturer's Defined Bad Sector File (MDBSF).

/WLT Rewrites the MDBSF to add bad sectors found during FMT operation.

/VE Verifies that a FMT operation was successfully completed.

/@Y Informs FMT that it is executing from an indirect command file. An FMT command in this form does not allow operator intervention in the process.

These switches are described in detail in Section 8.4.

8.2 MODES OF FMT OPERATION

FMT allows you to format volumes in two ways: normal operating mode, which formats an entire volume, and manual operating mode, which formats the sector (or track on DM: volumes) you specify in response to FMT prompts. FMT uses normal operating mode unless you specify manual mode with a /MAN in the command line.

FMT normally retries an operation twice when it encounters an error. If the operation still fails, FMT flags the sector as bad and displays the following message:

```
ERROR WRITING HEADER
```

If you specify the Verify switch with an FMT operation, and FMT encounters an error, it prints one of the following messages on your terminal:

```
ERROR READING HEADER
      or
HEADER COMPARE ERROR
```

FMT then continues the verification operation.

8.2.1 Normal Operating Mode

When you invoke FMT in normal operating mode (without the Manual switch), it prints the following message:

```
** WARNING - DATA WILL BE LOST ON ddn:  **
CONTINUE [Y OR N]
```

After a Y (yes) response, FMT returns the message:

```
START FORMATTING
```

It then performs the formatting functions you specify with switches in the FMT command. After an N (no) response or a carriage return <CR>, FMT returns control to the system monitor.

Normal FMT operation varies slightly according to the volume you are formatting (see Section 8.3).

8.2.2 Manual Operating Mode

If you specify manual operating mode (/MAN), FMT prints:

```
** WARNING - DATA WILL BE LOST ON ddn:  **
CONTINUE [Y OR N]
```

After a Y (yes) response, FMT returns the message:

```
ENTERING MANUAL MODE
```

It then displays the following prompts:

```
CYLINDER=
TRACK    =
SECTOR   =
```

After you enter your response to the prompts, FMT formats the sector or track you specify. FMT assumes the responses are in decimal unless they are preceded by a number sign (#). If you enter a parameter that is out of range of the volume FMT returns an error message and exits. Table 8-1 lists the valid ranges for FMT manual mode operations.

FMT manual operating mode works the same on all disk volumes, with one exception: On DM: volumes (RK06 and RK07) FMT formats a specific track of the volume.

For example:

```
FMT>DM0:/MAN
```

This command causes FMT to prompt:

```
** WARNING - DATA WILL BE LOST ON DM0:  **
CONTINUE [Y OR N] Y

ENTERING MANUAL MODE
CYLINDER=
TRACK    =
```

FMT then formats the entire track you specified.

Table 8-1
Valid Ranges for Manual FMT Operations

Disk Volume[1]	Sectors	Tracks	Cylinders
RP02/RPR02	0-9	0-19	0-199
RP03	0-9	0-19	0-399
RP04	0-21	0-18	0-410
RP05	0-21	0-18	0-410
RP06	0-21	0-18	0-814
RK05/RK05F	0-11	0-1	0-199
RK06	0-21	0-2	0-410
RK07	0-21	0-2	0-814
RM02	0-31	0-4	0-822
RM03	0-31	0-4	0-822

1 Note that FMT manual operating mode cannot be used with RX02 floppy diskettes.

8.3 FMT-SUPPORTED DEVICES

The following sections describe normal FMT operating mode. Table 8-2 lists the devices that allow formatting and their abbreviations.

Table 8-2
FMT-Supported Devices

Device	Abbreviation
RP04 disk pack	DB:
RP05 disk pack	DB:
RP06 disk pack	DB:
RK05 disk cartridge	DK:
RK05F fixed media disk	DK:
RK06 disk cartridge	DM:
RK07 disk cartridge	DM:
RPR02 disk pack	DP:
RP02 disk pack	DP:
RP03 disk pack	DP:
RM02 disk pack	DR:
RM03 disk pack	DR:
RX02 floppy diskette	DY:

The status FMT requires for the devices varies with the operating system. Table 8-3 lists the status required for devices with RSX-11M, RSX-11M-Plus, or IAS.

Table 8-3
FMT Device Status

Operating System	Device Status Required for FMT Operation
RSX-11M-PLUS	Mounted with Foreign Characteristics
RSX-11M V3.2	Unmounted
IAS	Mounted with Foreign Characteristics

8.3.1 DB: Devices (RP04/RP05/RP06 Disk Packs)

When FMT formats a DB: volume, it tries to write 22 headers at a time until it has formatted the entire volume. If FMT encounters an error, it attempts to write each header individually and designates which headers are bad.

If you specify the Verify switch (/VE), FMT verifies 11 headers at a time until it has verified the entire volume. If FMT encounters an error, it attempts to verify the headers individually to determine where the error occurred. It then reports any bad headers and continues the operation.

8.3.2 DK: Devices (RK05 Disk Cartridge or RK05F Fixed Media Disks)

When FMT formats a DK: volume, it tries to write each header individually until it has formatted the entire volume. If FMT encounters an error, it retries each header twice before reporting the header as bad.

If you specify the Verify switch (/VE), FMT verifies the headers 12 at a time, until it has verified the entire volume. If FMT encounters an error, it attempts to verify the headers individually to determine where the error occurred. It then reports any bad headers and continues the operation.

8.3.3 DM: Devices (RK06/RK07 Disk Cartridges)

FMT writes DM: headers one track (22 sectors) at a time and sets the header flags of those sectors marked bad in the MDBSF. If FMT encounters errors, it retries the operation twice before it designates which headers are bad.

If you specify the Verify switch (/VE), FMT verifies that each sector from 0 to 21 is addressable. It does this by issuing a full 256-word write, made up of the 2-word address pattern (the sector number and its complement) into each sector. Once the entire track has been written, each sector is read and the full 256 words of data are

compared with the expected data pattern. If an error occurs during this operation, FMT reports that sector as bad and continues the operation.

When FMT writes headers on DM: devices, it sets bad sector flags in the headers already marked as bad in the MDBSF. If you also specify the Verify switch, FMT indicates whether the bad sector was flagged in the MDBSF.

8.3.4 DP: Devices (RPR02/RP02/RP03 Disk Packs)

When FMT formats a DP: volume, it tries to write 10 headers at a time until it has formatted the entire volume. If FMT encounters an error, it attempts to write each header individually and designates which headers are bad.

If you specify the Verify switch (/VE), FMT verifies the headers 10 at a time, until it has verified the entire volume. If FMT encounters an error, it attempts to verify the headers individually to determine where the error occurred. FMT reports that sector as bad and continues the operation.

8.3.5 DR: Devices (RM02/RM03 Disk Packs)

When FMT formats a DR: volume, it tries to write 32 headers at a time until it has formatted the entire volume. If FMT encounters an error, it attempts to write each header individually and designates which headers are bad.

If you specify the Verify switch (/VE), FMT verifies the headers 16 at a time, until it has verified the entire volume. If FMT encounters an error, it attempts to verify the headers individually to determine where the error occurred. It then reports any bad sectors and continues the operation.

When FMT writes headers on DR: volumes, it sets bad sector flags in headers already designated as bad by the MDBSF. If the command also specifies the Verify switch, FMT indicates whether the sector was marked in the MDBSF.

8.3.6 DY: Devices (RX02 Floppy Diskettes)

You can use FMT to set an RX02 floppy diskette to either high (double) or low (single) density. If you specify the Verify switch, FMT writes and reads block 0 and the last block on the diskette to determine that the density is correct.

Note that manual operating mode cannot be used with DY: devices.

8.4 FMT SWITCHES

The following sections describe the switches you can use with FMT commands. The descriptions include information on restrictions for formatting specific devices and default values for the switches, where appropriate.

8.4.1 /BAD

The Bad switch spawns the Bad Block Locator task (BAD) after FMT completes an operation. If BAD is not installed on the system, FMT prints a warning message on your terminal and exits.

The format for an FMT command, using the BAD switch, is:

```
FMT>dev:/BAD
```

8.4.2 /DENS

The Density switch sets DY: floppy diskettes to either high or low density. The default is low density. (This switch can also use single and double density as options.)

The formats for an FMT command, using the DENS switch, is:

```
FMT>DYn:/DENS=HIGH (or DOUBLE)
FMT>DYn:/DENS=LOW (or SINGLE)
```

8.4.3 /ERL

The Error Limit Switch sets an error limit for the volume you are formatting. If the error count reaches this limit, FMT generates an appropriate message and terminates operation. The default error limit is 256 errors. Any value greater than 0 or less than or equal to 256 is valid.

The format for an FMT command, using the ERL switch, is:

```
FMT>dev:/ERL=n
```

8.4.4 /MAN

The Manual switch puts FMT in manual operating mode and permits you to format an individual sector (or track for DM: disk cartridges). FMT assumes cylinder, track, and sector numbers are decimal values unless they are preceded with a number sign (#).

In manual operating mode, FMT displays the following prompts:

```
** WARNING - DATA WILL BE LOST ON DK1:  **

CONTINUE [Y OR N]?
ENTERING MANUAL MODE
CYLINDER=
TRACK    =
SECTOR   =

OPERATION COMPLETE
```

The format for an FMT command, using the MAN switch, is:

```
FMT>dev:/MAN
```

8.4.5 /OVR

The override switch causes FMT to ignore the Manufacturer's Detected Bad Sector File (MDBSF) on DM: and DR: disk volumes. When FMT writes headers on these disks, it normally sets bad sector flags in those headers marked bad in the MDBSF. When the verification process discovers a bad sector, it reports that the sector was marked in the MDBSF. The Override switch inhibits this operation.

The format for an FMT command, using the OVR switch, is:

```
FMT>dev:/OVR
```

8.4.6 /VE

The Verify switch confirms that an FMT operation was successful. It does this by reading back the headers and determining that they were written correctly.

The format for an FMT command, using the VE switch, is:

```
FMT>dev:/VE
```

8.4.7 /WLT

The Write Last Track switch, when used with the Verify switch on DM: and DR: volumes, rewrites the MDBSF after an FMT operation to add the bad sectors that FMT found to the bad sectors already in the MDBSF. FMT also rewrites each bad sector's header to flag it as a bad sector.

The WLT switch requires a decimal number (n below) which is used as the volume's pack serial number.

The format for an FMT command, using the WLT switch, is:

```
FMT>dev:/WLT:n
```

8.4.8 /@Y

If you specify the @Y switch, FMT executes from an indirect command file. In this method of operation, FMT will not generate any operational messages or warnings to your terminal until the operation is complete and no operator intervention is possible.

The format for an FMT command, using the @Y switch, is:

```
FMT>dev:/@Y
```

8.5 FMT MESSAGES

This section describes the messages FMT generates, along with possible user action in response to the messages.

COMMAND I/O ERROR

Explanation: A hardware transmission error occurred from the keyboard.

User Action: Reenter the command.

COMMAND TOO LONG

Explanation: The command was longer than 80 (10) characters.

User Action: Enter a shorter command.

DEVICE DOES NOT SUPPORT FORMATTING

Explanation: A device was specified that does not allow the use of FMT.

User Action: Determine the correct device and, if FMT operation is legal, reenter the command.

DEVICE DRIVER MISSING

Explanation: The disk device driver is not loaded.

User Action: Load the driver (if it is loadable) and reenter the command, or use a different device in the command line.

DEVICE NOT IN SYSTEM

Explanation: The specified device was not identified as part of the system during system generation, or the device does not exist on the system configuration.

User Action: Determine the correct command line with the correct device abbreviation, and reenter the command.

DEVICE NOT READY

Explanation: The disk volume was not at operating speed when FMT attempted to access it.

User Action: Allow the volume to reach operating speed and reenter the FMT command.

DEVICE OFFLINE

Explanation: The device is not in the hardware configuration.

User Action: Determine the correct command line with the correct device abbreviation and reenter the command.

DEVICE WRITE LOCKED

Explanation: The volume is write-locked; any write access is prohibited.

User Action: Write-enable the unit and reenter the FMT command.

DISK IS AN ALIGNMENT CARTRIDGE

Explanation: The device is a factory-created disk used to align the heads in a disk drive and should not be used for other purposes.

User Action: Use a disk that is not an alignment cartridge and reenter the FMT command.

ERROR LIMIT EXCEEDED

Explanation: The number of errors FMT found on the disk pack exceeded either the number of errors specified with the ERL switch, or the default 256 error limit that FMT sets.

User Action: Set a higher error limit, if the ERL switch was used.

ERROR READING DATA

Explanation: FMT encountered an error in reading data from a disk.

User Action: None required; FMT retries the operation twice and continues the verification.

ERROR READING HEADER

Explanation: FMT encountered an error when it tried to read a header during a verification operation.

User Action: None required; FMT retries the operation twice and continues the verification.

ERROR SETTING DISKETTE DENSITY

Explanation: FMT tried to format a DY: floppy diskette and the operation failed.

User Action: Check the syntax and reenter the command, resetting the density.

ERROR WRITING DATA

Explanation: FMT encountered an error when it attempted to write sector headers.

User Action: None required; FMT retries the operation twice and continues the verification.

ERROR WRITING HEADER

Explanation: FMT encountered an error when it tried to write a header.

User Action: None required; FMT retries the operation twice.

FAILED TO ATTACH DEVICE

Explanation: FMT could not attach the device to be formatted.

User Action: Determine whether another task has attached the device. If so, wait until the task exits, or abort the task and run FMT again.

FAILED TO READ MANUFACTURER'S BAD SECTOR FILE

Explanation: A disk hardware error occurred while FMT attempted to read the MDBSF on the last track on the last track of a device.

User Action: Reenter the command, including the Override switch.

FATAL HARDWARE ERROR

Explanation: A fatal error occurred somewhere in the system hardware configuration.

User Action: Contact the DIGITAL Field Service representative.

HEADER COMPARE ERROR

Explanation: FMT found an error in comparing headers with the expected value during a verification error.

User Action: None required; FMT retries the operation twice.

INVALID SWITCH

Explanation: An illegal switch or a switch not valid for the specified device was used in an FMT command.

User Action: Check the syntax and reenter the command.

MANUFACTURER'S BAD SECTOR FILE CORRUPT

Explanation: The factory-written bad block data (MDBSF) on the last track of the disk is in an unusable format.

User Action: Reenter command with the Override switch (/OVR) to prevent FMT from trying to use the corrupt bad block data.

MARKED BAD IN MANUFACTURER'S BAD SECTOR FILE

Explanation: Indicates that bad block information is recorded in the MDBSF on the disk.

User Action: None required. This message is for information only.

PRIVILEGE VIOLATION

Explanation: FMT attempted an operation on a device that was mounted or allocated to another user.

User Action: Reenter the FMT command, using a device that is not allocated to another user.

RESPONSE OUT OF RANGE

Explanation: Parameters entered for manual formatting of an individual sector or track were out of the range of the volume.

User Action: Check Table 8-1 for legal parameters and reenter the command.

SYNTAX ERROR

Explanation: FMT detected a syntax error in the command line.

User Action: Determine the correct command syntax and reenter the command.

UNABLE TO RUN BADBLOCK UTILITY

Explanation: A FMT command specified the Bad switch (/BAD), but BAD could not be spawned. Either the operating system does not spawn tasks or BAD was not installed.

User Action: Run the BAD utility separately.

UNRECOVERABLE ERROR - n

Explanation: An I/O error (number n) caused FMT to terminate.

User Action: Reenter the FMT command and if the error occurs again, try the command, specifying a different device.

CHAPTER 9

BAD BLOCK LOCATOR UTILITY (BAD)

9.1 INTRODUCTION TO BAD

The Bad Block Locator Utility (BAD) tests disks and DECtapes for the location and number of bad blocks. BAD then records this bad-block information on the device. When you use the MCR Initialize Volume command (INI) the bad blocks are marked as in-use; that is, INI allocates the bad blocks to the file [0,0]BADBLK.SYS. Thus, the bad blocks cannot be allocated to other files. BAD supports any last track device, as well as vendor-supplied cartridges that do not have a prerecorded manufacturer's bad-sector file on the last track. You can use BAD in its task version, which runs at the same time as other tasks, or in its stand-alone version, which runs by itself on the computer. The stand-alone version is preferable if you have a system with a single disk drive.

9.2 INVOKING BAD

You can invoke the BAD utility as follows:

```
>BAD dev:[/sw]...
```

dev:

Specifies a physical device. The specification consists of two alphanumeric characters followed by a 2- or 3-digit octal unit number and colon.

/sw

Specifies an optional switch that qualifies the BAD command line. Multiple BAD switches for a device must be specified on one line. If you do not specify any switch, BAD begins its pattern checking of individual blocks.

9.2.1 BAD Switches

Table 9-1 contains a reference list of BAD switches along with a brief description of each. For a detailed description of BAD switches see Section 9.5.

Table 9-1
BAD Switches

Switch	Function
BAD switches for Task and Stand-Alone Versions	
/LI	Lists bad blocks as they are located
/MAN	Prompts you for additional bad blocks
/OVR	Creates bad block descriptor file on a last-track device
/RETRY	Recovers soft errors
/UPDATE	Reads the bad block descriptor file and prompts for input
BAD Switches for Stand-Alone Version Only	
/CSR=nnnnnn	Alters the CSR address of the device
/VEC=nnn	Alters interrupt vector address of the device
/WCHK	Causes a write check
/NOCHK	Negates /WCHK

9.2.2 BAD And Indirect Files

BAD can access an indirect file that contain a series of BAD command strings in the following manner:

```
>BAD @BADCMDS.CMD
```

In this example, BAD is invoked and accesses the file BADCMDS.CMD, which contains a sequence of BAD command strings. BAD executes the commands and returns control to MCR. The BAD utility allows nested command files; one command file can invoke another to a maximum depth of three.

BAD can also be invoked by an indirect file. Such an indirect file can contain command strings for more than one utility and is accessed by entering only the file specification preceded by the at sign:

```
>@INDIRECT.CMD
```

The default values for indirect file specifications are:

```
dev        - SY0:
uic        - The current UIC
file name  - No default
file type  - .CMD
version    - The latest version of the file
```

For complete information on how to use indirect files, refer to the RSX-11M/M-PLUS MCR Operations Manual.

9.3 PROCESSING BAD BLOCK DATA

This section contains information on how BAD tests the reliability of disks and DECtapes, formats bad block descriptor entries, and how the INI command uses bad block information.

9.3.1 Verifying Devices

BAD verifies disks and DECtapes by writing a test pattern onto each of the blocks on the device, reading the pattern back into a buffer in storage, and comparing the blocks in the buffer with those on the device. When BAD processes a disk or DECtape, all existing data is destroyed.

BAD writes the test pattern to several blocks in a single write operation. If an error occurs in writing, reading, or comparing any of these blocks, BAD tests each of the blocks individually. The test pattern, 165555 and 133333(octal), is replicated 128(decimal) per block. If BAD finds no bad blocks during individual testing, the error-logging subsystem may still log errors due to long data transfers.

9.3.1.1 **BAD and Non-Last Track Devices** - As BAD locates bad blocks, it stores their addresses in a memory buffer. After locating all bad blocks on a device, BAD records the addresses of the bad blocks on the last good block of the device. Consecutive bad blocks are recorded as single entries. There must be at least one good block in the last 256 (decimal) blocks of the volume for BAD to create this file, called the bad block descriptor file.

9.3.1.2 **BAD and Last-Track Devices** - BAD records bad block information differently on last-track devices, such as the RK06/07, RL01/02, and the RM02/03. The last track is divided into two areas, the Manufacturer's Detected Bad Sector File (MDBSF) and the Software Detected Bad Sector File (SDBSF). The MDBSF is created when the hardware servo/header writer formats the pack. This operation also sets bits in any header that is marked bad in the MDBSF, and sets the SDBSF to be empty. When you run BAD, entries are made in the SDBSF. The information contained in the two last-track files is combined to form [0,0]BADBLK.SYS when you issue the INI command.

9.3.2 Format of Bad Block Descriptor Entries

For non-last track devices, BAD uses the last good block as a descriptor file for bad blocks. The address of a bad block, or the first address in a sequence of consecutive bad blocks, is stored as a double-word entry in the bad block descriptor file. The first word of this double-word contains two entries: the high-order byte contains the number of bad blocks minus 1 and the low order byte contains bits 16 through 23 of the logical block number of a bad block or a range of bad blocks. The second word of the double-word contains bits 0 through 15 of that block number.

For last-track devices, bad block descriptor entries are also recorded as a double-word in the SDBSF. Word 1 of the double word contains the address of the cylinder on which the bad block exists. The high-order and low-order bytes of word 2 contain, respectively, the track and sector addresses of the bad block.

9.3.3 The INI Command and BAD

Use BAD with the MCR INI Command to produce a Files-11 volume. The INI command uses the bad block information to create the file [0,0] BADBLK.SYS. The BADBLK.SYS file has allocated to it those blocks found to be bad, thus ensuring that file system does not allocate a known bad block to a file.

For information on how to use the INI, refer to the RSX-11M/M-PLUS MCR Operations Manual.

9.4 USING THE BAD UTILITY

Before BAD can validate a device, that device must be formatted by the manufacturer, by the FMT utility, or by one of the diagnostic programs supplied in your distribution kit. These programs are described in the RSX-11 Utilities Manual and the RSX-11M/M-PLUS User Mode Diagnostics Reference Manual, respectively.

In the RSX-11M system, the volume must not be mounted. Issue the MCR dismount command if the device contains a mounted volume.

In an RSX-11M-PLUS system, the volume must be mounted as foreign.

The following example illustrates a typical sequence of steps for introducing the disk DK1: to an RSX-11M or RSX-11M-PLUS system.

System	Command
M/M+	ALL DK1:<CR>
M+	MOU DK1:/FOR<CR>
M/M+	FMT DK1:[/sw]<CR>
M/M+	BAD DK1:[/sw]<CR>
M/M+	INI DK1:[label][/sw]<CR>
M+	DMOU DK1:<CR>
M/M+	MOU DK1:[label][/sw]<CR>

You may execute BAD while other RSX-11 tasks are executing.

9.4.1 Programming Considerations

This section contains information you should know before you use the BAD utility.

9.4.1.1 **Use of Block Zero** - On bootable disks, block zero contains the bootstrap block. If block zero is bad, BAD prints a message warning the operator not to use the disk for a bootable system image.

9.4.1.2 **Device Controller Errors** - The error-logging subsystem may record errors even though BAD is not reporting bad blocks. These errors may be encountered during long data transfers and may originate with the device controller.

9.4.1.3 **Maximum Entries in the BAD Block Descriptor** - On non-last track devices BAD storage allows 102.(decimal) entries of bad block addresses. On last-track devices BAD allows 126. entries. On non-last track devices, a single BAD entry can address one bad block or several consecutive bad blocks. If more than the maximum number of entries is recorded, BAD terminates with an error message.

9.5 BAD SWITCH DESCRIPTIONS

9.5.1 Switches for both Task and Stand-Alone System Versions Of BAD

/LI

Causes all bad blocks to be printed by number in decimal on your terminal. This switch causes bad blocks to be listed as BAD performs a data pattern check on each block. Blocks entered manually which BAD tests as reliable are not listed. This switch is valid for all devices.

/MAN

Causes BAD to first prompt you for bad block information and to then perform data pattern checking. Any block that you enter is included in the bad block descriptor file.

/OVR

Causes BAD to ignore last track information and write a bad block descriptor file on the last good block before the last track. In other words, the override switch causes BAD to treat a last-track device as a non-last track device. If your device has no bad block file on the last track, or if you suspect the reliability of the last track, use the override switch before using the MCR INI command. The override switch is valid only for last-track devices.

NOTE

If you use this switch, the /BAD=[OVR] option for initializing a volume must also be used to construct the bad block file [0,0]BADBLK.SYS. See the RSX-11M/M-PLUS MCR Operations Manual for a description of the MCR INI command.

/RETRY

Causes BAD to attempt a recovery of hardware errors via the device driver. This also means that soft errors, such as an ECC correctable error, will be recovered and the block will be marked as good.

/UPDATE

Causes BAD to immediately read the bad block descriptor file and prompt you for additional bad block input. This switch does not cause BAD to write pattern checks.

NOTE

Updating the bad block descriptor file on file structured volumes does not cause the file [0,0]BADBLK.SYS to be updated.

9.5.2 The Manual and Update Switches

If you wish to enter bad blocks in manual or update mode, BAD will prompt you as follows:

```
BAD>LBN(S)=
```

You may then enter bad blocks in the format:

```
blocknum:number
```

Where number specifies the number of sequential bad blocks beginning at the specified block number blocknum. The colon is required when you specify a sequence of bad blocks in this form. Both blocknum and number default to decimal values, unless preceded by a number sign (#) to indicate an octal value. For example:

```
BAD LBN(S)=70.3
```

enters the block numbers 70,71, and 72 in the bad block descriptor file.

You can also specify a single bad block. For example:

```
BAD>LBN(S)=3
```

enters block 3 in the bad block file.

You can use both of these forms on the same command line. For example:

```
BAD>LBN(S)= 100:2,3,  200:100  45:1
```

enters blocks 100, 101, 3, 200 through 299, and 45 in the bad block file. You can separate bad block sequences with a space, tab, or comma.

When you enter a carriage return in response to the prompt, BAD will list all the sequences in the bad block descriptor file. For example:

```
BAD>LBN(S)=
  000100:002
  000003:001
  000200:100
  000045:001
BAD>LBN(S)=
```

The first number in the display represents the beginning block of the sequence; the second number represents the number of bad blocks. Bad block numbers are listed in decimal.

When a bad block sequence is entered, BAD determines if these bad blocks are adjacent to an already existing sequence. If you are using a non-last track device, BAD appends your bad block entry to the existing sequence. If you are using a last track device, BAD records individual bad blocks in core memory but lists entries at your terminal as part of existing bad block sequences.

When you have finished supplying information in manual or update mode, enter ESCAPE, ALTMODE, or <CTRL/Z> in response to the prompt. The bad block will then either be rewritten with the new bad block information if in update mode or pattern checking will start if in manual mode. Blocks entered manually which BAD tests as reliable are included in the bad block descriptor file.

9.5.3 Switches for Stand-Alone System Version Only

/CSR=nnnnnn

nnnnnn is a new CSR address

This switch alters the CSR address of the device so that it conforms to that of the device in te user's system. /CSR remains in effect and need not be repeated if more commands are issued.

/VEC=nnn

nnn is a new interrupt vector address

This switch alters the interrupt vector address so that it conforms to the vector address of the device in the user's system. /VEC remains in effect if more commands are issued.

/WCHK

This switch causes a write check operation to occur after each write operation. The switch is not available for devices DT, DX, or DY.

/NOCHK

This switch negates /WCHK. This switch returns you to the default.

Unlike PRESRV and DSC, BAD expects to see all switches on a single command line. For example:

```
BAD> DM3:/OVR/LI/VEC=300/CSR=174406
```

locates all bad blocks on DM3, ignores the last track data, lists all bad blocks, changes the interrupt vector to 300, and changes the CSRss to 174406. All switches are validated for prop syntax before the actual bad block detection takes place.

9.6 DEVICES SUPPORTED BY THE STAND-ALONE VERSION

The following devices are supported by the stand-alone version of BAD. If you have a task version of BAD, your executive will support any device suitable to your system's configuration.

Mnemonic	Type	CSR	Vector
DB	RH11-RP04/05/06 and RH70-RP04/05/06 Disk Pack	176700	254
DD	TU58 DECtape II	175600	300
DF	RF11/RS11 Fixed-Head Disk	177460	204
DK	RK11-RK03/05/05F Cartridge Disk	177404	220
DL	RL11/RL01/RL02 Cartridge Disk Pack	174400	160
DM	RK611-RK06/07 Cartridge Disk Pack	177440	210
DP	RP11-C/E RPR02/RP02/03 Disk Pack	176714	320[1]
DR	RH70/RM03 and RH11/RM02 Disk Pack	176700	340[1]
DS	RH11-RS03/04 and RH70-RS03/04	172040	310[1]
DT	TC11/TU56 DECtape	177342	214
DX	RX11/RX01 Floppy Disk	177170	264
DY	RX211/RX02 Floppy Disk	177170	350[1]

[1] Nonstandard Vector Address

9.7 BAD MESSAGES

This section lists the BAD messages, gives a brief description of the condition that causes each message, and suggests a response to the condition. BAD error messages are arranged alphabetically beginning with the text following after the device symbol, [ddu:].

BAD -- ddu: BAD BLOCK FILE NOT FOUND

Explanation: The bad block descriptor file could not be read in update mode.

User Action: You must use the device without updating the bad block file, or reformat the device and destroy all data.

BAD -- ddu: BAD BLOCK FILE OVERFLOW

Explanation: BAD detected more than 102 sets of bad blocks. For last-track devices, the maximum number of bad blocks that can be recorded is 126. This message usually indicates a device unit failure.

User Action: Either the volume is bad or the drive requires maintainance; contact your DIGITAL Field Service Representative.

BAD -- ddu: BAD BLOCK FOUND - LBN= nnnnnn.

Explanation: Bad blocks are reported in this format, where LBN is the Logical Block Number (decimal).

User Action: None. This message is informational and applies to the /LI switch only.

BAD -- ddu: BLOCK 0 BAD - DO NOT USE AS SYSTEM DISK

Explanation: This is a warning message. When block zero is bad, a bootstrap block cannot be written on the disk, making it useless as a system disk.

User Action: Label the disk to ensure that no one attempts to use it as a system disk.

BAD -- COMMAND I/O ERROR

Explanation: There was a hardware transmission error from the keyboard.

User Action: Retype the command.

BAD -- COMMAND TOO LONG

Explanation: The command was longer than 80. characters.

User Action: Retype the command.

BAD -- ddu: CSR ADDRESS NOT IN SYSTEM

Explanation: Self-explanatory. This message occurs only in the stand-alone system version of BAD.

User Action: Retype the command including the /CSR switch with the proper value.

BAD -- ddu: DEVICE OFFLINE

Explanation: In the stand-alone version of BAD, the specified device is not in the hardware configuration, or the Control Status Register switch (/CSR) is improperly set.

User Action: Retype the command, setting the /CSR and /VEC addresses for the device to the proper addresses.

BAD -- DUPLICATE BLOCK NUMBER - [numb]

Explanation: The block number sequence you entered is already present in the bad block file. The value [numb] is the sequence you entered.

User Action: Reenter the command line with another value. This message applies to the /MAN or /UP switch only.

BAD -- ddu: FAILED TO ATTACH

Explanation: BAD could not gain control of the unit to be tested.

User Action: Determine if another task has attached the unit. If so, wait until the task exists, or abort the task to gain control of the unit for BAD.

BAD -- ddu: FAILED TO READ MANUFACTURER'S BAD SECTOR FILE

Explanation: A disk-read hardware error occured while attempting to read the factory-written bad block data on the last-track device cartridge.

User Action: Reenter the command with the /OVR switch included.

BAD -- ddu: FAILED TO READ SOFTWARE BAD SECTOR FILE

Explanation: The software-detected bad sector file could not be read in update mode.

User Action: Reenter the command with the /OVR switch included.

BAD -- ddu: FAILED TO WRITE BAD BLOCK FILE

Explanation: BAD could not write the bad block file. This condition usually results from a disk write error.

User Action: Reenter the command. If the problem persists the disk pack should be discarded.

BAD -- ddu: FATAL HARDWARE ERROR

Explanation: Self-explanatory.

User Action: Contact your DIGITAL Field Service Representative.

BAD -- ddu: HANDLER/DRIVER MISSING

Explanation: The disk driver is not loaded.

User Action: Load the disk driver and reenter the command.

BAD -- ddu: ILLEGAL DEVICE

Explanation: The device to which bad block processing is directed does not support a Files-11 structure.

User Action: You must re-format your device before running BAD.

BAD -- INVALID BLOCK NUMBER - [numb]

Explanation: You entered an invalid block number sequence. The value [numb] is the sequence.

User Action: Type another value and reenter the command line. This message applies to the /MAN or /UP switch only.

BAD -- INVALID SWITCH

Explanation: Self-explanatory

User Action: Reenter the command with a proper switch.

BAD -- ddu: IS AN ALIGNMENT CARTRIDGE

Explanation: The factory written label on the last track of a last-track device cartridge indicates an alignment cartridge.

User Action: Mount and process another cartridge.

BAD -- ddu: NOT IN SYSTEM

Explanation: The requested device was not made part of the system during system generation, or the device does not exist on the host configuration.

User Action: Ensure that you entered the command line correctly.

BAD -- ddu: NOT READY

Explanation: The unit had not reached operating speed when BAD attempted to access it.

User Action: Allow the unit to reach operating speed and reenter the command line.

BAD -- ddu: MANUFACTURER'S BAD SECTOR FILE CORRUPT

Explanation: The factory-written bad block data in the last track of a last-track device is in an inconsistent format.

User Action: Reenter the command with the /OVR switch included.

BAD -- ddu: PRIVILEGE VIOLATION

Explanation: An operation was attempted for a device that was mounted or allocated to another user.

User Action: Mount another device and reenter the command line.

BAD -- SYNTAX ERROR

Explanation: BAD detected a syntax error on the command line.

User Action: Determine the correct syntax and reenter the command line.

BAD -- ddu: TOTAL BAD BLOCKS = [n].

Explanation: This is an information message indicating the total number (in decimal) of bad blocks on the disk.

User Action: Write the bad blocks count on the disk label.

BAD -- ddu: UNRECOVERABLE ERROR [n]

Explanation: An I/O error caused BAD to terminate. The value [n] is number of the I/O error returned by the driver.

User Action: If the same error persists, contact your local Digital field representative.

BAD -- ddu: VECTOR NOT MULTIPLE OF FOUR

Explanation: Self-explanatory.

User Action: Retype the command including the /VEC switch with the proper value.

BAD -- ddu: WRITE LOCKED

Explanation: The unit is write-locked.

User Action: Write-enable the unit and reenter the command line.

CHAPTER 10

BACKUP AND RESTORE UTILITY (BRU)

The RSX-11 Backup and Restore Utility (BRU) provides functions that allow you to back up and restore Files-11 volumes. BRU transfers files from a volume to a backup volume or volumes to ensure that a copy of the files is available in case the original files are destroyed. If the original files are destroyed, or if for any other reason the copy needs to be retrieved, you can restore the back-up files with the BRU command.

Backup and restore operations take place on disk and tape volumes:

- Disk to tape -- for backup operations
- Tape to disk -- for restore operations
- Disk to disk for either backup or restore operations

In addition to these basic data transfer functions, the Backup and Restore Utility provides command qualifiers to:

- Perform backup and restore operations by file specification, date, and time
- Control disk processing, which provides disk initialization features that allow you to enter the locations of bad blocks on a disk and initialize the disk before a backup or restore operation
- Control tape processing such as density, length, ANSI tape labeling, rewind, and append
- Verify accurate data transfers
- Display information such as backup set names and file names

10.1 OVERVIEW OF COMMAND QUALIFIERS AND DEFAULTS

Table 10-1 summarizes the command qualifiers available for backup and restore operations.

Table 10-1
Summary of BRU Command Qualifiers

Command Qualifiers	Options	Default
/APPEND		None
/BACKUP_SET:name		None
/BAD:	[MANUAL] [AUTOMATIC] [OVERRIDE]	BAD:AUTOMATIC
/BUFFERS:number		Number of buffers from the input disk
/COMPARE		
/CREATED:	[BEFORE:dd-mmm-yy hh:mm:ss] [AFTER:dd-mmm-yy hh:mm:ss]	Current date
/DENSITY:number		DENSITY:800.
/DIRECTORY		None
/DISPLAY		None
/ERRORS:number		ERRORS:25.
/EXCLUDE		None
/EXTEND:number		Number of blocks from the input disk
/HEADERS:number		Number of headers from the input disk
/INITIALIZE		None
/INVOLUME:name		None
/LENGTH:number		The length of the output tape
/MAXIMUM:number		Maximum number of files from the input disk
/MOUNTED		None

(continued on next page)

Table 10-1 (Cont.)
Summary of BRU Command Qualifiers

Command Qualifiers	Options	Default
/NEW_VERSION		None
/NOINITIALIZE		None
/NOPRESERVE		None
/NOSUPERSEDE		None
/OUTVOLUME:name		Input disk volume name
/POSITION:	[BEGINNING] [MIDDLE] [END] [block:number]	Index file position on the input disk
/PROTECTION:	(protection value) SYSTEM:value OWNER:value GROUP:value WORLD:value	Protection of the input disk
/REVISED:	[BEFORE:dd-mmm-yy hh:mm:ss] [AFTER:dd-mmm-yy hh:mm:ss]	Current date
/REWIND		None
/SUPERSEDE		/NOSUPERSEDE
/TAPE_LABEL:label		None
/VERIFY		None
/WINDOWS:value		Number of mapping pointers on input disk

10.1.1 Command Qualifiers For Selective Backup And Restore

The command qualifiers described in Table 10-2 allow you to backup or restore data, using one of the following criteria for file selection:

- File specification
- Date and time of creation
- Date and time of revision

Table 10-2
Command Qualifiers That Perform Selective Backup and Restore Operations

Qualifier	Options	Explanation
/CREATED	BEFORE:(dd-mmm-yy[hh:mm:ss]) AFTER:(dd-mmm-yy[hh:mm:ss])	Directs BRU to process files created before or after a specified date and/or time.
/EXCLUDE		Selectively excludes from a backup or restore operation all files specified on the command line.
/NEWVERSION		Directs BRU to resolve conflicts resulting from files with identical file specifications by creating a new version of the file. Both versions of the file are kept on the output volume.
/NOSUPERSEDE		When files on the output volume have file specifications identical with files on the input volume, resolves the conflict by keeping the file on the output volume rather than that on the input volume.
/REVISED	BEFORE:(dd:mmm:yy[hh:mm:ss]) AFTER:(dd:mmm:yy[hh:mm:ss])	Directs BRU to process files revised before or after a specified date and/or time.
/SUPERSEDE		In restore operations, restores files to an existing disk; resolves file specification conflicts by deleting the old file on the output disk and replacing it with the file being restored from the input disk.

The following qualifiers resolve file specification conflicts when, during a restore to an existing disk, a file specification on the output volume is identical to one for a file being transferred from the input volume:

- /NEWVERSION
- /SUPERSEDE
- /NOSUPERSEDE

10.1.2 Command Qualifiers for Controlling Disk Processing

The command qualifiers described in Table 10-3 allow you to control the way the operating system manages disk data transfer operations.

Table 10-3
Command Qualifiers that Control Disk Processing

Qualifier	Options	Explanation
/INITIALIZE		Directs BRU to initialize a disk before proceeding with the data transfer.
/INVOLUME	name	Specifies the volume label of the input disk.
/NOINITIALIZE		Specifies that you do not want to initialize the output disk; it already has a Files-11 structure and is mounted as Files-11.
/MOUNTED		Allows you to back up files from a disk that is mounted as a Files-11 volume (via the MCR MOUNT command).
/NOPRESERVE		Specifies that you do not want BRU to preserve file identifiers.
/OUTVOLUME	name	Specifies the volume label of the output disk.

10.1.3 Command Qualifiers for Controlling Tape Processing

The qualifiers in Table 10-4 allow you to control backup and restore tape processing.

Table 10-4
Command Qualifiers That Control Tape Processing

Qualifier	Options	Explanation
/APPEND		Appends new backup data to a tape with one or more backup sets.
/BACKUP_SET	name	Specifies the name of the backup set to be placed on tape. Refer to Section 10.5.1.
/DENSITY	number	Specifies the data density at which BACKUP writes to tape.
/ERRORS	number	Specifies the number of nonfatal I/O errors BRU tolerates on tape reads before automatically terminating execution.
/LENGTH	number	Specifies the length of the output tape in decimal feet.
/REWIND		Rewinds the first tape of a tape set before execution of a command line.
/TAPE_LABEL	label	Specifies a 6-character ANSI tape label.

10.1.4 Command Qualifiers for Verifying the Accuracy of Data Transferred

The command qualifiers in Table 10-5 allow you to detect differences between data on the input volume and data on the output volume.

Table 10-5
Command Qualifiers That Verify the Accuracy of Data Transfers

Qualifier	Explanation
/COMPARE	Compares the data on the output volume to the data on the input volume and reports any differences.
/VERIFY	Copies data from the input volume to the output volume, performs a compare operation, and reports differences.

10.1.5 Command Qualifiers for Displaying Information About Files Transferred

The command qualifiers listed in Table 10-6 display information about the data being transferred.

Table 10-6
Command Qualifiers That Provide Information

Qualifier	Explanation
/DISPLAY	Displays at your terminal the UFD and filename of each file being backed up.
/DIRECTORY	Display information (such as backup set names, file names, or volume number of a tape) on a specified tape volume.

10.1.6 Command Qualifiers for Initializing Disks

When you are initializing a disk using the BRU /INITIALIZE qualifier, you may want to specify various characteristics for the output disk. The qualifiers in Table 10-7 allow you to set the characteristics.

10.2 DISK AND TAPE DEVICE INFORMATION

You can use the Fast Backup and Restore Utility only on Files-11 volumes. The following sections give a brief overview of the Files-11 format and also present some general disk device information.

For more detailed information on Files-11, refer to the IAS/RSX-11 I/O Operations Reference Manual.

10.2.1 Files-11 Structures

BRU requires that an input disk be a Files-11 volume. Also, when you initialize a volume for Files-11, five control files are created. These five files are catalogued in the Master File Directory ([0,0] also called the MFD):

- The Index File -- identifies the volume to the operating system as a Files-11 structure contains control data for accessing all files on the volume. The index file is listed in the Master File Directory as INDEXF.SYS.
- The Storage Bit Map File -- controls the available space on the volume. This file is listed in the MFD as BITMAP.SYS. It contains a storage control file, which consists of summary information for optimizing the allocation of storage. It also contains the bit map itself, which lists available blocks of storage.
- The Bad Block File -- is listed in the MFD as BADBLK.SYS. It contains all of the bad blocks on the volume. Bad block processing is discussed in more detail in Section 10.4.2.
- The Master File Directory (MFD) -- listed in the MFD as 000000.DIR. The MFD is the root of the volume directory structure. It lists both the reserved files that control the volume structure and user file directories.

- The Core Image File -- listed in the MFD as CORIMG.SYS. This file contains the system checkpoint file.

The following sections give you background information you need to efficiently use the BRU command qualifiers for controlling operating system disk processing operations.

Table 10-7
Initialization Qualifiers

Qualifier	Options	Explanation
/BAD	MANUAL AUTOMATIC OVERRIDE	Enters the locations of bad blocks on volumes.
/BUFFERS	number	Specifies the number of directory File Control Blocks (FCBs) per volume kept in memory by the ACP.
/EXTEND	number	Specifies the number of blocks to extend a file when that file has exhausted its allocated space.
/HEADERS	number	Specifies the number of file headers to allocate initially to the index file.
/MAXIMUM	number	Specifies the maximum number of files that can be placed on a volume.
/POSITION	BEGINNING MIDDLE END BLOCK:number	Specifies the location of the index file, Master File Directory, and the storage allocation file on a disk.
/PROTECTION	protection value SYSTEM:value OWNER:value GROUP:value WORLD:value	Specifies the global protection status of the disk.
/WINDOWS	number	Specifies the number of retrieval pointers allocated for use with file windows.

Table 10-6
Command Qualifiers That Provide Information

Qualifier	Explanation
/DISPLAY	Displays at your terminal the UFD and filename of each file being backed up.
/DIRECTORY	Display information (such as backup set names, file names, or volume number of a tape) on a specified tape volume.

10.1.6 Command Qualifiers for Initializing Disks

When you are initializing a disk using the BRU /INITIALIZE qualifier, you may want to specify various characteristics for the output disk. The qualifiers in Table 10-7 allow you to set the characteristics.

10.2 DISK AND TAPE DEVICE INFORMATION

You can use the Fast Backup and Restore Utility only on Files-11 volumes. The following sections give a brief overview of the Files-11 format and also present some general disk device information.

For more detailed information on Files-11, refer to the IAS/RSX-11 I/O Operations Reference Manual.

10.2.1 Files-11 Structures

BRU requires that an input disk be a Files-11 volume. Also, when you initialize a volume for Files-11, five control files are created. These five files are catalogued in the Master File Directory ([0,0] also called the MFD):

- The Index File -- identifies the volume to the operating system as a Files-11 structure contains control data for accessing all files on the volume. The index file is listed in the Master File Directory as INDEXF.SYS.
- The Storage Bit Map File -- controls the available space on the volume. This file is listed in the MFD as BITMAP.SYS. It contains a storage control file, which consists of summary information for optimizing the allocation of storage. It also contains the bit map itself, which lists available blocks of storage.
- The Bad Block File -- is listed in the MFD as BADBLK.SYS. It contains all of the bad blocks on the volume. Bad block processing is discussed in more detail in Section 10.4.2.
- The Master File Directory (MFD) -- listed in the MFD as 000000.DIR. The MFD is the root of the volume directory structure. It lists both the reserved files that control the volume structure and user file directories.

- The Core Image File -- listed in the MFD as CORIMG.SYS. This file contains the system checkpoint file.

The following sections give you background information you need to efficiently use the BRU command qualifiers for controlling operating system disk processing operations.

Table 10-7
Initialization Qualifiers

Qualifier	Options	Explanation
/BAD	MANUAL AUTOMATIC OVERRIDE	Enters the locations of bad blocks on volumes.
/BUFFERS	number	Specifies the number of directory File Control Blocks (FCBs) per volume kept in memory by the ACP.
/EXTEND	number	Specifies the number of blocks to extend a file when that file has exhausted its allocated space.
/HEADERS	number	Specifies the number of file headers to allocate initially to the index file.
/MAXIMUM	number	Specifies the maximum number of files that can be placed on a volume.
/POSITION	BEGINNING MIDDLE END BLOCK:number	Specifies the location of the index file, Master File Directory, and the storage allocation file on a disk.
/PROTECTION	protection value SYSTEM:value OWNER:value GROUP:value WORLD:value	Specifies the global protection status of the disk.
/WINDOWS	number	Specifies the number of retrieval pointers allocated for use with file windows.

10.2.1.1 **The Index File** - The index file contains the following information:

- Bootstrap Block -- is virtual block number 1 of the index file. If the volume is a PDP-11 system device, this block contains a bootstrap program that loads the operating system into memory.

  If the volume is not a system device, this block contains a program that displays a message indicating that the volume does not have a hardware-bootable system.

- Home Block -- establishes the specific identity of the volume, providing such information as the volume name and protection, maximum number of files allowed on the volume, and volume ownership information. The home block is virtual block number 2 of the index file.

- Index File Bit Map -- controls the allocation of file headers and thus the number of files on the volume. The bit map contains a bit for each file header allowed on the volume. The index file bit map starts at virtual block number 3 of the index file and continues for the number of blocks necessary to contain the bit map.

- File Headers -- comprise the largest part of the index file. Each file on the volume has a file header, which describes such attributes as file ownership, creation date and time, and file protection. The file header contains all the information necessary for accessing the file.

10.2.2 Disk Volume Labels and Backup Set Names

If you omit the backup set name in a backup operation, BRU uses the volume label of the input disk as the backup set name of the output tape.

10.2.3 Devices Supported for Backup and Restore

Table 10-8 lists all the devices that BRU supports.

Table 10-8
Devices Supported By BRU

Mnemonic	Type	Class
DD	TU58 cassette (DECtape II)	Block-structured
DM	RK611/RK06/RK07 cartridge disk	Block-structured
DB	RH11/RP04/RP05/RP06 and RH70/RP04/RP05/RP06 disk pack	Block-structured
DK	RK11/RK05/RK05F cartridge pack	Block-structured

(continued on next page)

Table 10-8 (Cont.)
Devices Supported By BRU

Mnemonic	Type	Class
DF	RF11/RS11 fixed head disk	Block-structured
DL	RL11/RL01 cartridge disk	Block-structured
DP	RP11/RP02/RP03 disk pack	Block-structured
DR	RH70/RM03	Block-structured
	RH11/RM02 disk pack	
DS	RH11/RS03/RS04 and RH70/RS03	Block-structured
	RS04 fixed head disk	
DT	TC11/TU56 DECtape	Block-structured
DX	RX11/RX01 Floppy Disk	Block-structured
DY	RX211/RX02 Floppy Disk	Block-structured
MM	RH11/TM02-03/TE16/TU16/TU45/TU77 and RH70/TM02-03/TE16/TU16/TU45 and TU77 9-track magnetic tape	Tape
MT	TM11/TU10/TE10 7- or 9-track magnetic tape and TS03 9-track magnetic tape	Tape
MS	TS04 magnetic tape	Tape

10.3 BACKUP AND RESTORE COMMAND LINE DESCRIPTIONS

This section describes the rules for entering command lines for the Backup and Restore utility. The section includes the definition of the command line syntax and descriptions of command line parameters, command qualifiers, and prompts.

10.3.1 Command Line Syntax Definition

The general syntax of the BRU command line is:

```
BRU /qualifiers indevice1:,...[filespec,...] outdevice1:,...[filespec,...]
```

10.3.2 Descriptions of Prompts

Prompts

FROM:

TO:

INITIALIZE [Y/N]:

FROM

Requests that you enter the name(s) of the devices on which the input volume(s) reside in the form specified in the description below of the command parameters.

TO

Requests that you enter the name(s) of the output devices in the form specified in the description below of the command parameters.

INITIALIZE [Y/N]

Issued as a precaution to ensure that you want to erase the output volume. Enter Y (for YES) if you want to initialize the volume.

10.3.3 Description of Command Line Parameters

qualifiers

Specifies any of the command qualifiers specified above. If more than one qualifier is specified, they must be contiguous; that is, you may not separate the qualifiers with blanks or any other delimiters. The qualifiers may appear in any order.

You may abbreviate a qualifier as long as you use a unique abbreviation. For example:

```
BRU>/REW/INI/OUT:BACKUP MM0:  DK0:
BRU - STARTING TAPE 1 ON MM0:

BRU -- *WARNING* -- THIS DISK  WILL  NOT  CONTAIN  A  HARDWARE
BOOTABLE SYSTEM

BRU - END OF TAPE 1 ON MM0:

BRU - COMPLETED
```

When a qualifier has options, you must separate the qualifier from the option by a colon in the form:

```
/qualifier:option
```

indevice

Specifies the physical device or devices from which data is transferred. For tapes, you may specify more than one input device. Devices are specified in the form:

```
DD[uu]:
```

where DD represents the device mnemonic and uu represents the octal unit number associated with that device. The unit number may be specified as one or two digits; the default unit number is 0. For example, a TU77 tape drive can be referenced as MM00:, MM0:, MM:, MM01:, MM1:, and so forth, depending on your configuration. The colon is a required delimiter.

When more than one device is specified, the device mnemonics must be separated by commas.

filespec

Indicates the file specification used to select particular files or categories of files to be backed up or restored. The filespec takes the form:

[n,m]filename.filetype;version

You may specify up to 16 filespecs per command line.

Files can be backed up or restored selectively by UFD, filename, filetype, or version number. When backing up or restoring selectively by version number, you must specify either an explicit version number or no version number at all or a wildcard (*), which has the same effect as no version number. BRU does not support 0 or -1 as version numbers.

outdevice

Specifies the output device to which data is being transferred. For tapes, you may specify more than one output device. The rules are the same as for the "indevice" above.

10.3.3.1 **Entering Command Lines with No File Specifications** - When you enter a command line with no file specification, all the files on the input volume are copied to the output volume.

10.3.3.2 **Wildcards in Input Specifications** - The following wildcard features are provided for file specifications:

[*,*] means all group,member combinations.
[n1,*] means all member numbers under group n1.
[*,n2] means all group numbers for member n2.

BRU supports the wildcard (*) in all the elements of a file specification: file name, file type, and version number. BRU generally follows the RSX-11M rules for use of wildcards, except in the following two instances:

- When you omit a file specification element, BRU treats the omitted element as if it were a wildcard. That is, when you specify only file name and file type in a file specification, all version numbers are transferred in the backup or restore operation.
- When you specify particular UFDs on a command line, all the files in those UFDs are transferred in the backup or restore operation.

10.3.3.3 **Continuation Lines** - BRU allows you to continue a command line onto more than one line by using a hyphen (-) as the continuation character.

On RSX-11M Version 3.2 systems, BRU supports continuation lines only when invoked as BRU>. BRU does not support continuation lines when run from an indirect command file or when the command line is entered on the same line as the one on which BRU is invoked.

On RSX-11M-PLUS systems BRU supports continuation lines under all circumstances.

The following examples show continued BRU command lines on RSX-11M and RSX-11M-PLUS.

RSX-11M

```
>RUN BRU
BRU>/REWIND-
BRU>/INVOLUME:BACKUP-
BRU>/BACKUP_SET:25MAY79-
BRU>/TAPE_LABEL:BRU123
FROM:    DK0:
TO:      MM0:
BRU - STARTING TAPE 1 ON MM0:

BRU - END OF TAPE 1 ON MM0:

BRU - COMPLETED

BRU>^Z
```

RSX-11M-PLUS

```
>BRU /REWIND-
MCR> /INVOLUME:BACKUP-
MCR> /BACKUP_SET:25MAY79-
MCR> /TAPE_LABEL:BRU123 DK0:  MM0:
>
```

10.3.4 Description of Command Qualifiers

/APPEND

Directs BRU to append a backup set from the input volume to the last backup set on the output tape.

The output tape may not be a continuation tape in a backup set; it cannot contain a backup set that is continued on another volume.

/APPEND causes BRU to skip to the logical end-of-tape before it writes the new backup set, if the tape was positioned at the beginning. BRU searches the output volume for the last logical end-of-file. If the tape is a continuation tape, i.e., not the first tape of a tape set, BRU displays an error message.

If the last backup set does not end on the tape, BRU displays an error message.

If the tape is not positioned at the beginning, /APPEND will cause BRU to start writing where the tape is currently positioned. /APPEND/REWIND will cause BRU to rewind the tape and then space forward until the end of the volume.

/BACKUP_SET:name

Specifies the name of the backup set (refer to Section 10.5.2) to be placed on tape. Default is the volume name of the disk being backed up. This name may be up to 12 characters long.

When applied to an output tape volume, the backup set name assigns the name of the backup set being placed on the volume. BRU supports multiple backup sets on a single volume.

If you specify no backup set name for the output volume, BRU uses the volume name of the input disk to name the backup set.

When applied to an input tape volume, BRU searches the first tape for the specified backup set name. If you specify no backup set name with the input volume, BRU restores the first backup set it finds on the tape.

BRU does not rewind the first tape in a backup set unless you specify /REWIND.

```
/BAD:[  AUTOMATIC       ]
     [  OVERRIDE        ]
     [  MANUAL          ]
```

The /BAD qualifier is used with the /INITIALIZE qualifier during tape to disk or disk to disk operations.

For complete information on how to use the /BAD qualifiers, refer to Section 10.4.2, "Using the Bad Block Utility with Backup and Restore." The following are summary descriptions only.

For last-track devices, AUTOMATIC causes BRU to use the manufacturer-written bad block information and the software-detected bad sector file to create BADBLK.SYS. For nonlast-track devices, it uses the software bad block descriptor block to create BADBLK.SYS.

OVERRIDE applies only to last-track devices, causing the last-track device to appear to be a nonlast-track device. When OVERRIDE is specified, BRU uses the software bad block descriptor block to create BADBLK.SYS.

MANUAL specifies that BRU use both the manufacturer-written bad block information and either the software-detected bad sector file (for last-track devices) or the bad block descriptor block (for nonlast-track devices) to create BADBLK.SYS. Also, MANUAL accepts the addresses of bad blocks entered manually from the terminal.

/BUFFERS:fcbcount

It specifies the default number of directory File Control Blocks (FCBs) per volume kept in memory by the ACP when the volume is mounted. The more FCBs that are kept in memory, the faster that files contained in heavily used directories are found. The default value for fcbcount is the same as the default for the input disk.

The /BUFFERS qualifier is used with the /INITIALIZE qualifier during tape-to-disk or disk-to-disk operations.

/COMPARE

Compares the data on the output device with the data on the input device and reports any differences. No data transfer takes place during a COMPARE operation. The command string specifying the COMPARE operation must be identical to that entered when the data on the output disk or tape was created, with the exception of the /INITIALIZE and /NOINITIALIZE qualifiers.

/COMPARE Output: When /COMPARE detects differences, it displays a warning message at your terminal. /COMPARE always displays the device mnemonic on which the difference was detected and the type of record in which the difference is encountered (a control record, a header record, or a data record).

If the file is a header file, /COMPARE displays the file-ID for the file. If the file is a data file, /COMPARE displays the file-ID, the retrieval pointer for the file, and the name of the file if it is available.

/CREATED:[BEFORE:(dd-mmm-yy [hh:mm:ss])]
[AFTER:(dd-mmm-yy [hh:mm:ss])]

Backs up or restores files created on or before or after the specified date and/or time.

If you specify both a date and a time, the date and time must be enclosed in parentheses. If you specify only a date or only a time, the parentheses are not necessary. If you specify only a time, BRU uses today's date as the default.

/DENSITY:number

Specifies the density at which BRU writes to tape. The following table shows legal values you can specify.

Drive	Default Density	Optional Density
TU10	800	None
TU16	800	1600
TU45	800	1600
TU77	800	1600
TS04	1600	None

If you specify /DENSITY with /APPEND, you must specify the density at which the old tape data was written. For example, if the tape was first written at a density of 800, you must specify a density of 800. If you specify a density other than the original density, BRU displays a warning message and continues processing at the correct density.

If you enter an incorrect density for a restore operation, BRU displays an error message and terminates the operation.

/DIRECTORY

Lists backup set names or files on the specified tape volume.

Using /DIRECTORY to Display Backup Set Names: When specified with no backup set name, /DIRECTORY lists all the backup sets on the volume:

```
BRU /DIRECTORY MM0:

VOL1     BACKUP1 LABEL1  12-MAY-79
VOL1     BACKUP2 LABEL1  13-MAY-79
```

If a continuation tape is mounted when you enter the /DIR qualifier, BRU lists the backup sets on that volume, not the backup sets on the first or previous volumes. Also, /DIRECTORY displays continuation tape number.

Using /DIRECTORY to Display Files: To display the files in a backup set, enter the backup set name with /DIRECTORY in the form:

```
>
>RUN BRU
BRU>/BACKUP_SET:23MAY79A/DIRECTORY MM1:
VOL1.   23MAY79A        CDADOC  23-MAY-79 23:37:11
[000,000]
[303,013]
27DECE.LST;1
2JANA.LST;1
18JANC.LST;1
4JANA.LST;2
ASTCRSH.MAC;1
9DECA.LST;2
X.MAC;1
X.OBJ;1
X.TSK;1
APNDXC.TXT;1
X.MAP;1
[001,054]
RSX11M.STB;45
[002,054]
RSX11M.STB;36
[003,054]
27DECE.CDA;1
2JANA.CDA;1
18JANC.CDA;1
4JANA.CDA;1
5JANZ.CDA;1
19JANB.CDA;1
4JANB.CDA;1
6JAN79.CDA;1
9DECA.CDA;1
11JANA.CDA;1
15JANB.CDA;2
15JANC.CDA;2
11JAND.CDA;1
11JANA.CDA;2
15JANA.CDA;1
16JANA.CDA;1
12JANE.CDA;1
RSX11M.STB;3
[005,054]
[306,006]
APNDXB.MAC;1
BRU - COMPLETED ON MM1:

BRU>^Z
>BYE
>
HAVE A GOOD MORNING
```

If the backup set is not on the tape, BRU halts execution and displays a message at your terminal.

/DISPLAY

Prints at your terminal the file name and UFD of each file as the header for that file is being transferred by BRU.

/ERRORS:number

Requests termination of a restore operation after the specified number of nonfatal tape read errors. The default number of errors before termination is 25.

/EXCLUDE

Selectively excludes from a backup or restore operation the files specified on the command line.

/EXTEND:number

Specifies the default number of blocks by which a file can be extended when that file has exhausted its allocated space. This value is used by an ACP when the volume is mounted.

The /EXTEND qualifier is used with the /INITIALIZE qualifier during tape to disk or disk to disk operations.

/HEADERS:number

Specifies the number of file headers to allocate initially to the index file. The primary reason for preallocating file headers is to locate them near the storage bit map file. (The storage bit map file is generally located in the middle of the disk.) Proper placement of file headers can help reduce head motion during I/O operations.

The /HEADERS qualifier is used with the /INITIALIZE qualifier during tape to disk or disk to disk operations.

/INITIALIZE

Specifies that you want to initialize the output volume during a tape to disk or disk to disk operation. Initialization places a Files-11 structure on the disk, including the boot block (in some cases), the home block, and such files as INDEXF.SYS, BADBLKS.SYS, and 000000.DIR.

Along with the INITIALIZE qualifier, you can specify the following qualifiers when you are initializing a disk: BAD, BUFFERS, EXTEND, MAXIMUM, POSITION, PROTECTION, HEADERS, and WINDOWS. If you do not specify any of these qualifiers, BRU defaults to the characteristics of the input disk.

/NOINITIALIZE specifies that you do not want a Files-11 disk structure placed on the disk (one already exists). If you specify neither /INITIALIZE nor /NOINITIALIZE, BRU prompts at your terminal to ask whether you want to initialize the disk being processed.

When restoring a volume containing a disk boot block and a system that you want to be bootable, ensure that the volume to which you are restoring is the same as or greater than the size of the original volume. Also, the disk controller for the device on which the output device resides must be compatible with the device controller on which the original volume was created.

/INVOLUME:name

Specifies the volume label of the input disk or tape. This name can be up to 12 characters long.

For backup or disk-to-disk operations, /INVOLUME directs BRU to look for the volume label of the input volume in order to verify that the disk has the correct label. This check ensures that you do not back up the wrong volume.

For restore operations, /INVOLUME directs BRU to look on the input tape for the volume label in order to locate the correct backup set. If BRU cannot locate the label, it displays a message and terminates the operation.

/LENGTH:number

Specifies the length of the output tape in decimal feet. If the length specified exceeds the length of the tape, the entire length of the output tape will be used. In cases where you know the end of a tape must not be used, you can specify a shorter length to ensure that you do not write on that part of the tape.

/MAXIMUM:number

Specifies the maximum number of files that can be placed on a volume as determined by the number of file headers on the volume's index file. The default maximum is the maximum number of files on the input disk. The /MAXIMUM qualifier and the /HEADERS qualifier are particularly useful when you are initializing an output disk which is different in size from the input disk. Refer to section 10.4.5 for more detailed information about using the initialization qualifiers when you are transferring data between disks of different sizes.

/MOUNTED

Backs up or restores files on volumes that are mounted as Files-11 volumes (via the MCR MOUNT command).

BRU does not use F11ACP (Files-11) to access files on an input disk; rather, it uses logical I/O (QIO functions such as IO.RLB). BRU first backs up groups of headers, then backs up the files to which those headers point.

When backing up files from a mounted volume, disk activity (changes to or deletions of files) at while BRU is running causes the following results:

- If the file is being changed while BRU is backing up the disk, BRU copies only the data that comprises the file at the time of the transfer; any changes made to the file after the transfer will not appear on the volume to which you are backing up.

- If the file is deleted while BRU is backing up the disk, the data that comprises the file may be corrupted.

  If the file-ID from the deleted file is reused in a UFD that BRU has not yet backed up, BRU will back up the new file (with the previously allocated file-ID) when that file is encountered. When restored, this new file (with the duplicate file-ID) will appear as a synonym for the old file with the same file-ID.

- If the disk is changed (files are deleted or changed) after BRU generates the directory, the directory on the first tape of the tape set will not be accurate. Because BRU generates the directory for the backup set as its first processing step, changes to the disk after the directory is generated will not be reflected in the directory.

In order to back-up and restore files on mounted volumes, BRU must be built as a privileged task. For operations on unmounted volumes, BRU need not be a privileged task.

BRU will back up from both mounted and unmounted volumes. On RSX-11M-PLUS systems BRU treats a disk that is mounted foreign as an unmounted disk. BRU does not use the file system in either case. However, when you are doing a backup operation from a mounted volume BRU checks read access privileges of both UFDs and files against the UIC under which BRU is running.

BRU will restore to both mounted and unmounted volumes. To restore to an unmounted volume (or one mounted foreign on an RSX-11M-PLUS system), specify the /INITIALIZE qualifier to initialize the disk to a Files-11 structure. To restore to a mounted volume, specify /NOINITIALIZE to indicate to BRU that the disk is mounted and already has a Files-11 structure on it.

/NEWVERSION

Resolves filespec conflicts that occur during restore operations to an existing disk by creating new versions of the duplicate files.

/NOINITIALIZE

Specifies that you do not want to initialize the disk; it already has a Files-11 structure. The output disk must be mounted as a Files-11 volume. You cannot enter any of the initialization qualifiers listed above when you specify /NOINITIALIZE. If you enter any of these qualifiers, BRU issues an error message.

When you are restoring to a mounted volume (when you have specified /NOINITIALIZE), BRU uses the file system to access the output device. Therefore, a restore operation to a mounted volume is slower than a restore to an unmounted volume.

/NOPRESERVE

Specifies that you do not want to preserve file-IDs. If you specify /NOPRESERVE, BRU suppresses the warning message that file-IDs are not being preserved.

When file-IDs are not preserved BRU assigns new file-IDs starting at 6 and incrementing them sequentially.

/NOSUPERSEDE

When you are restoring to a mounted disk, specifies that when file specifications on the output disk are identical to those on the input disk, the file on the input disk is not transferred. That is, the file on the output disk is not superseded by the file on the input disk. NOSUPERSEDE is the default.

/OUTVOLUME:name

Specifies the volume label of the output disk. This label can be up to 12 characters long.

For backup operations, the name of the disk volume stored on the output tape volume is changed to the name specified with the /OUTVOLUME qualifier.

For restore operations or for disk-to-disk transfers, the name of the output disk volume is changed to the name specified with the /OUTVOLUME qualifier.

When you omit /OUTVOLUME, BRU provides the following defaults:

- When you omit /OUTVOLUME in backup operations, the input disk volume name is used as the volume name stored on tape.
- When you omit /OUTVOLUME in restore operations, the volume name stored on tape is used as the name of the output disk volume.
- When you omit /OUTVOLUME in disk-to-disk transfers, the volume name of the input volume is used as the volume name of the output volume.

```
/POSITION:[ BEGINNING          ]
          [ MIDDLE             ]
          [ END                ]
          [ BLOCK:number       ]
```

When initializing an output disk, specifies the location of the index file on the volume, usually to minimize access time. BEGINNING, MIDDLE, and END specify the beginning, middle, and end of a volume. BLOCK:number specifies a block number where the index file is to be placed. The BEGINNING position is generally used only when a disk contains predominently large contiguous files.

When you do not use the /POSITION qualifier, BRU places the index file in the same location as that on the input volume.

```
/PROTECTION:([protection value])
     protection values:
       SYSTEM:RWED
       OWNER:RWED
       GROUP:RWED
       WORLD:RWED
```

When initializing an output disk, specifies the default protection status for all files created on the volume being initialized. This protection value does not apply to files being transferred by BRU, but rather to subsequent files created on that volume by an ACP when the volume is mounted. The protection values above are standard for RSX-11M files; however if not specified, the values are defaulted to the protection values of the input disk.

```
/REVISED:[BEFORE:(dd-mmm-yy [hh:mm:ss])]
         [AFTER:(dd-mmm-yy  [hh:mm:ss])]
```

Backs up or restores files revised or created on, before, or after the specified date and time.

/REWIND

Rewinds the first magnetic tape of a tape set before executing a backup or restore operation.

When specified with an input tape, BRU rewinds the first tape of the tape set before searching for backup sets.

When specified with /APPEND, BRU rewinds the first set of the tape set and then searches for the logical end of the last backup set on the tape before executing the backup operation.

/SUPERSEDE

Specifies that when file specifications on the output volume are identical to file specifications on the input volume, the file on the output volume is deleted and replaced with the file from the input volume. NOSUPERSEDE is the default.

/TAPE_LABEL

Specifies the 6-character volume identifier on the ANSI VOL1 label to be placed on a tape during a backup operation or to be compared with the label on the tape on append and restore operations. This allows you to check that you have mounted the correct tape.

/VERIFY

Copies data from the input volume, performs a compare operation after the transfer, and reports any differences.

/WINDOWS:value

When initializing an output disk, specifies the default number of mapping pointers to be allocated for file windows. This value is used by an ACP when the volume is mounted. A file window consists of a number of pointers and is stored in memory when the file is opened. The default number of mapping pointers is the same as the default for the input disk.

Choosing a large number of mapping pointers may speed up file access; however, a large file window uses up system dynamic memory (pool space). If pool space is more critical than file access time, choose a smaller number of pointers.

Refer to Appendix F of the IAS/RSX-11 I/O Operations Reference Manual for further information.

10.4 USING BACKUP AND RESTORE

This section contains information on how to use the following BRU functions:

- Using the Bad Block Utility (BAD) with BACKUP and RESTORE
- Using the Format Utility (FMT) with BACKUP and RESTORE
- Using BRU to Initialize Disks
- Restoring Data Selectively

- Transferring Data Between Disks of Different Size
- BRU and File Characteristics

10.4.1 Using the Format Utility with Backup and Restore

The Format (FMT) utility places magnetic sector and timing marks on disks. In general, disks need to be formatted only once. However, you may want to run FMT to ensure that the sector and timing marks are accurate.

The format for invoking FMT is:

```
>FMT disk:
```

Where disk: is the logical name of the disk drive on which your pack resides, for example:

```
>FMT DB0:
```

10.4.2 Using the Bad Block Utility with Backup and Restore

Once the disk has been formatted, use the Bad Block (BAD) utility to locate bad blocks on the disk you are initializing.

BAD uses two types of processes for disks: one for last-track devices and the other for nonlast-track devices. Last-track devices have manufacturer-written bad block information on the last half of the last track. Nonlast-track devices have no such information.

BAD does pattern checking on the disk being formatted. That is, BAD writes a pattern on the tracks of the disk being formatted, reads the pattern from the disk, and then checks to ensure that the pattern was read correctly.

For last-track devices, BAD uses the results of this checking, along with the manufacturer-written bad block information, to create the software-detected bad sector file, which is also placed on the last track of the device.

For nonlast-track devices, BAD uses only the results of its pattern checking to create the bad block descriptor block. This block is then placed on the last good block on the disk.

BRU uses this bad block information to create BADBLK.SYS, a file occupying the bad blocks on a disk so that they cannot be used by the file system.

10.4.2.1 **BRU Bad Block Processing** - Once you have formatted the disk with the FMT utility and located the bad blocks using the BAD utility, you can run BRU to complete initialization of the disk.

If you specify /BAD with the INITIALIZE qualifier, you can use one of three options: AUTOMATIC (the default), MANUAL, or OVERRIDE. The following sections describe how to use these options.

10.4.2.2 **Using the AUTOMATIC Qualifier** - The AUTOMATIC qualifier specifies that BRU use the existing bad block information on the disk to create the file BADBLK.SYS. For last-track devices BRU uses the manufacturer-written bad block information and the software bad sector file. For nonlast-track devices BRU uses the bad block descriptor block.

10.4.2.3 **Using the OVERRIDE Qualifier** - OVERRIDE applies only to last-track devices. It makes the disk appear to be a nonlast-track device.

When you use OVERRIDE with BRU, ensure that the disk you are processing has previously been processed by the BAD utility with the BAD /OVR switch specified.

Using /OVR with BAD makes last-track devices look like nonlast-track devices by using the last good block before the last track as the bad block descriptor block. /OVR processing includes that last track as bad data when it creates the bad block descriptor block.

OVERRIDE processing for BRU assumes that the bad block descriptor block written by BAD exists on the disk being processed.

10.4.2.4 **Using the MANUAL Qualifier** - The MANUAL qualifier uses the manufacturer-written bad block information written on the last track and either the software-detected bad sector file or the bad block descriptor block to create BADBLK.SYS. In addition, it accepts the addresses of bad blocks you enter interactively at your terminal. If there is no software-written bad block information, a message will be displayed warning that BAD has not processed the disk.

When you specify /BAD:MANUAL, BRU will issue a prompt at your terminal. To enter bad blocks, respond to the prompt with the starting logical block number followed by a count of how many consecutive blocks are bad, in the following format:

```
LBN:COUNT
```

This format is compatible with both the BAD and INI utilities. BRU interprets both the LBN and the count as decimal numbers. You can specify the LBN in octal, but you must specify the count in decimal. To specify an octal value for the LBN, precede it with a #.

When you have finished entering bad blocks, type <RET> to return to BRU command level.

10.4.3 Using Backup and Restore to Initialize Disks

You can use BRU commands in conjunction with the FMT and BAD utilities to initialize (format a Files-11 structure on) a disk. The BRU initialization qualifiers provide the same capabilities as those of the INI utility. Selecting appropriate values for the initialization qualifiers requires an in-depth knowledge of Files-11. Refer to the IAS/RSX-11 I/O Operations Reference Manual for details of the Files-11 disk structure. The following example shows the steps involved in initializing a disk using BRU.

```
RUN $FMT
FMT>DK2:

** WARNING - DATA WILL BE LOST ON DK2: **

CONTINUE? [Y OR N]: Y

START FORMATTING

OPERATION COMPLETE

FMT>^Z
>BAD DK2:
BAD -- DK2: TOTAL BAD BLOCKS= 0.
>RUN BRU
BRU>/INITIALIZE/OUTVOLUME:BACKUP/HEADERS:24/MAXIMUM:45/BAD:-
BRU>AUTOMATIC/POSITION:END/WINDOWS:3
FROM:    DK1:
TO:      DK2:
BRU -- *WARNING* -- THIS DISK WILL NOT CONTAIN A HARDWARE BOOTABLE SYSTEM

BRU - COMPLETED

BRU>^Z
>
```

10.4.4 Selective Transfer of Data

BRU provides functions that allow you to transfer data selectively. That is, you can select categories of files to be backed up using file specification elements, date, and time. For example, you can specify that only files with the file type .OBJ or with UICs [1,54] be backed up; similarly, you can specify that only files created before August 4, 1978 at 2:00 PM be backed up.

10.4.5 Backup and Restore to Disks of Different Size

BRU allows you to transfer data between disks of different sizes. However, when you restore a volume containing a boot block, the new volume is not bootable if the controller for the output device is not compatible with the device controller on which the volume was created. BRU prints a warning message.

NOTE

When restoring to an existing volume, BRU will not issue the warning message.

If the new disk is smaller than the original, you may have problems with the preservation of file-IDs. In this event you should use some of the qualifiers described in Section 10.3.3 under the /INITIALIZE qualifier. If you make the index file smaller, BRU does not preserve the file-IDs and a message is printed.

If you are transferring to a larger disk than the original, you must use the /MAXIMUM and /HEADERS qualifiers to make full use of the larger disk. If you do not specify a maximum number of files or file headers, BRU uses the values of the original disk and you will end up with a large disk that can contain only a small number of files.

Once you have initialized a disk with a maximum number of files, that maximum cannot be extended. So the value for the /MAXIMUM qualifier should be carefully chosen.

Refer to the IAS/RSX-11 I/O Operations Reference Manual for a detailed description of Files-11 structure before choosing values for the initialization qualifiers.

10.4.6 BRU and File Characteristics

This section explains in detail how BRU treats the following file characteristics:

- Creation and revision dates
- File headers
- File synonyms
- Lost files

10.4.6.1 **Creation and Revision Dates of Files** - BRU always preserves the creation and revision dates of files that it transfers. However, since BRU creates UFDs during a restore operation to an unmounted volume the creation date of the UFD is the date on which BRU created it.

10.4.6.2 **File Headers** - BRU preserves all characteristics of a file, if possible. There are three exceptions:

- If there is insufficient room on the output volume to restore the file contiguously, it is restored noncontiguously.
- The file name is updated on the file's header to match the UFD entry.
- The physical end-of-file in the user attribute area is updated to reflect correctly the file's size.

10.4.6.3 **File Synonyms** - BRU copies file synonyms to tape as separate files.

If you restore files with synonyms to an unmounted volume and you preserve file ids, the file synonyms are restored as synonyms. However, if you do not preserve file ids or you restore to a mounted volume, file synonyms are restored as separate files.

10.4.6.4 **Lost Files** - A file that is not contained in any UFD is known as a lost file. BRU does not find lost files. To find lost files, use the VFY utility with the /LO qualifier before using BRU to back up the disk.

10.5 TAPE FORMAT INFORMATION FOR BACKUP AND RESTORE OPERATIONS

Magnetic tapes have ANSI standard tape label formats, as shown in Figure 10-1. The items at the beginning of the tape tell the tape drive what it is reading: BOT means beginning of tape; VOL1 means beginning of tape information; and HDR1 and HDR2 indicate the beginning of a backup set.

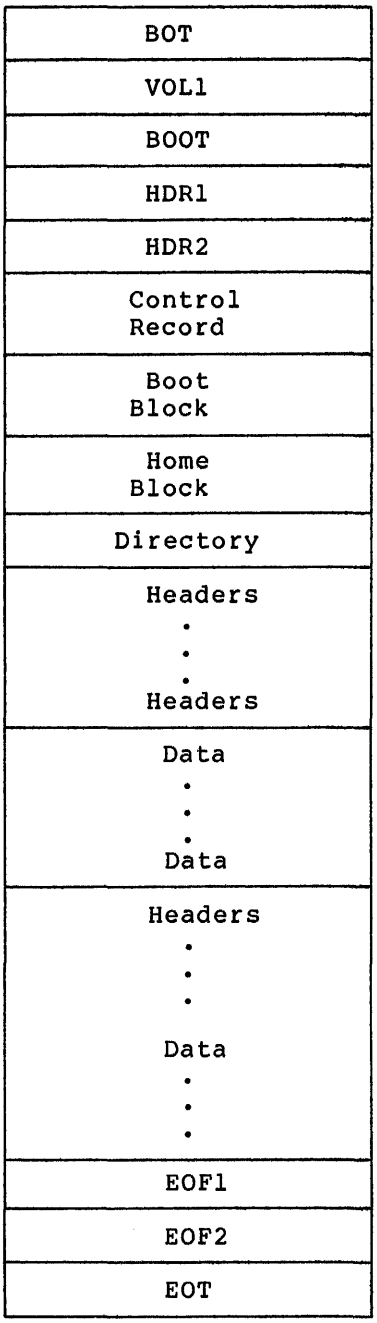


Figure 10-1 Standard Tape Format for Magnetic Tapes

10.5.1 Backup Sets

A backup set consists of all the data directed to a volume (or volumes) during a backup or restore operation. A backup set includes tape control data (one for each tape volume required), header records, and data records. Physically, more than one backup set may be contained on a tape, or a backup set can extend over several tapes.

The tape control data is written at the beginning of each backup set. It contains a control record and on the first tape of a backup set, the contents of the home block and boot block for the disk volume being backed up. Directory records are interspersed with UFD records to identify the owning UFD of the following directory records. Each entry in a directory record represents a file to be backed up and is identical to the file's entry in the UFD that contains it.

The header data consists of an identifying record, UFD records, and file header records. The UFD identifier record identifies the UFD containing the file headers following it.

Data records consist of an 80-byte data identification record indicating that the records following are data records, and 1 to 10 data blocks preceded by a 48-byte area containing file identifiers and retrieval pointers.

10.5.2 Tape Sets

More than one backup set may be contained on a tape, or a backup set can extend over several tapes. In either case the resulting output is called a tape set.

A tape set consists of the tape volume or volumes to which data is transferred during a backup operation.

10.5.3 Multivolume Tape Operations

When you specify a magnetic tape drive as the output device in a BRU operation, BRU writes the data contents of the input disk to the tape on the drive. This data transfer often involves more than one reel of tape and may utilize more than one tape drive.

You may specify multiple on-line tape drives in the following format:

```
BRU>/BACKUP_SET:23MAY79A DM0: MM0:,MM1:
BRU - STARTING TAPE 1 ON MM0:

BRU - END OF TAPE 1 ON MM0:

BRU - STARTING TAPE 2 ON MM1:

BRU - END OF TAPE 2 ON MM1:

BRU - COMPLETED
```

You can specify only one type of drive, either MM or MT, in a single BRU command. Although you can specify up to eight drives per command, you can specify an individual tape drive only once.

If the number of tape volumes required exceeds the number of tape drives available, BRU lets you replace tapes on the specified drive in round-robin fashion.

10.6 BACKUP AND RESTORE ERROR PROCESSING

This section lists BRU error messages, describes the meaning of the message, and suggests actions to correct the errors.

BRU -- *FATAL* -- ALLOCATION FOR SYS FILE EXCEEDS VOLUME LIMIT

Explanation: A system file (one of the following files: INDEXF.SYS, BITMAP.SYS, BADBLK.SYS, 000000.DIR) requires more space than is available on the output disk. This will usually occur if the output disk is smaller than the input disk.

User Action: You can use the POSITION qualifier to force allocation to start at the beginning of the disk and/or use the MAXIMUM and HEADERS qualifiers to reduce the size of INDEXF.SYS.

BRU -- *FATAL* -- AMBIGUOUS KEYWORD

Explanation: A keyword specified with a qualifier is not unique. For example /POSITION:B could mean either BEGINNING or BLOCK.

User Action: You can abbreviate keywords to the shortest unique abbreviation.

BRU -- *FATAL* -- AMBIGUOUS QUALIFIER

Explanation: A qualifier is not unique. For example, /RE could mean either REVISED or REWIND.

User Action: You can abbreviate qualifiers to the shortest unique abbreviation.

BRU -- *WARNING* -- APPENDING AT 800 BPI on device:
or
BRU -- *WARNING* -- APPENDING AT 1600 BPI on device:

Explanation: The wrong tape density was specified with the /APPEND switch. BRU will perform an APPEND only at the density at which the tape is written.

User Action: None. BRU will continue at the correct density.

BRU -- *FATAL* -- ATTACH FAILED on device:

Explanation: BRU could not attach the specified device.

User Action: Check to see if another task has the device attached, or if the device has a volume mounted on it.

BRU -- *WARNING* -- BAD BLOCK DATA ERROR

Explanation: A manually entered bad block location, count, or syntax was incorrect.

User Action: Enter the correct information.

BRU -- *WARNING* -- BAD BLOCK FILE CORRUPT. DATA IGNORED

Explanation: The bad block file created by BAD contains invalid data. Unless you enter the bad blocks manually, BRU will assume there are no bad blocks.

User Action: Use the /OVERRIDE qualifier to ignore the file.

BRU -- *WARNING* -- BAD BLOCK FILE FULL

Explanation: The manual addition of bad blocks has resulted in more than 102 sets of contiguous bad blocks.

User Action: None. You cannot enter more bad blocks than the file will hold.

BRU -- *WARNING* -- BLOCK EXCEEDS VOLUME SIZE

Explanation: You have manually entered a bad block that is larger than the size of the output disk.

User Action: Enter the correct block.

BRU -- *WARNING* -- BOOT BLOCK IS BAD

Explanation: BRU cannot write to the output boot block. Therefore the disk will not be hardware bootable.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- BOOT BLOCK IS CORRUPT

Explanation: The input disk does not contain a valid boot block. The output disk will not be hardware bootable.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- BOOT BLOCK READ ERROR

Explanation: An error occurred while BRU was reading the boot block.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- BOOT BLOCK VERIFY ERROR on device:

Explanation: During a backup operation, the boot block on the output device did not match the boot block on the input device.

User Action: None. BRU will continue with the operation.

BRU -- *WARNING* -- CANNOT RESTORE CONTIGUOUSLY

Explanation: The output device does not contain enough contiguous blocks to restore the indicated contiguous file. The file will be restored noncontiguously.

User Action: You can use the PIP utility to make the file contiguous again. Use the PIP qualifiers /DE and /TR to reclaim disk space by deletion or truncation.

BRU -- COMPLETED

Explanation: The BRU operation is complete.

User Action: Enter another BRU command or exit.

BRU -- *FATAL* -- CONFLICTING QUALIFIERS

Explanation: Two or more of the specified qualifiers are mutually exclusive: for example, /SUPERSEDE and /NOSUPERSEDE.

User Action: Retype the command line.

BRU -- *WARNING* -- DATA ID RECORD VERIFY ERROR

Explanation: There was an error while BRU was verifying an 80-byte DATA ID record.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- DATA RECORD VERIFY ERROR [UFD]filename.type;version
FILE ID number LBN number

Explanation: There was a difference in a data block on input and output devices. The file id of the file with the error and the lbn of the block follow the message.

If a [UFD] is printed with a file name, the UFD is the owner UFD from the file's header, not the UFD in which the file is contained.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- DATA WAS LOST DUE TO IO ERRORS [UFD]
filename.type;version

Explanation: A tape read error resulted in missed data. The files will be restored but may contain erroneous data.

If a [UFD] is printed with a file name, the UFD is the owner UFD from the file's header, not the UFD in which the file is contained.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- DEVICE CONFLICT

Explanation: Both a tape and a disk drive were specified as part of the input or output device specification.

User Action: The device must be either a disk or a tape but not both. This applies to both input and output specifications.

BRU -- *FATAL* -- DEVICE NOT IN SYSTEM

Explanation: A device was specified that does not exist on the system.

User Action: Correct the device specification.

BRU -- *FATAL* -- DEVICE NOT SUPPORTED

Explanation: The specified device was not a tape or a disk.

User Action: BRU supports only magnetic tape and disk devices.

BRU -- *FATAL* -- DIRECTIVE ERROR

Explanation: An internal error has occurred in BRU.

User Action: Retry the operation.

BRU -- *WARNING* -- DIRECTORY VERIFY ERROR

Explanation: There was a difference in a directory record on input and output devices.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- DISK IS AN ALIGNMENT CARTRIDGE

Explanation: The last track identified the disk as an alignment cartridge, which cannot be initialized as a Files-11 volume.

User Action: BRU can be used only on Files-11 volumes.

BRU -- *FATAL* -- DISK READ ERROR

Explanation: An unrecoverable read error occurred on the output disk. This could be caused by an undetected bad block.

User Action: Use the BAD utility to locate all bad blocks. Then use BRU with the /BAD:AUTOMATIC qualifier.

BRU -- *FATAL* -- DISK WRITE ERROR

Explanation: An unrecoverable write error occurred on the output disk. This could be caused by an undetected bad block.

User Action: Use the BAD utility to locate all bad blocks. Then use BRU with the /BAD:AUTOMATIC qualifier.

BRU -- *FATAL* -- DOUBLY DEFINED QUALIFIER

Explanation: The same qualifier was specified more than once on the command line.

User Action: Retype the command line correctly.

BRU -- *WARNING* -- DUPLICATE BLOCKS FOUND

Explanation: A manually entered bad block was already in the bad block file.

User Action: None. BRU will continue the operation.

BRU -- END OF tape number ON device:

Explanation: BRU has finished transferring data or verifying a tape.

User Action: None. This is an informational message.

BRU -- *FATAL* -- END OF VOLUME ENCOUNTERED. BACKUP SET NOT FOUND

Explanation: The backup set specified for a restore operation is not on the tape volume.

User Action: Mount the correct tape volume and retry the operation.

BRU -- *FATAL* -- ERROR READING COMMAND FILE

Explanation: An I/O error occurred while BRU was reading the indirect command file.

User Action: Retry the operation.

BRU -- *WARNING* -- ERROR READING DATA BLOCKS
I/O ERROR CODE-number
FILE ID number LBN number
or
RECOVERED

Explanation: An I/O error occurred while BRU was reading a data block from the disk. The file id of the file which contains the block and the lbn of the block are displayed as well as the I/O error code. If RECOVERED is printed after the message, the block was recovered by issuing 1-block reads from the disk.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- ERROR READING UFD [group,number]

Explanation: An I/O error occurred while BRU was reading a block from the specified UFD. Any files contained in this block of the UFD will not be backed up.

User Action: You may retry the operation. If the error still occurs, you can find the files by using the VFY utiltity.

BRU -- *WARNING* -- ERROR READING UFD HEADER [group,number]

Explanation: An error occurred while BRU was reading the header of the specified UFD. Files in this UFD will not be backed up.

User Action: Retry the operation. If the error still occurs, use the VFY utility to find the file.

BRU -- *WARNING* -- EOT MARKER ERROR

Explanation: An error occurred while BRU was writing or verifying the end of tape labels on the output tape.

User Action: On a write error, BRU will rewind the current tape and put it off-line. BRU will request that a new tape be mounted and will rewrite the tape.

On a verify error, BRU will continue.

BRU -- *FATAL* -- ERROR LIMIT EXCEEDED

Explanation: BRU has reached the specified number of tape read errors and terminated execution.

User Action: Retry the operation, using a different tape drive, or after cleaning the tape drive heads.

BRU -- *WARNING* -- EXTENDING INDEX FILE

Explanation: The initial number of file headers was too small. Either 256 or 16 more headers will be allocated depending on the number of blocks on the output disk.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- FAILED TO READ BAD BLOCK FILE

Explanation: BRU was unable to read the bad block information from a last-track output disk.

User Action: Retry the operation, using the /BAD:OVERRIDE qualifier.

BRU -- *WARNING* -- FILE HEADER READ ERROR [group,number] filename.type;version I/O ERROR CODE-number

Explanation: An I/O error occurred while BRU was reading a file header. That file will not be backed up.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- FILE HEADER VERIFY ERROR [group,number] filename.type;version

Explanation: The file header of the specified file on the output device is not the same as that on the input device.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- FILE ID AREA VERIFY ERROR

Explanation: The BRU-generated file id area of a data record was different on the input and output devices.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- FILE ID EXCEEDS MAXIMUM NUMBER OF FILES

Explanation: You specified a maximum number of files with the MAXIMUM qualifier, which was too small to perserve file ids.

User Action: Specify a larger value with the MAXIMUM qualifier, or use a larger output disk.

BRU -- *WARNING* -- FILE ID SEQUENCE NUMBER ERROR [UFD] filename.type;version

Explanation: There are two possible sources of this error:

1. The sequence number in the file id of a file does not match the sequence number of the file's entry in the UFD.

2. The sequence number of a UFD does not match the sequence number of the UFD's entry in the MFD.

Therefore, the file or UFD is not valid and will not be copied.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- FILE IDS WILL NOT BE PRESERVED

Explanation: File ids cannot be preserved because the bitmap on the output disk is too small. This is because the value specified with the MAXIMUM qualifier was too small.

User Action: None. BRU will continue the operation, not preserving file ids.

BRU -- *WARNING* -- FILE MARKED FOR DELETE [UFD] filename.type;version

Explanation: The marked-for-delete bit (SC.MDL) of the system controlled characteristics in the file header was set, indicating that the file was partially deleted. The file will not be copied.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- FILE NOT FOUND [UFD] filename.type;version

Explanation: During a backup operation, the specified file does not have a header in the index file. The file is not copied.

During a restore operation, BRU cannot find the file being verified on the output disk.

User Action: BRU will continue the operation.

BRU -- *FATAL* -- HANDLER NOT RESIDENT

Explanation: The device driver for the specified device is not loaded.

User Action: Load the driver for the specified device, or retype the command line specifying the correct device name.

BRU -- *WARNING* -- HEADER ID RECORD VERIFY ERROR

Explanation: The header id record on the output device is different from the one on the input device.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- HEADER READ ERROR [UFD]filename.type;version
I/O ERROR CODE-number

Explanation: An I/O error occurred while BRU was reading a file header in the index file during a backup operation.

If this error occurs during a restore operation, it is fatal.

User Action: None.

BRU -- *FATAL* -- HOME BLOCK READ ERROR
I/O ERROR CODE-number

Explanation: An I/O error occurred while BRU was reading the home block on the input device.

User Action: Retry the operation.

BRU -- *WARNING* -- HOME BLOCK VERIFY ERROR

Explanation: The home block on the output device is different from the home block on the input device.

User Action: BRU will continue, but it is suggested that you retry the operation.

BRU -- *FATAL* -- HOME BLOCK WRITE ERROR

Explanation: An unrecoverable I/O error occurred while BRU was writing the home block on the output device.

User Action: Use BAD.

BRU -- *FATAL* -- ILLEGAL USE OF DIRECTORY QUALIFIER

Explanation:

1. The DIRECTORY qualifier was specified with an output device.
2. The DIRECTORY qualifier was specified with a device other than tape.
3. Initialization qualifiers were specified with the directory qualifiers.

User Action: Refer to Section 10.3.3 for a description of valid uses of the DIRECTORY qualifier.

BRU -- *FATAL* -- INCONSISTENT INITIALIZE QUALIFIERS

Explanation: Qualifiers that initialize the output disk were specified, but the NOINITIALIZE qualifier was used.

User Action: Retype the command line.

BRU -- *FATAL* -- INDEX FILE HEADER READ ERROR
I/O ERROR CODE-number

Explanation: An I/O error occurred while BRU was reading the header of the index file on the input disk.

User Action: Retry the operation.

BRU -- *FATAL* -- INDEX FILE WRITE ERROR

Explanation: An unrecoverable I/O error occurred while BRU was writing the index file to the output disk.

User Action: Use BAD to identify the bad blocks on the output disk. Then initialize the disk and retry the operation.

BRU -- *FATAL* -- INDEXF.SYS IS FULL

Explanation: The index file cannot map any more file headers.

User Action: Retry the operation, using a larger value with the /MAXIMUM qualifier.

BRU -- *FATAL* -- INIT QUALIFIERS INVALID WHEN OUTPUT IS TAPE

Explanation: The initialize qualifiers may be used only when the output device is a disk.

User Action: Retype the command line.

BRU -- *FATAL* -- INPUT DEVICE EQUALS OUTPUT DEVICE

Explanation: The input and output devices must be different.

User Action: Retype the command line, specifying different devices for input and output.

BRU -- *FATAL* -- INPUT LINE TOO LONG

Explanation: The maximum length of a command line is 256 characters.

User Action: You may abbreviate qualifiers and keywords as long as the abbreviations are unambiguous.

BRU -- *FATAL* -- INTERNAL ERROR IN BRU

Explanation: BRU has detected an error within itself. This should not normally occur.

User Action: Please submit an SPR with all pertinent information.

BRU -- *FATAL* -- INVALID DATE

Explanation: A date was specified incorrectly or is out of range.

User Action: Specify the correct date.

BRU -- *FATAL* -- INVALID DENSITY

Explanation: A density was specified that was neither 800 nor 1600 BPI.

User Action: Retype the command line, specifying the correct density.

BRU -- *FATAL* -- INVALID FILENAME

Explanation: The name of the indirect command file is not syntactically correct.

User Action: Retype the command line.

BRU -- *WARNING* -- INVALID TAPE FORMAT

Explanation: An invalid tape record was read during a restore operation.

User Action: None. The invalid record will not be restored.

BRU -- *FATAL* -- INVALID VALUE

Explanation: A value specified for a qualifier has illegal syntax or is out of range.

User Action: Refer to Section 10.3.3 to determine the valid values for the particular qualifier.

BRU -- *WARNING* -- INVALID DATE ON FILE HEADER [UFD] filename.type;version

Explanation: An invalid date was encountered on a file header during an incremental backup. The file is copied.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- MANUFACTURER BAD BLOCK SECTOR FILE IS CORRUPT

Explanation: BRU was unable to read the bad block information from a last track output disk.

User Action: Retry the operation, using the /BAD:OVERRIDE qualifier.

BRU -- *FATAL* -- MFD HEADER READ ERROR

Explanation: An I/O error occurred while BRU was reading the header of the Master File Directory.

User Action: Retry the operation. If the header still cannot be read, the files on the disk are lost and may be recovered using the VFY utility.

BRU -- *WARNING* -- MFD READ ERROR

Explanation: An I/O error occurred while BRU was reading a block of the MFD. BRU cannot copy the UFDs in that block of the MFD.

User Action: Retry the operation. If the block cannot be read, use the VFY utility to recover the lost files.

BRU -- *FATAL* -- MISSING COLON

Explanation: A qualifier keyword that accepts a value was not followed by a colon.

User Action: Retype the command line.

BRU -- *FATAL* -- MORE THAN 1 LEVEL OF INDIRECTION

Explanation: BRU does not support more than one level of indirect command files.

User Action: Retype the command line.

BRU -- MOUNT TAPE n ON ddn:

Explanation: There is no tape on the specified drive or the tape is not at load point. This message will print every two minutes until the tape is mounted.

User Action: Mount the tape specified on the drive specified.

BRU -- MOUNT ANOTHER TAPE

Explanation: BRU is requesting that a new tape be mounted after encountering a fatal tape write error.

User Action: Mount a new tape on the drive.

BRU -- *FATAL* -- NAME MORE THAN 12 CHARACTERS

Explanation: A name, such as a backup set name, is longer than 12 characters.

User Action: Specify a name not greater than 12 characters.

BRU -- *WARNING* -- NO BAD BLOCK DATA FILE FOUND

Explanation: The BAD utility has not been run on the output disk to produce a file of the disk's bad blocks.

User Action: None. BRU will continue the operation. Refer to Section 10.4 for information on bad block processing by BRU.

BRU -- *WARNING* -- NO FILES FOUND [UIC] filename.type;version

Explanation: During a backup operation, BRU found an entry in the UFD or MFD that had no corresponding header in the index file.

During a restore, compare, or verify operation, BRU cannot find the specified file on the output device.

User Action: BRU ignores the entry.

BRU -- *WARNING* -- NON FATAL QUALIFIER CONFLICTS BEING IGNORED

Explanation: You entered a qualifier that conflicts with the rest of the command line and is nonfatal if ignored. For example, /REWIND on a disk-to-disk operation.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- NON OCTAL NUMBER

Explanation: A decimal number was used where an octal number was expected.

User Action: Retry the operation, using the correct octal number.

BRU -- *FATAL* -- NUMBER OF HEADERS EXCEED MAXIMUM NUMBER OF FILES

Explanation: The number of initial file headers must be at least five less than the maximum number of files in order to account for the system files.

User Action: Use the /HEADERS qualifier to specify the number of initial file headers.

BRU -- *FATAL* -- OUTPUT DISK TOO FRAGMENTED TO RESTORE

Explanation: Internal tables in BRU have overflowed due to extreme fragmentation of the output disk. If the output disk was initialized, then it has an unacceptable number of bad blocks and should not be used as a backup medium.

User Action: Use a new disk as the output device.

BRU -- *FATAL* -- OUTPUT VOLUME IS FULL

Explanation: There are no free blocks on the output disk. This can occur when the output disk is smaller than the input disk, or during an append to a tape that is already full.

User Action: If the output disk is too small, do an incremental backup of only the files you want. If you were doing an append to a tape that is already full, repeat the operation using a new tape.

BRU -- *FATAL* -- OVERRIDE INVALID WITH NON LAST TRACK DEVICE

Explanation: The OVERRIDE keyword may be used only when the output disk is a last-track device.

User Action: Refer to Section 10.4.

BRU -- PLEASE ANSWER YES OR NO

Explanation: BRU requires a response.

User Action: Enter YES or NO at your terminal.

BRU -- *FATAL* -- PRIVILEGE VIOLATION

Explanation: The input volume is mounted as a Files-11 structure but the /MOUNTED qualifier was not entered on the command line.

In copying from a mounted volume, the user did not have read access to the specified UFD or file.

User Action: Either dismount the disk or specify the /MOUNTED qualifier on the command line.

BRU -- *FATAL* -- RAN OUT OF SPARE FILE IDS

Explanation: The output disk is too fragmented to preserve file ids.

User Action: Retry the operation using the /NOPRESERVE qualifier. You may want to use a different output disk.

BRU -- *WARNING* -- RECORD NOT EXPECTED SIZE

Explanation: The record read on the output device during a verify or compare operation was not the expected size.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- REQUIRED INPUT DEVICE MISSING

Explanation: The input device was not specified on the command line or in response to the prompt.

User Action: Retype the command line.

BRU -- *FATAL* -- REQUIRED OUTPUT DEVICE MISSING

Explanation: The output device was not specified on the command line or in response to the prompt.

User Action: Retype the command line.

BRU -- REWIND ERROR ON ddn:

Explanation: An I/O error occurred during a tape rewind. This error is fatal if it occurs on the first tape of a tape set or during a rewind for verify. The error is not fatal if BRU is rewinding a tape it is finished with.

User Action: None.

BRU -- *FATAL* -- SEARCH FOR HOME BLOCK FAILED

Explanation: The home block could not be found on the input disk. Either the home block is bad or the disk is not Files-11.

User Action: Check to see that you have the correct disk.

BRU -- *FATAL* -- STACK OVERFLOW IN SORT ROUTINE

Explanation: A stack internal to BRU has overflowed. This should not normally occur.

User Action: Please submit an SPR with all pertinent information.

BRU -- STARTING TAPE n ON ddn:

Explanation: This message tells you which tape is being copied to on which drive.

User Action: None. This is an informational message.

BRU -- STARTING VERIFY PASS

Explanation: This message tells you which tape is being verified.

User Action: None. This is an informational message.

BRU -- *FATAL* -- SYNTAX ERROR

Explanation: The command line is invalid.

User Action: Retype the command line.

BRU -- TAPE LABEL ERROR ON ddn:
I/O ERROR CODE -number

Explanation: An I/O error occurred while BRU was reading or writing a tape label. A write error is fatal; a read error is not fatal as long as BRU can continue reading the tape.

User Action: If a write error occurred, retry the operation using a different tape.

BRU -- *WARNING* -- TAPE LABEL VERIFY ERROR

Explanation: A difference was found between the input and output on a tape label.

User Action: None. BRU will continue the operation.

BRU -- *WARNING* -- TAPE NOT AT BOT. NO REWIND OR APPEND SPECIFIED

Explanation: For a backup operation to tape, BRU will not start processing a tape that is not a BOT unless the /APPEND qualifier was specified for BRU.

User Action: Use the /APPEND qualifier if you want to start writing at the current tape position. Use the /REWIND qualifier to position the tape to BOT.

BRU -- *WARNING* -- TAPE OUT OF SEQUENCE. PLEASE MOUNT CORRECT TAPE

Explanation: The wrong tape volume was mounted on the tape drive during a restore from tape operation.

User Action: Mount the correct tape on the drive.

BRU -- *WARNING* -- TAPE POSITIONING ERROR. BACKSPACE FAILED

Explanation: During a backup operation, the tape was not positioned properly for a future append operation.

User Action: Rewind the tape before attempting to append.

BRU -- *FATAL* -- TAPE POSITIONING ERROR. NO EOV ENCOUNTERED
I/O ERROR CODE-number

Explanation: The space files operation to find the end-of-volume for an append operation failed.

User Action: Retry the operation.

BRU -- *WARNING* -- TAPE READ ERROR

Explanation: An I/O error occurred while reading a tape.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- TAPE TO TAPE NOT SUPPORTED

Explanation: BRU will not back up a tape to another tape.

User Action: None.

BRU -- *WARNING* -- TAPE WRITE ERROR
I/O ERROR CODE-number

Explanation: An I/O error occurred while writing to tape. BRU will rewind the tape and request that another tape be mounted.

User Action: If the error is related to the tape drive, terminate BRU and start over on another drive.

BRU -- *WARNING* -- THIS DISK WILL NOT CONTAIN A HARDWARE BOOTABLE SYSTEM

Explanation: The output disk will not be hardware bootable. This can be caused by:

1. The input disk not being bootable
2. The system image not being restored
3. Restoring to a disk of different size or type

NOTE

This warning will not be issued when restoring to an existing volume.

User Action: None.

BRU -- *FATAL* -- TOO MANY DEVICES

Explanation: As an input or output device a disk may be specified only once. However, up to eight tape drives may constitute the input or output.

User Action: Retype the command line specifying only one disk or no more than eight tape drives.

BRU -- *FATAL* -- TOO MANY FILE SPECIFICATIONS

Explanation: More than 16 file specifications were specified on the command line.

User Action: Retype the command line. You may use wildcards to reduce the number of file specifications on the command line.

BRU -- *FATAL* -- UFD OR MFD REQUIRES UNSUPPORTED EXTENSION HEADERS

Explanation: BRU does not support extension headers for MFDs or UFDs.

User Action: This error should not occur. Please submit an SPR with all pertinent information.

BRU -- *WARNING* -- UFD RECORD VERIFY ERROR

Explanation: There is a difference between input and output devices on a UFD record.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- UNKNOWN KEYWORD

Explanation: A keyword was specified that was not recognized by BRU.

User Action: Retype the command line. See Table 10-1 for a list of valid keywords.

BRU -- *FATAL* -- UNKNOWN QUALIFIER

Explanation: A qualifier was specified that was not recognized by BRU.

User Action: Retype the command line. See Table 10-1 for a list of valid qualifiers.

BRU -- *FATAL* -- UNSUPPORTED STRUCTURE LEVEL

Explanation: The file structure level on the input disk is not supported by BRU.

User Action: Check to see that you have the correct disk.

BRU -- *WARNING* -- VBN NOT IN FILE

Explanation: A file id was encountered which is larger than the maximum file id in the index file. The file is ignored. This error message will occur if a UFD entry was corrupted on the input disk.

User Action: None. BRU will continue the operation.

BRU -- *FATAL* -- VERIFY LOST

Explanation: During the verify pass of a disk to tape backup, BRU has lost synchronization between the input and the output. This is usually caused by tape position being lost or by backing up from a disk that is mounted and being changed during the backup operation.

User Action: Retry the operation.

BRU -- *FATAL* -- VOLUME NOT A BACKUP TAPE

Explanation: The tape mounted for an append or restore operation was not generated by BRU.

User Action: Check to see that you have the correct tape.

BRU -- *FATAL* -- VOLUME NOT READY

Explanation: The device is not on-line.

User Action: Put the device on-line and retype the command line.

BRU -- *FATAL* -- VOLUME WRITE LOCKED

Explanation: The output device is not write enabled.

User Action: If the output device is a tape, insert a write ring. If it is a disk, press the write enable switch on the disk drive.

BRU -- *FATAL* -- WRONG BACKUP SET

Explanation: The backup set name specified with the /BACKUPSET qualifier does not match the backup set name on the tape.

User Action: Retry the operation with the correct backup set name.

BRU -- *FATAL* -- WRONG INPUT VOLUME LABEL

Explanation: The input volume label specified with the /INPUTVOLUME qualifier does not match the volume label of the input device.

User Action: Retry the operation with the correct input volume label.

CHAPTER 11

DISK SAVE AND COMPRESS (DSC)

The Disk Save and Compress (DSC) utility program copies entire Files-11 disk files either to disk or to tape, and from DSC-created tape back onto disk. At the same time, DSC reallocates and consolidates the disk data storage area: it concatenates files and their extensions into contiguous blocks whenever possible, and may reduce the number of retrieval pointers and file headers required for the same files on the new disk volume.

DSC copies files that are randomly scattered over a disk volume to a new volume, without the intervening spaces. This eliminates unused space between files and cuts down on the time required to access them.

A complete DSC operation is a cycle that begins with data on one disk and ends with the same data on another disk, in compressed form. The operation can use one command (for a direct disk-to-disk transcription) or two commands (for a disk-to-tape and tape-to-disk transcription).

After a DSC copy operation, individual files are written in contiguous blocks whenever possible, and the blocks available for new files are located in a contiguous area at the end of the new volume.

DSC reads and writes data to two of its own buffers when it performs copy or compare operations. (See Figures 11-1 and 11-2.) Each buffer normally is large enough to contain four disk blocks and a 16-byte buffer prefix. However, the block factor switch (BL) in a DSC command line allows you to increase the number of blocks in each buffer, up to the maximum space available for DSC on your system.

In a disk-to-disk copy operation, DSC:

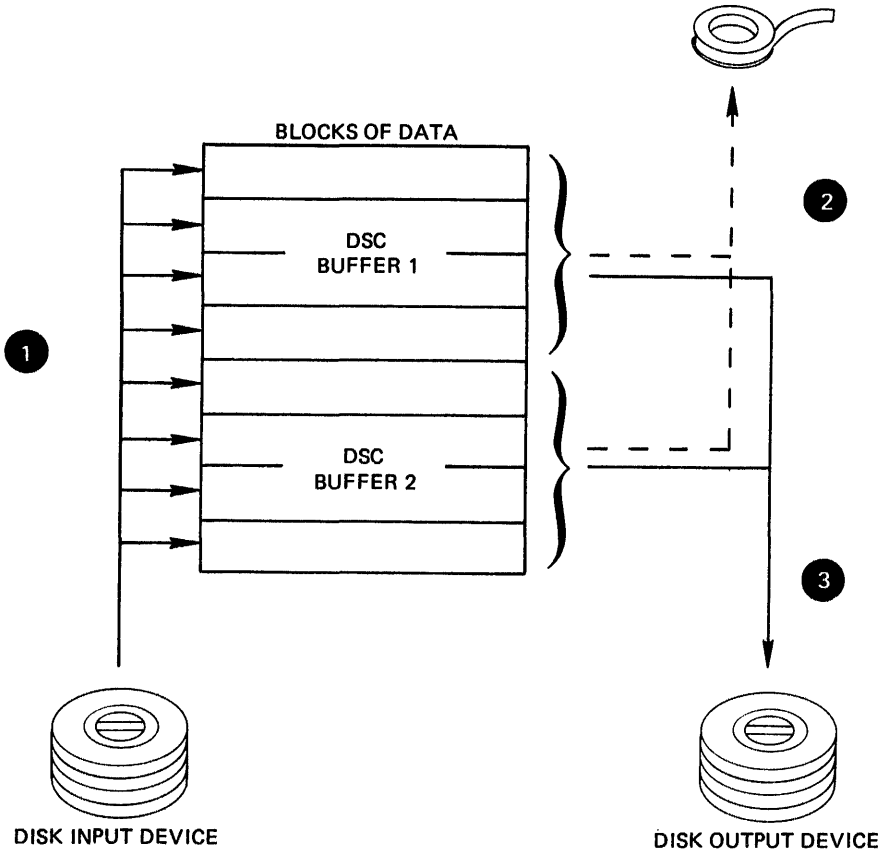
1. Copies data from disk to a DSC buffer
2. Copies data from the DSC buffer to another disk

In a disk-to-tape/tape-to-disk operation

1. Copies data from disk to a DSC buffer
2. Writes data from the DSC buffer to tape
3. Copies data from tape to a DSC buffer
4. Writes data from the DSC buffer to another disk

You can execute operations 3 and 4, to restore data to disk, at any time, as long as the data on the tape is intact.

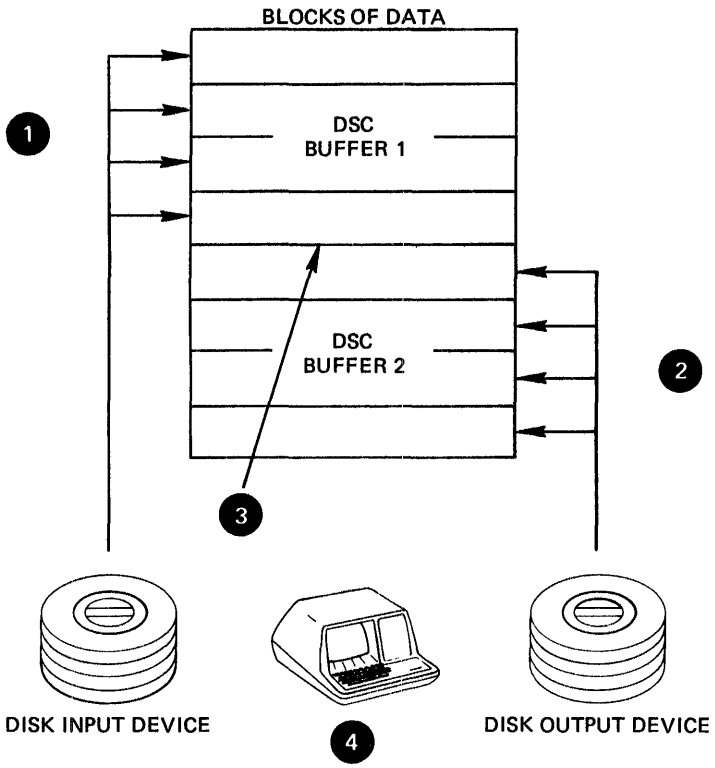
If the contents of one disk are transferred to a disk with a larger capacity, the new disk takes on the attributes of the original disk, except that additional storage space is available.



1. DSC reads 8 (default) or more blocks of data from the disk input device to two buffers.
2. In disk-to-tape copy operations, DSC writes data from the buffers to magnetic tape.
3. In disk-to-disk copy operations, DSC writes data from the buffers to the disk output device.

DSC repeats step 1 and 2 or 3 until it copies the entire input device.

Figure 11-1 Data Transfer for DSC Copy Operation



1. DSC reads 4 blocks of data from the disk input device to a buffer.
2. DSC reads 4 blocks of data from the disk output device to the second buffer.
3. DSC compares the contents of the two buffers.
4. DSC prints the differences on your terminal.

DSC repeats steps 1 through 4 until it has compared the entire device.

Figure 11-2 Data Transfer for DSC Compare Operation

After a disk-to-disk copy operation, you can access the contents of the file on the new disk directly. However, after a disk-to-tape operation you cannot access the data on tape directly, because it is stored in a format recognizable only to DSC. To access this data, you must perform a second copy operation and transfer the data to another disk volume.

When DSC copies and compresses a disk containing a saved system (a task image file created from an RSX-11M or RSX-11M-PLUS system image by an MCR SAVE command), it moves all task files to different physical addresses. However, because the Task Control Block (TCB) entries for each task contain file identifications rather than Logical Block Numbers (LBNs), a system can function normally after a SAVE when it is rebooted. The file identifications will not change.

NOTE

You can copy bootable system disks created on RSX-11M versions 3.0 and higher, IAS versions 2.0 and higher, and RSX-11M-PLUS. These disks will retain their bootable characteristics. However, bootable system disks created on earlier versions of the operating systems should not be copied using DSC.

You can also use DSC to recover from hardware malfunctions which have made a portion of a disk volume unreadable. If the contents of a block allocated to a data file cause a read error, DSC copies the garbled contents to the output device and generates a warning message labeling the garbled data block. You can then access the block and correct its contents.

11.1 DSC-SUPPORTED VOLUMES

You can use DSC with a variety of mass storage devices. The status DSC requires for the devices varies with the operating system. DSC can only access unmounted volumes on RSX-11M systems. However, IAS and RSX-11M-PLUS systems require volumes to be mounted with foreign characteristics. Table 11-1 lists the devices that can be used with DSC operations.

Table 11-1
DSC-Supported Devices

Abbreviation	Type	Class
DD*	TU58 cassette (DECtape II)	Block structured
DM	RK611/RK06/RK07 cartridge disk	Block structured
DB	RH11/RP04/RP05/RP06 and RH70 RP04/RP05/RP06 disk pack	Block structured
DK	RK11/RK05/RK05F cartridge disk	Block structured
DF*	RF11/RS11 fixed head disk	Block structured
DL	RL11/RL01/RL02 cartridge disk	Block structured
DP	RP11/RP02/RP03 disk pack	Block structured
DR	RH11/RH70/RM03/RM02/RP07 disk pack	Block structured
DS*	RH11/RS03/RS04 and RH70/RS03 RS04 fixed-head disk	Block structured
DT*	TC11/TU56 DECtape	Block structured
DX*	RX11/RX01 floppy diskette	Block structured
DY*	RX02 floppy diskette	Block Structured
MM	RH11/TM02-03/TE16/TU16/TU45/TU77 and RH70/TM02-03/TE16/TU16/ TU45 and TU77 9-track magnetic tape	Tape
MT	TM11/TU10/TE10 7- or 9-track magnetic tape and TMA11/ TS03 9-track magnetic tape	Tape

* Indicates that the device cannot be used with stand-alone DSC.

Table 11-2 lists the status required for devices used by online DSC with each operating system.

Table 11-2
DSC Device Status

Operating System	Device	Device status required for DSC operation
RSX-11M-PLUS	Disk	Mounted with foreign characteristics
	Tape	Mounted with foreign characteristics
RSX-11M V3.2	Disk	Unmounted
	Tape	Unmounted
IAS	System disk	Mounted with DCF characteristics
	All other volumes	Mounted with foreign characteristics

11.2 INITIATING AND TERMINATING ONLINE DSC

You can initiate the online version of DSC in any of the ways explained in Chapter 1 of this manual. To terminate online DSC, type (CTRL/Z).

11.3 INITIATING AND TERMINATING STAND-ALONE DSC

You can bootstrap a stand-alone version of DSC (DSCS8.SYS) from disk or from tapes supplied with the operating system. The exact method depends on which operating system you are using.

Under either RSX-11M or RSX-11M-PLUS, you can bootstrap stand-alone DSC in one of two ways:

1. Enter the privileged MCR BOOT command as follows:

```
BOO[T] [1,50]DSCS8
```

   This command bootstraps DSCS8.SYS from the disk.

2. Mount a volume containing stand-alone DSC and load the appropriate beginning bootstrap address.

Under IAS, you can invoke stand-alone DSC by hardware bootstrapping a volume containing it.

When stand-alone DSC is booted, it displays the message:

```
RSX-11S V2.2 BL26 DISK SAVE AND COMPRESS UTILITY V3.0
DSCS8>
```

The prompt indicates that DSC is ready to accept commands.

Terminate stand-alone DSC by halting the processor.

11.4 DSC COMMAND FORMAT

Commands for DSC use the format:

```
DSC>outdev[s]:[filelabel][/switch]=indev[s]:[filelabel][/switch]
```

The parameters of this command format are:

Output Parameters

outdev:

The physical volume(s) to which data is copied. The format for outdev: is AA[nn]: where AA are the ASCII characters for the volume abbreviation, [nn] is an optional 1- or 2-digit octal unit number for the volume, and the colon (:) is a line terminator. If you omit the volume number, 0 is the default. DSC uses tape drives, in the order specified in the command line. If more tapes are required than specified, DSC accesses the tapes drives available in round-robin fashion.

DSC ignores multiple disk specifications. However, up to eight tape drives, separated by commas, can be specified as output devices in an online DSC operation. Stand-alone DSC permits only two tapes to be used as output devices.

file label

Identifies the disk volume ID or tape file or tape set DSC creates in a data transfer. You can specify a file label with either disk or tape output volumes. If you do not specify a file label when you create a tape or disk with DSC, the input disk volume label becomes the output disk volume label or tape file label.

switch

One or more of the optional switches described in Section 11.5.

Input Parameters

indev:

The physical volume(s), in the format AA[nn]:, from which data is copied (see outdev: above).

file label

Identifies the DSC-created tape file that is being transferred to disk or compared. If you do not specify a file label, DSC transfers the first file it encounters after its current position on the tape. DSC ignores the specification of an input file label when the input volume is a disk.

NOTE

Each file on a DSC-format tape set contains the entire contents of the disk it backed up.

switch

One or more of the optional switches described in Section 11.5.

11.5 DSC FILE LABELS, SWITCHES, AND OPTIONS

DSC commands can contain file labels and switches. Some switches also use options to specify values. Table 11-3 summarizes the DSC switches and options. Note that all of these switches can be used with both online and stand-alone DSC. See Table 11-6 for switches available only for stand-alone DSC.

Table 11-3
DSC Switches and Options

Switch	Format	Description
Append	/AP	Appends a DSC file to the first volume of a magnetic tape set that already contains a DSC file which is currently the last file of the set.
Bad Block	/BAD= MAN NOAUTO MAN:NOAUTO	Allows manual entry of bad block locations; can supplement, override, or ignore the disk's own bad block file.
Block Factor	/BL	Sets the number of 256 word blocks DSC can include in each of its buffers.
Compare	/CMP	Compares input and output volumes for differences.
Density	/DENS=1600 /DENS=800:1600	Overrides the DSC default storage density for magnetic tapes of 800 bits per inch. DENS=1600 creates magnetic tapes at 1600 bits per inch density and 800:1600 (the split density switch) creates tapes with volume headers at 800 bits per inch and the rest of the tape at 1600 bits per inch. Note that the DENS=1600 switch is valid with TU16, TE16, or TU45 drives. The DENS=800:1600 is valid with TU16 or TU45 drives when they are not controlled by the TM03 formatter.
Rewind	/RW	Rewinds all magnetic tapes before DSC executes the current command.
Verify	/VE	Copies data from the input volume and compares it with the output volume.

11.5.1 File Label

The file label identifies the data copied from a disk and stored on one or more tapes or on another disk. If you do not specify a file label, DSC uses the volume ID of the input disk volume label as the output volume label.

The file label can consist of from 1 to 12 alphanumeric characters. However, DSC only uses the first 9 characters to identify the file it creates containing the disk's entire contents. Place the file label after the device specification and before any switches. Terminate the file label with one of the following:

- An option switch
- An equal sign (indicating the end of the output side of the command line)
- A carriage return (indicating the end of the command line)

For example:

```
DSC>MM01:,MM02:SYSFILE=DB1:
```

DSC uses the file label SYSFILE in the command above to identify the file on tape that contains the data to be copied from the input disk, DB1:.

You can also use the file label when restoring data from tape to disk. If you enter a file label as part of the input specification, DSC searches the first volume for a file with that name. When it finds that file, DSC transfers it to the output volume. If, however, you do not specify an input file label, DSC transfers the first DSC-created file it locates on the first input volume. In both cases, using the Rewind switch on the input side of the command causes the tape to be rewound before the search for the file starts.

If you use a file label as part of the output specification, it will be used as the file label of the output disk. If you do not specify an output file label, the output disk will have the same label as the original disk. (Even if the disk was copied to tape first, DSC ignores the file name on the tape and uses the name on the original disk.)

For example:

```
DSC>DB1:=MM01:,MM02:SYSFILE
```

In this command, the Rewind switch is not specified on the input side. Therefore, DSC searches the first volume specified, MM01:, beginning at the current position, for a DSC-created file named SYSFILE. If DSC finds SYSFILE on MM01:, it completes the data transfer. If, however, SYSFILE is not found on the first volume, DSC issues an error message and terminates the operation.

If you enter the above command without a file label, DSC transfers the first DSC-created file it finds to DB1: regardless of the file name. (This file may or may not be SYSFILE.) If you do not specify the Rewind switch, the tape may or may not be positioned at the beginning before DSC begins its operation.

11.5.2 Append Switch

The Append switch (/AP) directs DSC to begin writing a file to the first specified volume of a tape set that contains only DSC-created files. If more than one DSC-created file exists on the first volume, and the last file extends to other volumes, DSC displays an error message and terminates the operation.

Enter the Append switch as part of the output specification. The volume to which files will be appended must be specified as the first volume of the output side of the command string, as follows:

```
outdev:[filelabel]/AP=indev:[filelabel][/switch]
```

When you use the Append switch with the output specification, DSC searches from the current position on the first specified tape output volume for the last logical end-of-file (EOF) created by a previous DSC command. If the last DSC-created file does not end on that volume, DSC terminates the operation and issues the following message:

```
OUTPUT TAPE AAnn:  IS FULL
```

If the first specified tape output volume contains a portion of a DSC file that began on a previous volume, DSC terminates the operation and issues the following error message:

```
OUTPUT TAPE AAnn:  IS A CONTINUATION TAPE
```

If DSC locates the end of a file on the tape that began on another volume, DSC terminates the operation and issues the following error message:

```
OUTPUT TAPE AAnn:  IS NOT THE ONLY REEL IN ITS SET
```

For example:

```
DSC>MM01:,MM:SYSFILE/RW/AP=DX1:
```

This command appends the contents of DX1: to the last DSC-created file already present on the first output volume specified, MM01:. DSC first rewinds the tape on MM01: and searches for the last EOF block on the tape. When it determines that only complete DSC-created files exist on the volume on MM01:, DSC appends the new file, SYSFILE, to the file or files already on the tape. If necessary, DSC extends SYSFILE to additional volumes. Since the Rewind switch is specified, the search will not begin until the tape is rewound.

You can only use the Append switch with output tape volumes. Any other use of the switch causes DSC to generate a message and terminate operation.

11.5.3 Bad Block Switch

Use the Bad Block switch (/BAD) with output disk volumes to control the way DSC uses bad block information.

Three options for the Bad Block switch allow you to supplement the output disk bad block file with manually-entered bad block data, to ignore the bad block file altogether, or to use only manually-entered bad blocks. The bad block descriptor of the disk is never altered by DSC.

The format for the BAD switch and its options are:

```
/BAD=MAN
/BAD=NOAUTO
/BAD=MAN:NOAUTO
```

MAN

Allows manual-entry of bad block data, which is included in the bad block file created on the disk.

NOAUTO

Causes DSC to ignore the bad block descriptor on the disk, resulting in an empty bad block file during the DSC operation. Note that in this case DSC will attempt to write in any block it selects.

MAN:NOAUTO

Causes DSC to enter only manually-entered bad block data in the bad block file. Thus, DSC bypasses only manually entered bad blocks when selecting blocks to write in.

When you specify MAN or MAN:NOAUTO with the Bad Block switch, DSC responds with the following prompt:

```
DSC>LBN(S)=
```

DSC issues this prompt after it accepts the original command line but before it transfers any data.

Enter the locations of bad blocks following the LBN(S)= prompt. You can specify multiple bad block entries on one command line, using either a space, tab, or comma to separate each entry. You can also use separate lines for each entry. Enter bad block data immediately following the equal sign (=), in the format:

```
DSC>LBN(S)=n:m
```

n

Is the logical block number (LBN), in octal, of the initial bad block in the group.

m

Is the number, in octal, of consecutive blocks contained in the group. If you do not specify m, it defaults to 1.

To specify a decimal number for either m or n, place a period (.) after the number.

After you enter the first group of bad blocks, DSC reissues the LBN(S)= prompt. At this point, you can enter additional bad blocks by repeating the above procedure.

To terminate manual bad block entry, enter a carriage return after the LBN(S)= prompt.

When you have entered all the bad blocks and terminated the entry process, DSC begins the data transfer.

For example:

```
DSC>DB1:/BAD=MAN:NOAUTO=MM01:,MM02:SYSFILE/RW
DSC>LBN(S)=702:7 (TAB) 644:2
DSC>LBN(S)=4057,5001:3
DSC>LBN(S)=(CR)
DSC>
```

DSC restores the output disk, DB1:, from the tape file SYSFILE contained on MM01: and MM02:, skipping only the blocks you entered manually. In the example above, the following blocks will not be used:

702	644	4057
703	645	5001
704		5002
705		5003
706		
707		
710		

Compare the above example with the following example:

```
DSC>DB1:/BAD=NOAUTO=DB0:
```

This example transfers data to the lowest LBNs on DB1:, regardless of the content of the resident bad block descriptor.

11.5.3.1 **Obtaining Bad Block Information** - You can obtain bad block information in three ways:

1. Running the Bad Block Locator utility (BAD), described in Chapter 9 of this manual
2. Running the Field Service stand-alone diagnostic
3. Running INI[TVOL] with the BAD=[option] switch.

The BAD utility (number 1 above) automatically provides bad block information using LBNs, and creates a bad block file that DSC can use.

The Field Service stand-alone diagnostic reads every word in a block and displays bad block messages on the console terminal. (This diagnostic is recommended for the user who wants more comprehensive testing of a volume). However, since the output is the physical address of each bad block, you must convert this address to logical block numbers before DSC can use it.

11.5.3.2 **Conversion To Logical Block Numbers** - All DSC bad block information must identify bad blocks by LBN.

The manufacturer-furnished or diagnostic bad block information usually identifies bad blocks by physical address (sector-track-cylinder). Before you enter this information manually for DSC, convert the physical addresses to LBN, using the following formula:

```
(((cylinder number*tracks/cylinder)+track number)*sectors/track)+sector number
```

For example, suppose a bad sector of an RP06 (19 tracks per cylinder and 22 sectors per track) has the following physical address:

```
Cylinder Number = 536 (8), 350 (10)
Track Number = 16 (8), 14 (10)
Sector Number = 13 (8), 11 (10)
```

The LBN for the example above is calculated as follows:

```
(((350.*19.)+14.)*22.)+11.=146619.
```

11.5.4 Block Factor Switch

The Block Factor switch (/BL) allows you to set the number of blocks DSC uses in each of its buffers during I/O operations. The default DSC block factor is four blocks or the last value specified in a BL switch.

11.5.4.1 **Using the BL Switch** - The format for the Block Factor switch is:

```
outdev:[filelabel]/BL=n=indev:[filelabel]/BL=n
```

The value of n can be any positive integer, decimal or octal, less than or equal to the maximum block factor available to DSC. This maximum depends on the amount of memory DSC can access under the system configuration.

The BL switch can be specified either on the input or output side of a DSC command line.

If the input volume is tape, DSC determines the block factor from the header label of the input file and ignores specification of the BL switch.

If you specify the BL switch on both sides of a DSC command line with a disk volume, DSC uses the last value it receives, that is, the one from the input side of the command. However, if you specify the BL switch only on the output side of a command, DSC uses that value.

DSC requires 2020(8) bytes of memory for each additional block of buffer space you specify. If the BL switch in a DSC command requires more memory than DSC has available, DSC displays the message BAD BLOCKING FACTOR and exits.

For example:

```
DSC>DB1:/BL=11=DB0:
```

In this example, DSC attempts to increase the number of blocks in each of its buffers to 11. DSC requires an additional 16160(8) bytes of memory for the expansion (7 additional blocks times 2020(8) bytes).

If DSC does not have access to 16160(8) additional bytes of memory on your system, it will display the error message.

If the expansion succeeds, DSC reads and writes 11 blocks of data at one time during an I/O operation instead of 4. This decreases the time required for DSC operations.

Once DSC has expanded its buffers in response to a new value of /BL, that value becomes the default value. DSC does not reduce its task image size if a command is executed at a lower block factor. However, if you specify a lower block factor in a subsequent command, DSC will create that volume at the lower factor.

11.5.4.2 **System-Dependent Requirements for BL Switch** - Table 11-4 describes the memory requirements for buffer expansion with RSX-11M or RSX-11M-PLUS operating systems. Online DSC on a mapped system (11M or 11M-PLUS) expands automatically if memory is available. (See RSX-11M or RSX-11M-PLUS System Generation and Management Guide for details of building DSC with additional memory.)

Table 11-4
Operating System Limits for DSC Block Factor

Operating System	Default Blocking Factor	Maximum DSC Size	Maximum Blocking Factor
RSX-11M/M-PLUS Mapped Systems	4	32K words	36(10)
RSX-11M Unmapped Systems[1]	4	20K words	10(10)

[1] On unmapped RSX-11M systems the DSC task must be rebuilt with additional memory for the block factor to be increased.

NOTE

If you create DSC tapes on an RSX-11M or RSX-11M-PLUS system, at a blocking factor greater than 10, they will be unusable on unmapped RSX-11M systems which can only use DSC tasks of 28K words.

11.5.5 **Compare Switch**

The Compare switch (/CMP) directs DSC to compare the contents of two disks or a disk and a tape set. Multiple tape specifications are valid, but multiple disks are not. If the input volume in a compare operation is tape, use the Compare switch on the input side of the DSC command line. If both input and output volumes are disk, you can use the Compare switch on the output side of the command line. The compare switch performs only comparison operations; no copy operation is involved.

To perform both a copy and compare operation, use the DSC Verify switch (see Section 11.5.8).

Specify the Compare switch as follows:

```
outdev:[filelabel]/CMP=indev:[filelabel]
```

When DSC detects an end-of-volume (EOV) on any reel or end-of-file (EOF) on other than the first reel of a tape set, the CMP switch causes DSC to rewind and unload the current volume and resume comparison with the next volume, until it detects an EOF.

When DSC finds a difference between the volumes it is comparing, it displays a warning on your terminal. This warning lists the output volume number, file identification, and the Virtual Block Number (VBN) where the difference was found. DSC then continues the comparison.

When DSC begins a comparison involving tape, it first positions the specified or implied file as described in Section 11.5.7. DSC positions a single volume tape at the end of the current file when the comparison ends. Each reel of a tape set is rewound and unloaded as the operation on it is completed. DSC then resumes the comparison, using the next volume of the set.

11.5.6 **Density Switch**

The Density switch, with its two options, allows you to override the DSC default storage density of 800 bits per inch for TU16, TE16, TU77, and TU45 tape drives. The following two sections discuss these switches. Although you can use other tape drives with DSC, only the above-mentioned drives can support the Density switch. With TE16 drives, you can use the DENS=1600 switch, but you cannot use the split density switch. In addition, tape drives controlled by a TM03 cannot use the split density switch.

You do not have to specify a Density switch when a tape is the input in a DSC command. DSC determines the density of all input tapes by first reading the tape at 800 bits per inch and, if that fails, then reading at 1600 bits per inch.

If you specify a density switch with a disk, DSC issues an error message and halts the operation.

If you specify a density switch with tape drives other than those above, DSC ignores the switch and does not alter the default 800 bits per inch density. Note that TS04 drives write all tapes at 1600 bits per inch. Therefore, it is not necessary to specify a density switch with DSC commands that use TS04's.

11.5.6.1 **1600 Bits per Inch Switch** - The 1600 Bits per Inch Density switch directs the TU16, TE16, TU77, or TU45 drive to operate as an output volume at a density of 1600 bits per inch. The drive then writes all volumes in the tape set at that density. For example:

```
DSC>MM01:,MM02:SYSFILE/RW/AP/DENS=1600=DB1:
```

In this example, MM01: and MM02:, are written at 1600 bits per inch density.

11.5.6.2 **Split Density Switch** - The Split Density switch (/DENS=800:1600) directs the TU16 or TU45 drives (using the TM02 tape formatter) to write the entire tape set, except for the first two

blocks on each volume, at 1600 bits per inch. The first block on each volume contains header information and the second block is a dummy boot block that displays the following error message if an attempt is made to boot the volume:

```
THIS VOLUME DOES NOT CONTAIN A BOOTABLE SYSTEM
```

In the following example, DSC records the first two blocks of the first volume at 800 bits per inch and the remainder of each volume at 1600 bits per inch.

```
DSC>MM01:,MM02:SYSFILE/RW/DENS=800:1600=DB1:
```

NOTE

Tapes created using the Split Density switch violate ANSI standards.

11.5.7 **Rewind Switch**

The Rewind switch (/RW) directs DSC to rewind all volumes in a tape set before performing any other DSC operation, such as a copy or a compare. You can use it to rewind either input or output volumes (see Table 11-6).

If you enter the RW switch as part of the input specification, DSC rewinds all of the tapes before the DSC operation begins. If you specify a file label with the RW switch, DSC rewinds the tapes and searches for the file you specified from the Beginning of Tape (BOT) on the first volume.

If you do not specify a file label, DSC transfers the first DSC-created file it encounters on the first volume. After one volume of a tape set has been copied, DSC rewinds it and places it offline. If, however, the current file ends on the first or only tape of a set, the tape is positioned to read the next file on the input tape. The RW switch only rewinds tapes at the beginning of a DSC operation.

If you enter the RW switch as part of the output specification, DSC begins by writing or comparing data from the beginning of the rewound output tape. Thus, in a copy operation, DSC overwrites any data previously stored on that tape, unless the Append switch is included in the command.

If you do not enter the RW switch with the output specification, and the first volume is not positioned at BOT, DSC begins its operation after the first DSC-created EOF it reads.

If the tape is positioned at BOT, DSC overwrites it from there unless you specified the AP switch. If the file extends beyond the first volume of a tape set, that volume and all subsequent volumes are rewound and unloaded as they are filled. Otherwise, the tape is left at EOF to append another file to the first volume of the set. If DSC finds an EOV during its search for the last EOF, the command is aborted and DSC issues an error message.

Table 11-5 summarizes the use of the Rewind switch with various DSC operations and with and without a file label.

Table 11-5
Commands Using DSC Rewind Switch

Switch	Specification	File Label	Action
RW	Input/Output	With/without	Rewinds first tape before copy operation begins.
RW	Input	With	DSC searches for specified file from the beginning of the first tape volume.
RW	Input	Without	DSC transfers the first file it encounters on the first volume.
RW	Output	File labels specified when tape is output volume are ignored when the tape is restored to disk.	DSC writes data, starting at the beginning of the first tape volume.
No Rewind Switch	Output		DSC writes data, beginning after the first end-of-file block it encounters. (If tape is already at BOT, DSC starts there.) During copy operations to multiple tapes, DSC rewinds the tape as it is filled, and takes it offline.

The RW switch can be used only with magnetic tapes. If you use it with any other volume, DSC prints an error message.

An example of the use of the Rewind switch follows:

```
DSC>MM01:SYSFILE/RW=DB1:
```

DSC rewinds the volume on drive MM01: and overwrites any data on the tape. The contents of DB1: are written to a single file identified as SYSFILE. DSC does not rewind the tape when the operation is finished unless the file extends to another volume. If the file does extend, DSC rewinds and unloads the filled tape. DSC ensures that subsequent tapes are at the beginning of tape before using them for read or write operations. Each subsequent volume, including the last one in the tape set, is rewound and unloaded when it is filled.

The following example shows the restoration of a DSC-created file:

```
DSC>DB1:=MM02:,MM01:SYSFILE/RW
```

In this example, DSC restores a volume (DB1:) using a tape created by a previous DSC operation. DSC rewinds the first volume (on MM02:) and searches for a previously created DSC file labeled SYSFILE. If the file is found, DSC transcribes it. If it is not found on MM02:, DSC issues a message and terminates the operation. DSC will not go on to MM01: if the file does not begin on MM02:. Each volume of the tape set is rewound and unloaded when the data it contains has been copied or compared.

NOTE

When you refer to tapes for the first time since your system was booted, you must use the Rewind switch. If you do not use the switch, the tape driver will return an error.

11.5.8 Verify Switch

The Verify switch (/VE) entered as part of the output specification directs DSC to perform a copy operation followed by a compare operation to verify that the two volumes are the same. (DSC does not allow you to specify either the Verify or Compare switch if both input and output volumes are tape.)

If either the input or output volume is tape, the Verify operation takes place at the end of the Copy operation for each volume. In other words, DSC writes MM01: and compares MM01:; writes MM02: and compares MM02:, until the entire DSC operation is complete. In a disk-to-disk DSC operation, the Verify operation begins when the Copy operation is finished.

You specify the Verify switch as follows:

```
outdev:[filelabel]/VE=indev:[filelabel][/switch]
```

If you do not specify a file label for a tape set, the VE switch causes DSC to begin copying the first volume of the set.

When DSC detects EOV or EOF on any volume of a tape set during a copy operation, it repositions the input volume to the beginning of the current file and begins the verify operation.

During a verify operation, if DSC detects EOV on any volume, or EOF on other than the first volume of a tape set, it rewinds and unloads the tape when the operation is complete. After an EOV, the copy operation resumes, using the next volume from the beginning of the tape.

NOTE

If you specify a tape as one of the volumes, DSC requires extra time after the copy operation to rewind the tape and search for the current file before it begins to verify.

11.6 DSC OPERATION OVERVIEW

DSC initially accesses the first primary file header and writes the blocks mapped by its retrieval pointers to the output volume. DSC then checks the primary file header to determine whether it points to any extension headers. If extension headers exist, DSC transcribes them and the blocks they map until the entire file, with all of its extensions, has been written to the output volume. Only then does DSC access the next primary file header in the index file. For example:

```
DSC>DB1:=DB2:
```

In this example, DSC copies all the files on DB1: to DB2:.

When DSC copies file extensions it updates the output retrieval pointers and file linkages involved in the transfer as required. This not only involves collapsing retrieval pointers, but also may reduce the number of file extensions required if enough retrieval pointers are eliminated.

As a result of a copy operation, each primary file header is followed by all of its extensions. Volumes created in a Merge mode copy operation have complete files written to contiguous blocks (except where blocks have been flagged as bad in earlier operations on the volume). DSC writes data, beginning at the lowest LBN possible on the disk.

If an input file is contiguous, DSC will search for an area on the output volume with enough contiguous blocks to contain the file. If no such area exists, DSC will issue an appropriate message and terminate the copy operation.

If an input file is not contiguous, data is allocated in as few contiguous sections as possible, in the first unoccupied blocks available on the output volume.

11.7 STAND-ALONE DSC

Stand-alone DSC (DSCS8.SYS) does not support all the features of the online version. DECtapes, floppy diskettes, DF/DS fixed-head disks, and TU58 cassettes cannot be used with stand-alone DSC. However, in data transfer operations, stand-alone DSC uses all of the switches described in Section 11.5.

When you run stand-alone DSC without any of the switches described below the following default values apply:

- Logical unit numbers of devices are limited to 0 and 1
- Only two input or output volumes can be specified per command
- One controller is supported for each device type
- Nonstandard vector addresses are generated for RM03, RP02, and RP03 disk volumes, and TU10/TE10, and TS03 magnetic tape drives

You can overcome some of these limitations by using four switches, listed in Table 11-6. These switches change the standard DSC default values to the values your system requires. However, the switches can only be used with stand-alone DSC.

Table 11-6
Stand-Alone DSC Switches

Format	Switch	Description
/CSR=xxxx	Control Status Register switch	Alters control status addresses for specific device types
/TM02=x	TM02/TM03 Formatter switch	Alters the physical unit formatter
/UNIT=x	Unit switch	Specifies the physical unit that a LUN accesses
/VEC=xxx	Vector Address switch	Alters the vector address of a unit

The four switches supplied with stand-alone DSC can appear together in a single command line to alter the appropriate values of a single device or device type. However, you can only alter values for one device type, or generate one data transfer operation, in a single stand-alone DSC command line.

Therefore, when you use these switches, you must enter at least two command lines; one to specify the switches with a device or device type, and one to initiate the DSC data transfer operation.

NOTE

Once you use the switches, DSC uses them in all subsequent commands until you either specify new switches in a new command or terminate DSC.

The general format for a stand-alone DSC command with switches is:

```
DSCS8> AAnn:/switch1=x.../switchn=y
```

AAnn:

The device identifier and unit number

switch1 through switchn

One or more of the stand-alone switches described in Sections 11.7.1 through 11.7.4

x,y

The value you assign to the switch

11.7.1 Control Status Register Switch

Use the Control Status Register switch (/CSR) to alter the device Control Status Register addresses generated by stand-alone DSC so that they conform to the addresses required by your system.

Table 11-7 displays the CSR and vector addresses of the devices supported by stand-alone DSC.

Table 11-7
System-Generated CSR and Vector Addresses

Device Type	CSR	Vector
DB:	176700	254
DK:	177404	220
DL:	174400	160
DM:	177440	210
DP:	176714	300*
DR:	176700	320*
MM:	172440	224
MT:	172522	320*

* Indicates nonstandard vector address.

The following example illustrates the correct use of the CSR switch:

```
DSCS8>MM1:/CSR=160546
DSCS8>DB0:/CSR=160646
```

In this example, the system using DSC has the CSR addresses of the MM1: tape drive and the DB0: disk drive set to 160546 and 160646, respectively. After you enter these values, you can enter another DSCS8 command line to initiate a copy and/or compare operation. Neither of the commands that use the CSR switch in the example cause a copy operation to begin.

If a DSC operation involves multiple devices of the same type, only specify the CSR once for each device type.

11.7.2 **TM02 Switch**

The TM02 switch (/TM02) alters the physical unit number of the TM02/TM03 formatter on the RH controller from the value generated by stand-alone DSC to the value required by your system.

Stand-alone DSC assigns a physical unit number of 0 to the TM02/TM03 formatter on the RH controller. You can change this to any octal digit from 1 to 7 for each MM: device. For example:

```
DSCS8>MM1:/TM02=1
```

This command alters the TM02/TM03 position on the RH controller from 0 (the generated value) to 1 for MM1:. The TM02 switch alters only the device you specify. If another MM: device requires an altered

assignment, which is the usual case, specify the new device and its assignment in a separate command. The TM02 switch only works with MM: devices; it cannot be specified with an MT: or disk device.

11.7.3 Unit Switch

You can use the Unit switch (UNITS=) to change the unit numbers DSC accepts for device specifications. Stand-alone DSC is generated with, and accepts only, two logical unit numbers, 0 and 1. In addition, the unit numbers used in the command string cannot be changed. These constraints can be amended somewhat with the UNITS= switch. The numbers 0 and 1 must still be specified in the command line, and the number of devices cannot be increased. However, DSC can access devices with physical numbers other than 0 and 1. For example:

```
DSCS8>DP0:/UNIT=5
```

After you enter the above command and a command line that references DP0:, the disk actually accessed is the disk assigned as physical unit 5. If DP1: also requires a physical unit change, it will have to be specified in another command.

11.7.4 Vector Address Switch (/VEC=)

The Vector Address switch (/VEC=) switch changes the stand-alone DSC vector addresses to the addresses required by your system. Each unit of the device type is accessed by the specified vector address. For example:

```
DSCS8>DB1:/VEC=320
```

After you enter the above command, all DB: devices will be accessed with a vector address of 320.

Stand-alone DSC uses nonstandard vector addresses to resolve conflicting unit configurations. These conflicts occur when a system contains:

- A TU16 drive and TE10/TU10 drive or TU16 drive and a TS03 drive
- Any combination of RP02/03, RP04,05/06, and RM02/03 (such as an RP02 disk and an RP04 disk).

For example, before you can reference TE10, TU10, or TS03 tapes, you must use the VEC switch to alter the DSC vector setting of 320 to the correct value for your system.

```
DSCS8>MT1:/VEC=224
```

After you enter this command, all MT devices will be accessed with a vector address of 224 (instead of the DSC-generated vector address of 320).

If the VEC switch is not used to alter the DSC setting, DSC waits for a response from the incorrect vector address. This response never comes.

11.8 DSC OPERATION

As outlined in the beginning of this chapter, DSC's complete data transfer process consists of either a direct disk-to-disk operation or a two-step, disk-to-tape/tape-to-disk operation. DSC reads and writes data to and from its own internal buffers during these operations.

The following sections describe DSC's operation in each of these data transfers.

11.8.1 Data Transfer from Disk

After you enter a DSC command specifying a copy operation from a disk, DSC scans the input disk to ensure that it is in Files-11 format. DSC begins by copying an approximation of the disk index file. Because this file is updated to reflect the status and location of blocks as they are allocated on the new disk, the index file bit map, the storage bit map file, and the bad block file are not transcribed exactly: DSC transcribes only the data necessary for the construction of these files on the new disk. However, the index file bit map still reflects the maximum number of files on the input disk.

DSC accesses the input volume index file's index of active file headers in numerical order to locate the next active primary file header. DSC transfers that header, the blocks it maps, and all extension headers and related blocks that are part of the file, to the output medium. It then accesses the next active primary file header from the index file. DSC continues this operation, each time writing a complete file, until it has transferred all the active files.

DSC accesses and transcribes only the blocks allocated to active files. It ignores unallocated blocks interspersed throughout the input disk. This results in contiguous data blocks on the output disk, following the copied files.

If DSC accesses a file that contains bad data, DSC transcribes whatever it reads from the block. When DSC restores the file to disk, it writes the block's contents as it originally read them. The logical block still contains garbled data, but the new physical block can be accessed and its contents corrected. A message identifying these bad areas is displayed on the console terminal.

In summary, to transfer data from a disk, DSC:

1. Verifies that the disk is online, and in Files-11 format
2. Transcribes disk index files, updated for their new status
3. Reads the data to a DSC buffer

11.8.2 Data Transfer to Tape

When the output volume in a DSC operation is tape, DSC writes the contents of the input disk to a tape on the drive you specify. This data transfer usually involves multiple reels of tape (a tape set) and multiple tape drives.

The tapes that DSC creates serve as a back-up of the disk's contents. You can only use DSC-created tapes by copying them back to a disk volume and restoring the disk's contents to their original form.

Although the tapes contain many individual files from the input volume, DSC treats the tapes as if they contained a single file -- a file of the disk's entire contents.

When DSC begins writing the volume's contents to tape, it allows writing to more than one volume. The first block DSC writes to tape is a header that contains the volume name (obtained from the file label) and the relative volume number. This header identifies the tape set and the volume's place within that set. It ensures that when DSC begins to restore the disk it will load each volume in the tape set in order.

After the header, the tape set includes the data required to reconstruct directory files, maps and pointers, and the actual files copies from the disk.

NOTE

When the disk is restored, the directory files are at the beginning of the disk, regardless of their position on the original disk.

To initiate the write-to-tape operation, first ensure that the tape devices are online. You can specify multiple tape drives in the following way:

```
DSC>AAnn(0):,AAnn(1):,...AAnn(7):[filelabel]=indev:
```

An example of a command in this format is:

```
DSC>MM0:,MM1:,MM4:,MM2:SYSFILE=DB1:
```

You have the option of entering a file label in this command after specifying the last device. You can specify only one type of tape drive, either MM: or MT: or MS:, in a single DSC command. Although you can specify up to eight drives on each side of the command line (two drives in the stand-alone version) you can specify each individual drive only once.

If the number of volumes in the tape set exceeds the number of tape drives available, DSC uses volumes on the specified drives in round-robin fashion. Using the above example, the order of replacement would be

```
MM0:  MM1:  MM4:  MM2:  MM0:  MM1:  MM4:  MM2:  ...
```

until an end-of-file is reached.

In summary, to transfer data to tape, DSC:

1. Verifies that the first or only volume of a tape set is online and write-enabled
2. Verifies that subsequent volumes of a tape set are at Beginning of Tape (BOT), online when required, and write-enabled
3. Transcribes data from a DSC buffer to the tape

11.8.3 Data Transfer from Tape

DSC can only use the tapes it creates to (1) reconstruct a disk or (2) perform compare and verify operations.

When you mount the tapes and specify tape drives as input devices, DSC sequentially accesses and writes the tape contents to the output volume. As it transfers the data, DSC creates and updates directory files.

Tape drives specified as input devices must be online. The volumes in the tape set must be referenced in the correct order in the command line.

If you specify a file label, DSC transfers only the contents of the file identified by that label. If you do not specify a file label, DSC transfers only the first DSC-created file it encounters on the first volume of a set.

In summary, to transfer data from tape, DSC:

1. Verifies that the tape drives are online
2. Accesses the volumes in a tape set in round-robin order
3. Creates directory files
4. Reads the data into a DSC buffer

11.8.4 Data Transfer to Disk

DSC's operation is not complete until the data involved in the transfer is restored to disk.

To receive input, a disk must be online. Any disk large enough to contain all the input data can be specified as the output volume in a data transfer to disk.

The disk should have an up-to-date bad block descriptor, or have bad block data entered in a DSC command with the BAD switch. This ensures that the data written on the disk will be accessible. You can update the bad block descriptor before a DSC operation by running the BAD utility.

After identifying the bad blocks on the output disk, DSC examines that volume to ensure that it has enough free blocks to contain all the data being transferred. DSC compares the number of blocks being transferred from the input volume(s) with the number of blocks available on the output volume. DSC issues an error message and exits if too few blocks are available.

DSC begins constructing index and storage bit map files when it begins transcribing files. DSC updates the file headers to reflect the location of the files on the new disk. This updating is required because blocks that were previously scattered are now copied to a contiguous set of blocks, beginning at the lowest LBNs available on the disk. DSC will write the primary file header and its contents and associated file extension headers and the extensions they map, as a unit to a contiguous series of blocks. Note that the output disk contains an index file of the same size as the original disk. This is especially important when the contents of a large disk (such as an RP04) are restored to a smaller disk (such as an RK05) or vice versa.

Compression of files in this manner is beneficial when a file header's retrieval pointers are almost used up. Because DSC rearranges a disk so that contiguous blocks are allocated to a single data file, the number of retrieval pointers required to map the location and length of the file contents can be significantly reduced. DSC can also reduce the number of file extensions and extension headers.

Note that when DSC writes to a disk, it begins writing data into the lowest block possible. Free blocks generally have higher LBNs and are in a contiguous section of the disk.

The data presently on the disk is overwritten by the new data. Therefore, you cannot use DSC to transfer the contents of several small disks to a single large disk. Each copy operation eliminates whatever previously occupied the disk.

In summary, to transfer data to a disk, DSC:

1. Verifies that the disk is online.

2. Verifies that the disk has an up-to-date bad block descriptor or that bad blocks are specified manually (through the /BAD=NOAUTO switch). Displays a warning message if no bad block information is available and the BAD switch was not specified.

3. Verifies that the disk has enough free blocks to contain all the data to be transferred.

4. Creates index and directory files (in the first part of the disk).

5. Writes the data from a buffer.

11.9 DSC MESSAGES

DSC notifies you of fatal error conditions as well as less serious conditions that could cause difficulties in DSC operations. Each message generated by DSC has the prefix DSC--, and each is identified by a numeric code.

DSC messages are printed on your terminal in either a long or a short form: online DSC prints the long form and stand-alone DSC prints the short form. You can determine the meaning of the short form message from the number provided with the message. The text accompanying the long form message of that number explains the error.

For example, specifying a tape in the wrong format generates the following message in long form from DSC:

```
FATAL *** 17 OUTPUT TAPE MM1:  NOT ANSI FORMAT
```

The same error generates the following message in short form on stand-alone DSC:

```
FATAL *** 17 - MM1
```

Table 11-8 is a quick reference to the single letter codes used in General messages and in I/O messages.

Table 11-8
General Error and I/O Error Message Codes

Type of Code	Symbol	Meaning
General Error Message	Code A	Failed to read storage map header
	Code B	Input data out of phase
	Code C	Nondata block encountered
	Code D	Input file out of phase
	Code E	File attributes out of phase
	Code F	File header out of phase
I/O Error Message	A	Reading index file bit map
	B	Reading index file header
	C	Reading storage bit map
	D	Reading boot or home block
	E	Reading file header
	F	Input (or output device)
	G	Writing index file bit map
	H	Writing storage bit map header
	I	Reading input device
	J	In input tape labels
	K	Reading file attributes
	L	Reading file header
	M	Reading index file data
	N	Reading summary data
	O	Writing file header

11.9.1 DSC Messages: Text, Explanation, and Corrective Action

The following are the messages DSC can return.

1 UNDEFINED ERROR

Explanation: DSC encountered an unidentifiable internal error.

User Action: First, retry the operation. If error persists, submit a Software Performance Report.

2 CONFLICTING DEV. TYPES

Explanation: An illegal combination of device types was specified.

User Action: Check for typographical errors in device abbreviations: make sure that the disks and tape drives are not specified on the same side of the command string.

3 MIXED TAPE TYPES

Explanation: Two different types of tape drives were specified in the command string.

User Action: Reenter the command, specifying only one type of tape drive.

4 ILLEGAL SWITCH

Explanation: The command string was entered with a switch that cannot be used with that command.

User Action: Reenter the command with only correct switches specified.

5 FILE LABEL TOO LONG

Explanation: A file label consisting of more than 12 alphanumeric characters was specified.

User Action: Retry the operation with a shorter file label.

6 SYNTAX ERROR

Explanation: An error occurred in the command string format.

User Action: Reenter the command in the right order.

7 DUP. DEV. NAME;

Explanation: The same device was entered more than once in the command.

User Action: Reenter the command, specifying each device only once.

8 TOO MANY DEV'S

Explanation: More than the legal number of devices were specified on one side of the command line.

User Action: Reenter the command, specifying no more than eight devices per side.

9 DEV. XXn: NOT IN SYSTEM

Explanation: The specified device is not present in the configuration of the operating system being used.

User Action: Check the device identifier that was entered in the command string. Reenter the command.

10 DEV. XXn: NOT FILES-11

Explanation: The specified input device is not formatted as a FILES-11 device.

User Action: Check the input device to ensure it is the one desired, and reenter the command.

11 BAD BLOCK SYNTAX ERROR

Explanation: A syntax error occurred when bad block data was manually entered.

User Action: Check the command that was entered, and reenter it.

12 BAD BLOCK COUNT TOO LARGE

Explanation: Too many bad blocks were manually entered in a single group.

User Action: Check the blocks being entered. If possible, enter several small groups instead of one large group.

13 BAD BLOCK CLUSTER OUT OF RANGE

Explanation: A manually entered bad block or group of bad blocks does not exist on the output disk.

User Action: Check the numbers of the blocks entered, and reenter the command.

14 OUTPUT TAPE XXn: NOT AT BOT

Explanation: The specified continuation tape is not at the Beginning of Tape (BOT).

User Action: Remount or reset the tape at load point and reenter the command.

15 OUTPUT TAPE XXn: FULL

Explanation: The specified tape is full, and files cannot be appended to it.

User Action: Change the output tape, and reenter a command to begin a new tape set.

16 OUTPUT TAPE XXn: NOT ONLY REEL IN SET

Explanation: The Append switch was used with a tape that was not the first tape of a set created by DSC.

User Action: Change tapes, and reenter the command (see message 15).

17 TAPE XXn: NOT ANSI FORMAT

OUTPUT TAPE

Explanation: If device is an output tape, the AP switch was specified and tape is not in ANSI format.

User Action: Reenter the command and either omit the AP switch to write the specified tape or change tapes.

INPUT TAPE

Explanation: If device is an input tape, the tape is not in the correct format for a DSC operation (that is, the tape was not created by DSC).

User Action: Check the tape and change it, if necessary. Reenter the command.

18 OUTPUT TAPE XXn: NOT DSC TAPE

Explanation: An Append switch was specified with a tape that was not created by DSC.

User Action: Reenter the command and either omit the Append switch or change to a DSC-created tape.

19 TAPE XXn: A CONTINUATION TAPE

Explanation: If device is an input tape, the tape was mounted out of sequence is not the first of a set.

User Action: Reenter the command, and specify input tapes in the proper order.

21 FAILED TO FIND HOME BLOCK

Explanation: DSC failed to find the home block on the input disk. Either the disk is bad, the home block is bad, or the disk is not in FILES-11 format.

User Action: Check the disk in question, change drives if possible, and reenter the command.

22 FILE STRUCTURE LEVEL ON XXn: NOT SUPPORTED

Explanation: The device is not a Files-11 Structure Level One disk, and, therefore, cannot be used.

User Action: Replace the device and retry the operation.

23 I/O ERROR A ON XXn:
... (Additional error information)

Explanation: The I/O error indicated explains why the index file bitmap on the device could not be read.

User Action: Retry the operation after correcting the cause of the error on the device if possible.

24 I/O ERROR B ON XXn:
... (Additional error information)

Explanation: The I/O error indicated explains why the index file header on the device could not be read. The specified file is lost.

User Action: Retry the operation after correcting the cause of the error on the device if possible.

25 CODE A

Explanation: The file header for the storage bit map file cannot be read.

User Action: The disk is unusable and, therefore, cannot be copied. Replace the disk and reenter the command.

26 I/O ERROR C ON XXn:
... (Additional error information)

Explanation: The I/O error indicated explains the error that occurred when DSC read the specified file.

User Action: Retry the operation.

27 I/O ERROR D ON XXn:
... (Additional error information)

Explanation: The I/O error indicated explains the read error that occurred when DSC read the home or boot block of the disk.

User Action: Retry the operation on a new drive.

31 I/O ERROR E ON XXn: file id
... (Additional error information)

Explanation: The I/O error indicated explains the error that occurred when DSC read the specified file header.

User Action: Retry the operation.

32 INPUT DEVICE XXn: fileid, Y, Y NOT PRESENT

Explanation: The specified file does not have a file header in the index file; therefore, the file is not copied.

User Action: This is a warning only. If desired, the operation can be retried on a different disk drive.

33 INPUT DEVICE XXn: file id Y, Y IS DELETED

Explanation: The specified file was partially deleted on the input disk and was not copied.

User Action: This is a warning only. No action is required.

34 INPUT DEVICE XXn: FILE ID Y, Y UNSUPPORTED STRUCTURE LEVEL

Explanation: The specified input disk is not a FILES-11 Structure Level One disk and cannot be used.

User Action: Replace the disk and retry the operation.

35 INPUT DEVICE XXn: FILE ID Y, Y FILE NUMBER CHECK

Explanation: An incorrect file header was read from disk, causing the specified file to be lost.

User Action: Retry the operation.

36 INPUT DEVICE XXn: FILE ID, Y, Y FILE HEADER CHECKSUM ERROR

Explanation: Incorrect file header contents caused the specified file to be lost.

User Action: Retry the operation.

37 INPUT DEVICE XXn: FILE ID Y, Y SEQUENCE NUMBER CHECK

Explanation: The sequence number is incorrect.

User Action: Retry the operation and/or replace the disk.

38 INPUT DEVICE XXn: FILE ID Y, Y SEGMENT NUMBER CHECK

Explanation: The linkage connecting file segments was broken; the specified file is lost.

User Action: Retry the operation.

39 DIRECTIVE ERROR error number

Explanation: An internal error occurred, usually the result of a system overload.

User Action: Retry the operation.

40 I/O ERROR F
... (Additional error information)

Explanation: The I/O error indicated explains that an uncorrectable read/write error occurred on the specified input or output device.

User Action: This message is a warning only. No action is required unless another error message is displayed. If another error message is displayed, correct the cause of the error and reenter the command.

41 I/O ERROR I on XXn:
File ID Y, Y VBN Z, Z
... (Additional error information)

Explanation: An I/O error occurred, which resulted in bad data being read from the specified virtual block number of the indicated file on the indicated device.

User Action: This a warning message only. You should examine the block specified to determine the extent of the error.

42 VERIFICATION ERROR ON XXn:
FILE ID Y, Y, Y virtual block number Z, Z

Explanation: The input and output devices specified for a verification operation did not match.

User Action: This is a warning message only. No user action is necessary.

43 BAD DATA BLOCK ON XXn:
FILE ID, Y, Y, Y VBN Z, Z

Explanation: A parity error occurred when DSC copied the contents of a block from a disk. The block specified on the output disk contains erroneous data.

User Action: When the copy operation is completed, the data contained in the specified block should be examined and corrected.

44 MOUNT REEL x ON XXn: AND HIT RETURN

Explanation: This is an instruction only.

User Action: Mount the volume number requested on the specified tape drive and enter a carriage return when ready.

45 STARTING VERIFY PASS

Explanation: The copy operation is complete and DSC is beginning the verify operation (specified with the VE switch).

User Action: This is an informational message only. No user action is required.

46 RESUME COPYING

Explanation: The verify pass (specified with the VE switch) is complete, and DSC is continuing the copy operation (if there is more material to copy).

User Action: This is an informational message only. No user action is required.

47 XXn: IS WRITE LOCKED. INSERT WRITE RING AND HIT RETURN.

Explanation: The indicated device is write-locked.

User Action: Make sure the device is the one you want, write-enable it, and enter a carriage return.

48 INPUT FILE ON XXn: WILL BE RESYNCHRONIZED

Explanation: The tape position was lost while DSC was reading the input tape. The file specified in the message, as well as some subsequent files, may be lost. DSC may display additional error messages.

User Action: Retry the operation from the beginning.

49 OUTPUT DEVICE XXn: FULL
FILE ID Y, Y, Y

Explanation: The specified device cannot accommodate the indicated contiguous file in a contiguous set of blocks. This may mean that there is an inconsistency in the input tapes.

User Action: Reenter the command, using a less fragmented output disk.

50 OUTPUT FILE HEADER FULL ON XXn: x, FILE ID Y, Y, Y

Explanation: Too many bad blocks on the output disk have caused the generation of more retrieval pointers than can be stored in the header(s) of the file. The copy operation on the specified file is aborted and DSC copies the next file to the input volume.

User Action: After DSC completes the copy operation, use PIP to delete the unusable file on the output volume and to copy the file from the input volume to the output volume.

If the original operation included the Verify switch, this error will cause many errors to be flagged during the verify operation. To avoid this, abort the DSC operation and reenter the command without /VE. Then go through the procedure outlined above. After you copy the invalid file with PIP, you can use DSC with the Compare switch to verify the new disk.

51 OUTPUT FILE HEADER ON XXn: NOT MAPPED - FILE ID Y, Y, Y

Explanation: Space for the specified file header was not allocated. The file is lost.

User Action: Retry the operation; a new disk may be required.

52 I/O ERROR G ON XXn:
... (Additional error information)

Explanation: The I/O error indicated explains the error that occurred while DSC was writing the Index File Bitmap.

User Action: Retry the operation.

53 FAILED TO READ FILE EXTENSION HEADER ON XXn: FILE ID Y, Y, Y

Explanation: When copying from the input disk, DSC searched for an extension header, but did not find one. The remainder of the specified file was lost. A problem may exist with the input disk, or a preceding I/O error may have caused an inconsistency.

User Action: Retry the operation.

54 FAILED TO ALLOCATE HOME BLOCK

Explanation: The home block cannot be created on the specified disk device because it has too many bad blocks.

User Action: Replace the disk and reenter the command.

55 INDEX FILE ALLOCATION FAILURE

Explanation: Too many bad blocks exist to allow the allocation for the specified file.

User Action: Replace the disk, and reenter the command.

56 OUTPUT DISK XXn: IS NOT BOOTABLE

Explanation: Logical block number 0 of the specified disk or tape is bad.

User Action: This is a warning only. No action is required.

57 INVALID BAD BLOCK DATA

Explanation: The bad block data on the output disk is invalid.

User Action: Run the BAD utility on the disk and manually enter bad block data, or reenter the command specifying another disk.

58 BAD BLOCK FILE FULL

Explanation: Too many bad blocks exist on the output disk.

User Action: Replace the disk, and retry the command.

59 NO BAD BLOCK DATA FOUND

Explanation: No bad block data exists for the specified output disk.

User Action: If bad block data is not desired, ignore the message. Otherwise, run the BAD program on the disk; manually enter bad block data; or reenter the command, using a new disk.

60 OUTPUT DEVICE XXn: IS A DIAGNOSTIC PACK. DO NOT USE IT!

Explanation: The specified output disk is a diagnostic pack, and cannot be used.

User Action: Mount a new output disk, and reenter the command.

61 CODE B ON XXn:
FILE ID Y, Y, Y VBN Z, Z

Explanation: The tape position was lost when DSC read the virtual block number specified. Some data may be lost.

User Action: Determine the extent of the error. If necessary, try the tape on another drive, or create another tape.

62 CODE C ON XXn:
FILE ID Y, Y, Y VBN Z, Z

Explanation: The tape position was lost when DSC read the data file specified. Data beyond the VBN mentioned is lost.

User Action: Recreate the tape, or retry the operation on a different tape drive.

63 CODE D ON XXn:
FILE ID Y, Y, Y EXPECTED P, P, P FOUND Y

Explanation: The tape position was lost while DSC read the tape specified in the message. All of "Y, Y, Y" and some of "P, P, P" are lost.

User Action: Retry the entire operation.

64 FAILED TO MAP OUTPUT FILE ON XXn:
FILE ID P, P, P VBN Z, Z

Explanation: An inconsistency occurred when DSC was writing the specified file to output disk. The file header did not specify the correct number of virtual blocks required to write the file and the file is lost.

User Action: Retry the operation.

65 OUTPUT DISK XXn: IS TOO SMALL -- nn BLOCKS NEEDED

Explanation: The output disk is not large enough to accommodate the data to be transferred.

User Action: Retry the operation, specifying a larger output disk.

66 I/O ERROR C ON XXn:
... (Additional error information)

Explanation: The I/O error indicated explains the error that occurred while DSC was reading the storage bitmap.

User Action: Retry the operation.

67 I/O ERROR H ON XXn:

Explanation: The message that follows explains the I/O error that occurred while DSC was writing the header of the storage bitmap file.

User Action: Retry the operation.

68 I/O ERROR J ON XXn:
... (Additional error information)

Explanation: The I/O error indicated explains the error that occurred when DSC was reading the tape labels on the specified device.

User Action: Retry the operation on a different tape drive.

69 INPUT TAPE ON XXn: MUST BE AT BOT

Explanation: The specified tape must be at Beginning of Tape (BOT). This message is also displayed during a Verify operation, to indicate that the current volume is rewinding to enable the verify pass.

User Action: If /VE was not specified, check the tape and remount at BOT.

70 WRONG INPUT TAPE ON XXn:
EXPECTING file id, FOUND file id

Explanation: The input tapes were specified out of sequence.

User Action: Check the tapes, and reenter them in proper order after receiving mount instructions.

71 CODE E ON XXn: AFTER FILE ID Y, Y, Y

Explanation: This is the result of a read error from tape. When trying to read an attribute block, DSC accessed some other block. The file following the file specified in the error message is lost.

User Action: Retry the operation.

72 I/O ERROR K ON XXn:
AFTER FILE ID Y, Y, Y
... (Additional error information)

Explanation: The message that follows explains the I/O error that occurred while DSC was reading the attributes of the specified file.

User Action: Retry the operation.

73 I/O ERROR L ON XXn:
AFTER FILE ID Y, Y, Y
... (Additional error information)

Explanation: The message that follows explains the I/O error that occurred while DSC was reading the file header from tape.

User Action: Retry the operation.

74 INPUT TAPE XXn: RESYNCHRONIZED AT FILE ID Y, Y, Y

Explanation: The tape position has been recovered. Some data preceding the file specified was lost.

This message is usually received with one or more error messages, all indicating that the input tape was either read incorrectly or recorded badly.

User Action: The tape should be re-created and the operation reinitiated.

75 TAPE FILE filelabel NOT FOUND

Explanation: The input tape specified does not contain the file identified as "filelabel."

User Action: Check the file label and the tape, reenter the command when the correct tape and file label are specified.

76 EXPECTED EXTENSTION HEADER NOT PRESENT ON XXn: FILE ID Y, Y, Y

Explanation: A required file extension header could not be found on the tape being read.

User Action: If the error message was preceded by one or more I/O warning messages, the operation should be retried. If not, the input tape is bad and should be regenerated.

77 CODE F ON XXn: AFTER FILE ID Y, Y, Y

Explanation: This is the result of a read error from tape. When trying to read a file header, DSC accessed some other block type. The file following the file specified in the error message is lost.

User Action: Retry the operation.

78 I/O ERROR M ON XXn:
... (Additional error information)

Explanation: The message following the device name explains why the Index File data could not be read.

User Action: Retry the operation.

79 INDEX FILE DATA NOT PRESENT

Explanation: When reading the input tape, DSC accessed a file other than the index file, due to a tape error or an I/O error.

User Action: Re-create the tape, or retry the same tape on a different tape drive.

80 I/O ERROR N ON XXn:
... (Additional error information)

Explanation: The message explains the I/O error that occurred while DSC was restoring the index and storage bitmap files from the specified input tape.

User Action: Retry the operation, using a different input tape drive.

81 VOLUME SUMMARY DATA NOT PRESENT

Explanation: Either DSC did not create the input tape or the tape contains incomplete data.

User Action: Check the tape and reenter the command.

82 I/O ERROR O ON XXn: - FILE ID Y, Y, Y
... (Additional error information)

Explanation: The message explains the I/O error that occurred while DSC was writing the specified file header.

User Action: Retry the operation.

83 BAD BLOCKING FACTOR

Explanation; The specified blocking factor is too large for the current operating system.

User Action: Specify a smaller blocking factor and repeat the command.

11.9.2 DSC I/O Messages

In online and both stand-alone versions of DSC, errors identified as I/O errors are accompanied by one or more of the following messages which explain the type of I/O error that occurred.

*BAD BLOCK NUMBER

Explanation: The block does not exist on the disk; an internal DSC error has occurred.

*BAD BLOCK ON DEVICE

Explanation: A bad area was encountered on the device, resulting in a block that cannot be read or written without error.

BLOCK CHECK OR CRC CHECK

Explanation: A parity error occurred indicating that bad data may have been transferred.

User Action: Retry the operation.

DATA OVERRUN

Explanation: A physical block on tape contains more bytes than were requested.

*DEVICE NOT READY

Explanation: The device is not ready or not up to speed.

DEVICE OFFLINE

Explanation: The device is not in the system.

User Action: Check the device, the device specification in the command string, and reenter the command.

DEVICE WRITE LOCKED

Explanation: The disk drive is write locked.

User Action: Write enable the disk drive and reenter the command.

END OF FILE DETECTED

Explanation: The tape position was lost.

User Action: Retry the operation.

END OF TAPE DETECTED

Explanation: The tape position was lost.

User Action: Retry the operation.

END OF VOLUME DETECTED

Explanation: The tape position was lost.

User Action: Retry the operation.

FATAL HARDWARE ERROR

Explanation: A hardware malfunction occurred.

User Action: Retry the operation: if the error repeats, call field service.

HANDLER NOT RESIDENT

Explanation: The device driver (handler) was not loaded.

User Action: Load the appropriate device driver and retry the operation.

INSUFFICIENT POOL SPACE

Explanation: The operating system is overloaded.

User Action: Retry the operation.

PARITY ERROR ON DEVICE

Explanation: An uncorrectable read error occurred.

User Action: Retry the operation.

PRIVILEGE VIOLATION

Explanation: A device was mounted as Files-11 or is allocated to a different user.

User Action:

RSX-11M/M-PLUS Users: Dismount the disk and retry the operation.

IAS Users: Dismount the disk, mount it as a foreign device and retry the operation.

ERROR CODE IS <driver code>

Explanation: An I/O error which DSC cannot translate occurred.

User Action: Translate the error code and retry the operation, if possible.

ILLEGAL FUNCTION

Explanation: Tapes on drives have not been rewound since the system was booted.

User Action: Rewind the tapes, using the Rewind switch in a DSC command.

11.9.3 Stand-Alone DSC Messages

The following messages appear only with the stand-alone versions of DSC.

ILLEGAL VECTOR ADDRESS

Explanation: An illegal vector address was specified.

User Action: Correct the vector specification and reenter the command. Vector addresses must be a multiple of 4 and less than or equal to 374(8).

INVALID CSR ADDRESS

Explanation: A system trap occurred when the specified CSR address was referenced.

User Action: Correct the address and reenter the command.

INVALID TM02 ASSIGNMENT

Explanation: The TM02 switch applies only to TU16/TE16/TU45 tapes and cannot specify an assignment greater than seven.

User Action: Correct the error and reenter the command.

SPECIFIED UNIT NUMBER EXCEEDS MAX. OF 1

Explanation: The stand-alone version of DSC does not accept unit numbers greater than 1.

User Action: Correct the error and reenter the command. Specify the Unit switch if required.

CHAPTER 12

VOLUME PRESERVATION UTILITY (PRESRV)

12.1 INTRODUCTION TO PRESRV

The PDP-11 Volume Preservation Utility (PRESRV) is a stand-alone program that allows you to create copies of volumes. PRESRV copies volumes to and from the following device types:

- Disks (only those specifically indicated in Table 12-4)
- Magnetic tapes
- DECtapes
- Cassette tapes

PRESRV is most often used to preserve a system disk or master disk for backup purposes, usually on magnetic tape or another disk. Unlike BRU and DSC, PRESRV copies bad blocks. Tapes used by PRESRV to restore a disk must be the result of a PRESRV copy from disk.

PRESRV creates copies in three formats:

- Files-11
- Logical tape
- Image

Files-11 format saves only those blocks that have data allocated to them.

Logical tape format creates real or quasi-magnetic tape output. Logical tape format also puts a copy of PRESRV on the backup volume, so PRESRV can subsequently be bootstrapped from that volume.

Image mode format saves all blocks of the disk, regardless of allocation.

12.1.1 Files-11 Format

Files-11 format is a file structure that applies only to disks and DECtapes. When PRESRV is copying in Files-11 format, only blocks containing user data or Files-11 control information, called allocated blocks, are saved. Allocated blocks of the input volume are copied to identical output blocks of the output volume, and the remaining blocks of the output volume remain unchanged. Normally disks are read/written in Files-11 mode except as described below.

Input: To override a Files-11 format, specify the /TP switch for a logical tape format or the /IM switch for an image mode read. The /TP and /IM switches are described in Sections 12.3.2.2 and 12.3.2.3, respectively.

If the input and output devices are of an identical device type and neither a switch nor a file name are specified (DK0:=DK1:), PRESRV performs an image mode copy. Image mode is described in Section 12.1.3.

Output: If the input is in logical tape format, and the output device is a disk, the resulting output will be in Files-11 format. However, the /TP switch may be used to override this default (see Section 12.3.2.2). If the input and output devices are of the identical device type and you do not specify a switch or a file name, PRESRV defaults to an image mode copy. To override this default, specify either /TP or /FI for output in logical tape or Files-11 format, respectively.

If the input is a disk in Files-11 format and the output is a disk of a different device type, the output disk will be written in logical tape format (DP:NAME[/TP=DK:). In this case only a file name is required. If you wish output in Files-11 format, specify /FI on the output side of the command line to override the logical tape default (DP:/FI=DK:).

NOTE

When the output volume is in Files-11 format and is larger than the input volume, the Files-11 control information copied on the output volume makes it appear that the output volume has the same number of blocks as the input volume. This means that copying a small disk to a large disk (for example, an RK to an RP) in Files-11 format results in a loss of capacity on the larger volume.

Conversely, when the input volume is larger than the output volume (for example, an RP to an RK), the copy may fail if an attempt is made to copy to blocks that do not exist on the smaller volume.

A Files-11 copy from disk or DECtape is limited to a single input volume per copy command. Similarly, a Files-11 copy to an output disk or DECtape is also limited to a single output volume per copy command.

12.1.2 Logical Tape Format

PRESRV writes information in logical tape format or magnetic tapes and cassettes when the input device is a disk. PRESRV also writes in logical tape format when the input and output devices are different types of disks. When the input and output disks are of the same type, PRESRV defaults to an image mode copy unless you specify the /TP switch to copy in logical tape format.

When a logical tape copy is to be produced, the output side of the command string must include a file name, to identify the file created by the copy operation.

When the input volume is magnetic tape or cassette, PRESRV assumes logical tape format by default, unless the copy is to be made to a device of the same type as the input device. When the input volume is either disk or DECtape, logical tape format is recognized only when the /TP switch is specified on the input side of the command string. Otherwise, the input is assumed to be Files-11.

The following examples illustrate the concepts discussed above.

```
PRE>MT0:URFILE=DP0:
```

In this example, DP0: is read in Files-11 mode (/TP was not specified on input). The information is written on MT0: in logical tape format, as a file named URFILE. The output can occupy more than one magtape volume.

```
PRE>DT0:FILA=DK0:
```

In this example, DK0: is read in Files-11 format. The information is recorded on DT0: in logical tape format, as a file named FILA. FILA can occupy more than one DECtape.

NOTE

A logical tape copy (disk or magnetic tape) can occupy more than a single volume. A logical tape copy on magnetic tape can also contain copies of more than one disk. However, a new copy of a disk volume can not begin on any magtape volume other that the first.

12.1.3 **Image Format**

Image format is used to copy all physical blocks of a single input volume, regardless of content, to an output volume.

Image format transfers occur if the input and output devices are the same type and no other copy format is in effect. In other words, no format switches (for example, /FI or /TP) are specified, and no file name is specified on the output side of the command string. (Specifying a file name on the output side of the command string implies logical tape format.) For example:

```
PRE>MT0:=MT1:
```

NOTE

When executing an image format copy between two magtape devices, PRESRV checks to ensure that both devices are on-line before beginning the copy. If these checks fail, the operation will be aborted with the message

DEVICE OFFLINE

When EOT is sensed from the output tape, PRE sets up a counter to permit five additional blocks to be written on the output tape. However, each time an EOF is sensed on the input tape, the counter is reset to allow an additional five blocks to be written on the output tape following the EOF block.

After writing five blocks beyond EOT on output, or after EOF has been sensed on input and the output tape is already beyond EOT, PRE will issue a fatal error message and prompt for a new command:

```
FATAL ERROR ON OUTPUT DEVICE
PRE>
```

In some cases, it may be desirable to force a transfer in image format from disk or DECtape to a device of a different type. To do this, specify the /IM switch on the input side of the command string. For example:

```
PRE>MT0:TOM1=DK0:/IM
```

As a result of this command, a logical tape file named TOM1 is created on MT0:. TOM1 contains all the physical blocks of DK0:.

NOTES

1. This type of command is limited to just one input volume.

2. It is illegal to specify the /IM switch for magnetic tape input.

12.2 PRESRV OPERATING PROCEDURES

Because PRESRV is a stand-alone program, it is convenient to bootstrap a preserved volume to get PRESRV into memory. When PRESRV copies in logical tape format, it places the following information on the first output volume: a label block, a bootstrap block, and an image of PRESRV. After the bootstrap operation, the tape is positioned between the image of PRESRV and the data to be preserved. See Figure 12-1.

NOTE

When the output device is a magnetic tape and the first copy is to be created, the tape must be on-line and at load point (BOT) for a valid bootstrap. Otherwise, PRESRV will write a dummy bootstrap and the tape will not be bootable.

Bootstrapping the first volume of a logical tape brings into memory a stand-alone system with a running version of PRESRV. (See the CAUTION in Section 12.2.1.) You can then type a command string to perform any desired PRESRV function.

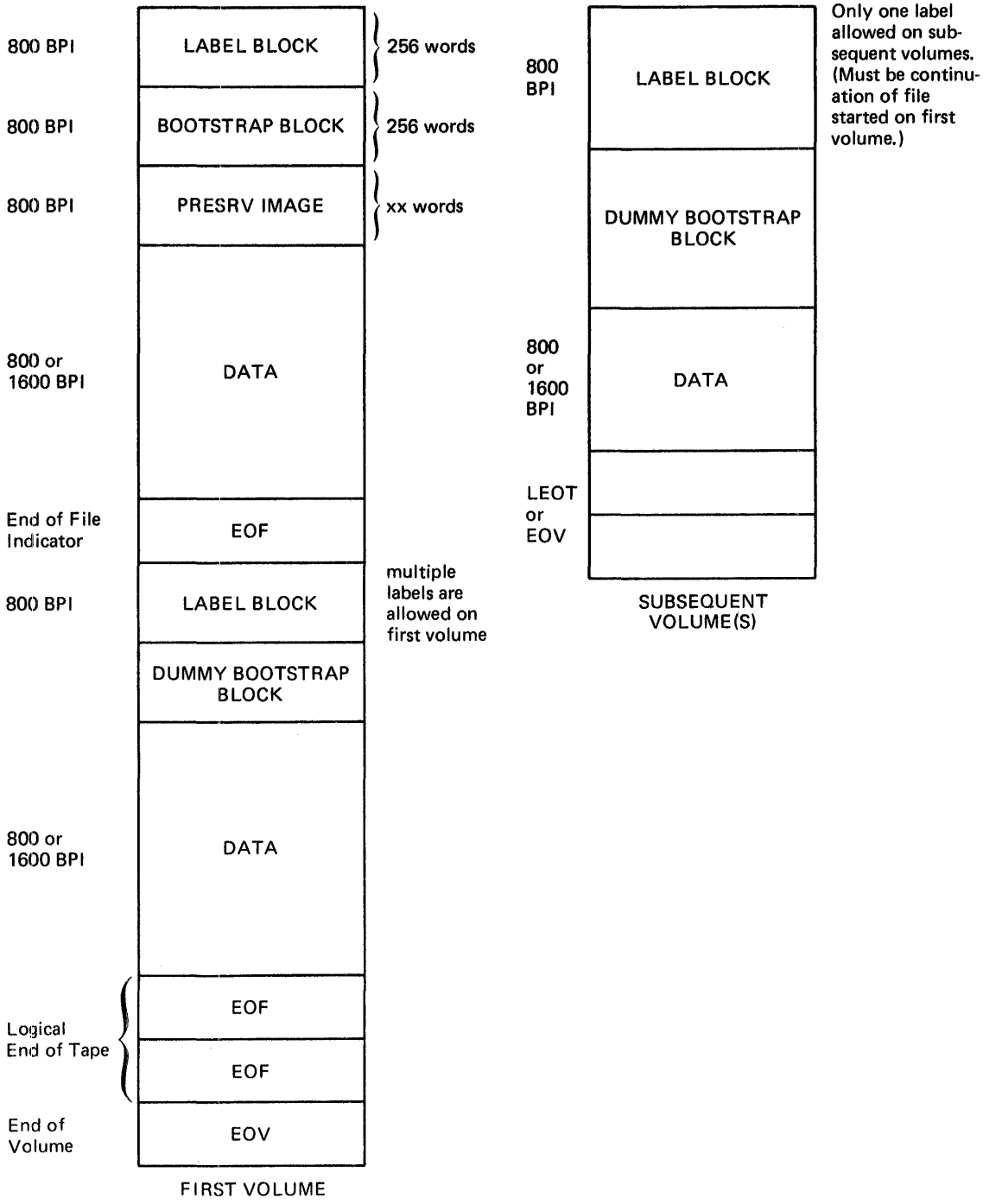


Figure 12-1 Logical Tape Format

To create a volume from which PRESRV can be bootstrapped, specify a copy operation from one type of device to another type of device. For example:

```
PRE>DT0:PRESRV=DK1:
```

The copy operation can be stopped before it is complete, because the necessary bootstrap information will have been copied into the first few blocks of the resulting logical tape.

12.2.1 **Bootstrap Procedure**

To run PRESRV, mount the device to be copied and bootstrap the device on which PRESRV is recorded. Bootstrapping is accomplished by either setting the correct address in the console register, loading the address, and then depressing the START switch; or using the RSX-11M BOO command (on an RSX-11M system). The BOO command begins a software bootstrap operation.

There are many different methods of initiating a hardware bootstrap operation, each of which is system dependent.

For details on bootstrap procedures, consult your system documentation.

When PRESRV is bootstrapped, it displays an identification message of the form:

```
RSX-11S Vxx
VOLUME PRESERVATION PROGRAM- Vyy
FOR HELP TYPE /HE
nn. BLOCKS AVAILABLE FOR BUFFERING

PRE>
```

PRESRV is built as a task running under the RSX-11S Executive and thus displays the version of the stand-alone executive in use, the version number of PRESRV, and the number of 256-word blocks (nn.) available for buffering. You must use the number of blocks available for buffering (displayed in the identification message) to determine if a given PRESRV media copy has sufficient buffering space.

If the total space required for input, output and verification (optional) buffers is greater than nn. blocks, PRESRV responds with the following message:

```
INSUFFICIENT BUFFER SPACE
```

Change the blocking factor(s) and retry the operation.

The number of blocks available for buffering should be kept in mind when using the blocking switch (/BL) described in Section 12.3.2.4. An error message is issued if the combined blocking factors for the input and output volumes exceed the number of blocks available.

CAUTION

PRESRV contains predefined CSR and vector addresses for devices. If these addresses differ from those established in your system, or if your system includes TU16 9-track magtapes (instead of TU10-type tapes) or RP04/06 disks (instead of RP02/03 disks) you must modify PRESRV's standard device information. See Section 12.3.5 for a summary of PRESRV's CSR and vector addresses.

To tailor PRESRV to conform to a specific system, bootstrap the standard version of PRESRV into memory and then use the /CSR, /VEC, and /TM02 switches, described in Sections 12.3.5.1, 12.3.5.3, and 12.3.5.4 respectively, as required. Then create a logical tape copy, which will contain the modified bootable version of PRESRV tailored to your system's configuration.

The following example shows how to modify in memory the vector address of an RP04 disk pack to 254. The vector address applies to both DB0: and DB1:, and to any other DB-type device reassigned later to either DB0: or DB1: by means of the /UNIT switch (see Section 12.3.5.2).

CSR and TM02 values can be changed in similar fashion.

```
PRE>DB0:/VEC=254
```

The next example shows how to create a bootable image of the above modified version of PRESRV.

```
PRE>DT0:ABBA/TP=DK0:
```

NOTE

This copy operation need not run to completion, because the first few blocks on DT0: are sufficient to bootstrap the tailored version of PRESRV into memory.

12.2.2 Tape Handling

PRESRV supports forms of multifile volumes and multivolume files on magnetic tape, cassettes, DECtape, and disk. Table 12-1 indicates the permissible combinations of files and volumes in logical tape format.

Table 12-1
Legal Multifile/Volume Format in Logical Tape

Magtape/Cassette	DECtape/Disk
Multiple files on first volume only. Last file can extend to second and succeeding volumes.	One file only, which can extend to second and succeeding volumes.

Each file on a magnetic tape or cassette is terminated with an end-of-file (EOF) tape mark. A file copied in logical tape format is prefixed with a file label that identifies the file.

NOTE

Regardless of the device type, PRESRV records within each file enough information to restore a disk in FILES-11 format and to recognize the end of the file.

12.2.2.1 **Logical Tape Input from Magnetic Tape or Cassette** - PRESRV determines where to begin reading a tape as follows:

If a file label is specified, PRESRV rewinds the input tape and searches for the file label. If a file label is not specified, PRESRV assumes that the tape is correctly positioned to read the first file or next file (or file segment) of a volume set. To read the first file of a volume set, without specifying its file label, position the first volume to Beginning-of-Tape (BOT).

After processing any file, PRESRV positions the tape to read the next file of a set.

When PRESRV reaches the end of a tape volume by sensing two successive EOF tape marks it requests that the next input volume be mounted. When you mount the next volume, PRESRV assumes the volume is positioned at BOT. You are responsible for correctly positioning the tape to BOT, and for assigning the same unit number as the preceding volume.

12.2.2.2 **Logical Tape Output to Magnetic Tape** - To write the first copy of a set to magnetic tape or cassette, position the volume at BOT. If output is to be stored on two or more volumes of a set, the first volume mounted must be the first volume of the set.

After a copy operation is finished, PRESRV positions the tape to record a new file on the current volume by writing two EOF marks and then backspacing over the second EOF mark. An exception occurs when the copy ends on a volume that is not the first volume. In this case only, the backspace does not happen: any file added at that point is inaccessible to PRESRV. In the normal process of recording a new file, PRESRV overwrites the second EOF indicator that marks the current end of volume.

When PRESRV reaches the end of an output volume, it prints a message requesting that the next output volume be mounted. PRESRV checks to ensure that the new tape is at BOT. If not at BOT, the mount message is repeated until it is. You must ensure that the volume is correctly positioned, and assigned the same unit number as the preceding volume.

Time may be saved by pre-mounting magnetic tapes (except cassettes), in the following way:

1. Pre-mount the next volume on a different unit, and assign it a different unit number than the current volume's unit number. Make sure it is positioned to the load point (BOT).

2. When PRESRV prints the request to mount the next volume, switch the unit numbers of the old and new volumes.

3. The old volume can now be rewound as PRESRV processes the new volume.

For output to magnetic tape, PRESRV assumes that the volume is correctly positioned at Beginning of Tape (BOT) initially and after the EOF tape mark subsequently.

On input from magnetic tape, PRESRV functions differently depending on whether a file label is specified in the command line. If a file label is specified, the input volume is rewound and searched for the named file. If no file label is specified, the next file on the tape is read.

A file can extend across a volume boundary, but no new files can begin on the second or subsequent volume of a multivolume file.

NOTE

Be sure that tapes are rewound when PRESRV requests that a new volume be mounted. All tapes must be mounted, and assigned the same unit number.

12.2.3 Disk and DECtape Handling

PRESRV processes disks and DECtapes as described in the following sections.

12.2.3.1 **Input from Disk and DECtape** - When disk or DECtape input is in logical tape format, PRESRV processes as many volumes as necessary to copy the complete input file. However, when the input disk or DECtape is in Files-11 or image format, PRESRV processes only a single input volume.

12.2.3.2 **Output to Disk and DECtape** - PRESRV can record only one file on a disk or DECtape output volume, regardless of format. This restriction exists because the first access to a disk or DECtape is to logical block 0. However, PRESRV can create a single logical tape copy that extends over more than one volume.

When the output volume is disk or DECtape, and Files-11 or image format is specified or implied, PRESRV prints an error message if more than one volume is needed.

NOTE

Disks and DECtapes can be pre-mounted by following the general guidelines suggested in Section 12.2.2, above.

12.3 PRESRV COMMAND LINE AND SWITCHES

The PRESRV command line has the following format:

outdev:[label]/switch1.../switchn=indev:[label]/switch1.../switchn

outdev and indev

Specify the physical devices on which the output and input volumes are mounted, for example, DK0: or DT1:. The device name consists of two ASCII characters followed by an optional 1- or 2-digit octal unit number 0 through 7. If the unit number is omitted, 0 is assumed. The unit number is always 0 or 1. The /UNIT switch must be used to reference units other than 0 or 1.

label

Specifies a label for the file when a volume is written as a logical tape. It is used to identify files on logical tapes. See Section 12.2.2. The label can consist of up to nine alphanumeric characters. A label must be specified when the output is a logical tape. It is optional for input. The label parameter is applicable only for files on a logical tape.

switch

Specifies an ASCII name identifying the switch option desired. Option switches are summarized in Table 12-2.

Table 12-2
Summary of PRESRV Switches

Category	Switch	Function
List	/HE	Causes a summary of PRESRV commands, switches, and defaults to be printed on the terminal.
	/LI	Causes a list of all files on a logical tape volume to be printed on the terminal.

(continued on next page)

Table 12-2 (Cont.)
Summary of PRESRV Switches

Category	Switch	Function
Format	/FI	On input, indicates Files-11 format for reading a disk volume overriding implied Image Mode. On output, overrides the /TP default or implied /IM.
	/IM	Indicates image mode; that is, transfer all physical blocks of an input disk. Never used with tape. /IM is used only on the input side, overriding default Files-11 mode.
	/TP	Indicates a logical tape format output volume (which overrides the default image mode when device types are identical). /TP also overrides the default Files-11 mode on an input disk.
	/BL:nnn	Specifies a blocking factor for I/O operations. Octal is assumed; decimal is indicated by a trailing period (.). This value applies only to the side of the command in which it is specified. If /BL is not specified, the default blocking factor is used. (See Section 12.3.2.4.)
Administrative	/ER	Indicates that the PRESRV operation is to be terminated if an I/O error occurs.
	/RW	Causes a cassette or magnetic tape volume (either input or output) to be rewound at completion of PRESRV operation. NOTE PRESRV rewinds only the last volume of a multi-volume set when it is completely finished. Intermediate volumes must be rewound by the user.
	/VE	Causes the output volume to be verified after it is written.
Magnetic Tape	/20	Indicates a recording density of 200 bpi for 7-track tape. (Image transfers only.)

(continued on next page)

Table 12-2 (Cont.)
Summary of PRESRV Switches

Category	Switch	Function
Magnetic Tape (Cont.)	/55	Indicates a recording density of 556 bpi for 7-track tape. (Image transfers only.)
	/80	Indicates a recording density of 800 bpi for 7-track tape. (Image transfers only.)
	/PE	Indicates a recording density of 1600 bpi (phase encoded). Can be used only for TE16, TU16, TU45, and TU77 tape drives. If /PE is not specified, a density of 800 bpi is assumed. Do not specify /PE for the TM03 formatter.
Stand-alone PRESRV	/CSR	Changes the control status register for a device type.
	/UNIT	Sets the physical unit number of either PRESRV unit 0 or 1 to a value from 2 through 7.
	/VEC	Changes the interrupt vector address for a device type.
	/TM02	Sets the unit number of the TM02/TM03 formatter on an RH controller for a specific TU16 magnetic tape unit. NOTE /CSR, /UNIT, /VEC, and /TM02 can all be specified in the same command.
	/TIM	Specifies the correct time of day to the system.
	/BO	Causes PRESRV to bootstrap the system from the specified device. NOTE Only Physical Unit 0 may be booted with this command if the device is a disk.
	/FO	Indicates that PRESRV is to format the output disk pack.

PRESRV options are specified by means of switches included in the command string or by stand-alone switches. Switches fall into five categories according to the functions they perform.

- List switches
- Format and blocking switches
- Administrative switches
- Magnetic tape switches
- Stand-alone PRESRV switches

12.3.1 List Switches (/HE and /LI)

The list switches are entered one at a time in response to the prompt (PRE>). Each switch is processed immediately by PRESRV. /HE obtains a printed summary of PRESRV commands, switches, and defaults. /LI obtains a list of all files of a logical tape volume. If the set consists of more than a single volume, each of the volumes must be listed separately. When the designated device is a magtape or cassette, the /LI switch causes an automatic rewind, followed by a listing of the file label and a description of all the files on that volume. On magnetic tape, if the two EOF tape marks indicating End of Information are sensed before physical EOT, PRESRV performs a backspace. If the current volume is the first of the set, the tape is then positioned so that a new file may be added.

The formats of the /HE and /LI switches follow.

```
PRE>/HE

PRE>dev:/LI
```

dev:

Indicates the device and unit number on which the volume is mounted.

NOTE:

The /LI switch references only a single volume. Since all files contained in a set of volumes begin on the first volume of a tape set, listing the first volume lists all of the files in the set. Subsequent volumes will list only their contents.

12.3.2 Format and Blocking Switches

Format switches specify the format of the input and output volumes. Three switches specify format.

Switch	Function
/FI	Specifies Files-11 format.
/IM	Specifies that a disk or DECtape volume is to be read in image format.
/TP	Specifies logical tape format.

The /BL switch indicates the blocking factor.

If format switches are not included in the command line, PRESRV uses the following default values:

For the input portion of the command line, tape devices (magnetic and cassette tape) default to logical tape format, while Files-11 devices (disk and DECtape) default to Files-11 format.

For the output portion of the command line, logical tape format is used by default.

There are two exceptions:

1. If the input and output devices have the same device type and no format switches are included in the command line, the transfer is in image mode. The output volume will be an exact image of the input volume, regardless of the format of the input volume. Only a single volume will be copied.

2. If the input volume is being read in logical tape format, and output is to disk or DECtape, the data portion of the input file is written into the logical blocks corresponding to the blocks the data occupied on the volume from which it was originally copied. Output is limited to a single output volume.

Default blocking factors are provided with the description of the /BL switch in Section 3.2.4. Each block is 256 (decimal) words.

12.3.2.1 **Files-11 Format Switch (/FI)** - The Files-11 format switch (/FI) can be used on either the input or the output side of the command line.

On the input side, /FI indicates that the input Files-11 volume is to be read in Files-11 format, overriding a default image-mode transfer. Only allocated blocks will be transferred.

The /FI switch is particularly useful for input volumes if an image or logical tape transfer fails due to an unreadable block on a Files-11 device that is being preserved. Because Files-11 transfers result in copying allocated blocks only, use of the /FI switch results in a successful transfer if the unreadable block is not allocated.

NOTE

If the input volume is not in Files-11 format, PRESRV generates an error message.

On the output side, /FI indicates that the output Files-11 volume is to be written in Files-11 format, overriding a default logical tape format or implied image mode transfer.

Examples

```
PRE>DK0:/FI=DK1:/FI
```

Use of the /FI switches overrides the default for this case, which is an image transfer. A Files-11 copy of DK1: is produced on DK0:.

```
PRE>DK0:STEVE/TP=DK1:
```

In this case, DK0: will contain the file labeled STEVE, in logical tape format. STEVE will contain only blocks that were allocated on DK1:.

```
PRE>DK0:/FI=DT:
```

Use of the /FI switch indicates that the transfer is to result in DK0: containing a Files-11 copy of the information on DECtape. The default for the output for this case is logical tape format. Using the /FI switch in this manner is not recommended, because the resulting copy limits the capacity of DK0: to 578 blocks (the capacity of the DECtape).

12.3.2.2 **Logical Tape Format Switch (/TP)** - The logical tape format switch (/TP) indicates the following:

1. The input volume is to be read in logical tape format, or
2. The output volume is to be written in logical tape format.

When the PRESRV input is in logical tape format, the following message is printed on the terminal.

```
MOUNT INPUT VOLUME nn AND TYPE <CR>
```

Mount the physical volume and then press the RETURN key to continue the PRESRV operation. If the volume is already mounted and correctly positioned, press RETURN only.

When the PRESRV output is in logical tape format, the following message is printed on the terminal.

```
MOUNT OUTPUT VOLUME nn AND TYPE <CR>
```

The user response is the same as in the case of the MOUNT INPUT VOLUME message.

NOTE

Never use /TP to request that a tape be read or written in logical tape format (the fact that the device is a tape implies logical tape) unless input and output are both magtapes

NOTE

When PRESRV displays the message requesting that a tape be mounted and the user types <CR> in response, PRESRV will check to ensure that the tape is, in fact, on-line. If that check fails, the message and the subsequent check will be repeated until the check is successful.

Examples

```
PRE>DK0:LAB/TP=DK1:
```

The volume in DK1: is read in Files-11 mode and is to be preserved in logical tape format in File LAB on DK0:. Use of the /TP switch overrides the default (which is an image transfer in this case because device types are identical). A label must be specified for the copy file.

```
PRE>DK1:=DK0:LAB/TP
or
PRE>DK1:=DK0:/TP
```

The file LAB on DK0: is in logical tape format, if it is the copy file created in the previous example. DK0: is now restored on DK1: in Files-11 mode.

NOTE

When the output format is to be logical tape and the output device is a disk, that disk must be physical drive 0.

12.3.2.3 **Image Mode Switch (/IM)** - The image mode switch overrides the Files-11 default in copying disk-structured volumes. By using the /IM switch, an operator can copy any disk that is formatted compatibly with DIGITAL's hardware. In image mode, every block on the input volume is copied to the output file.

The image mode switch can also be used to override the Files-11 default, as shown in the first of the following examples. This allows a logical tape to be used to restore a disk that was originally in any format compatible with DIGITAL hardware.

Examples

```
PRE>MT0:AFIL=DP0:/IM
```

Copy from disk to tape.

```
PRE>DP0:=MT0:AFIL
```

Restore disk from tape.

12.3.2.4 **Blocking Switch (/BL:nn)** - The blocking switch (/BL:nn) indicates the following:

1. The blocking factor to be used when reading a disk or DECtape volume.

2. The blocking factor to be used when creating a logical tape or disk or DECtape output volume.

Specify the blocking switch to save time. Use the blocking switch to request that the specified number of 256-word blocks be read as a unit from a disk or DECtape, or be written as a unit to either a disk or DECtape, or be written in logical tape format.

Note, however, that the label of a logical tape specifies the blocking factor that was used when the file was created. The /BL switch cannot be used to override that value when the file is used as input.

The blocking switch has the following format:

```
/BL:nn (octal)
or
/BL:nn. (decimal)
```

nn

Indicates the number of 256.-word segments to be written as a physical block on tape or to be read from or written to disk to fill or empty the corresponding PRESRV buffer. The value of nn is limited by the number of blocks available, which is printed (in decimal) on the terminal when PRESRV is initiated.

/BL can be used on both sides of a command line. The associated values need not be identical.

Each device has a default blocking factor that is used when no blocking switch is specified and if the buffer space is available to perform the default blocking. The defaults are shown in Table 12-3.

Table 12-3
Default Blocking Factors

Device	Format[1]	Default Block Size[2]	Max. Block Capacity (Decimal)
DK	BS	12.	4800 (RK05 Cartridge Disk)
DF	BS	8.	1024 per platter (RS11 Fixed Head Disk)
DP	BS	10.	40000 (RP02), 80000 (RP03) Disk Packs
DS	BS	8.	1024 (RS03), 2048 (RS04) Fixed Head Disks
DT	BS	2.	578 (TC11/TU56) DECtape
DB	BS	22.	167200 (RP04 Disk Pack)
MT	LT	1.	Variable TE10 or TS03 Magtape
MM	LT	1.	Variable TE16, TU11, TU45, or TU77 Magtape
CT	LT	1.	Variable TA11/TU60 Cassette
DX	BS	6.	494 (RX11/RX01) Floppy Disk

1 BS - Block Structured
LT - Logical Tape

2 Decimal point indicates decimal value.

The defaults provided in Table 12-3 can be overridden by any of the following means:

1. For disks and DECtape, the /BL switch can be used with the input or output file specifier to establish a blocking factor.

2. For logical tape output, the /BL switch can be used to establish the blocking factor.

3. For magnetic tape image transfers, PRESRV calculates a blocking factor that uses all available buffering space.

NOTES

1. The /BL switch has no effect when the input device is a logical tape. The blocking factor is the same as that used when the tape was created.

2. The blocking factor used for disk type devices should be a submultiple of the number of blocks on the volume (for example, 4800 for RK) when copying in image mode.

Refer to Section 12.4 for the blocking calculations used to match available buffering space to blocking specifications for a file.

Example

```
PRE>DK0:/BL:20.=MT0:
```

Twenty 256-word blocks are transferred in each I/O operation. Forty blocks must be available for use in blocking (assuming that the tape was created with /BL:20).

12.3.3 Administrative Switches

Three administrative switches are provided.

Switch	Function
/ER	Aborts the operation if an I/O error occurs.
/RW	Rewinds a tape after PRESRV completes.
/VE	Compares the output volume(s) created by PRESRV with the volume(s) submitted as input to a copy operation.

12.3.3.1 **Abort on I/O Error Switch (/ER)** - If the abort on I/O error switch (/ER) is attached to a file specifier, an unrecoverable I/O error causes the termination of the current PRESRV operation. After issuing an error message, PRESRV prints

```
PRE>
```

and is ready to accept another command.

The /ER switch applies to either input or output file specifiers. For example:

```
PRE>DK1:/ER=DK0:
```

If a write error occurs and the /ER switch has not been specified, PRESRV responds with the following message:

```
VOLUME FAULTY - "R" TO RETRY, "C" TO CONTINUE>
```

Type C to continue or R to retry the operation.

NOTE

If you type the ALTMODE key when PRESRV is ready to accept input, the current operation is aborted.

12.3.3.2 **Rewind at Completion Switch (/RW)** - The rewind at completion switch (/RW) applies to magnetic tape and cassette tape volumes. It indicates that the volume is to be rewound after PRESRV has completed the specified transfer.

The /RW switch applies to input and output file specifiers.

NOTE

Only the last volume of a multivolume input or output set is rewound.

12.3.3.3 **Verification Switch (/VE)** - The verification switch (/VE) causes PRESRV to verify the copy operation. For example, PRESRV compares the output device to the input device to make sure there are no differences.

When the verification operation begins, PRESRV responds with the following message:

```
**BEGIN VERIFICATION**
```

This will cause a second pass over the input and output media so that every block written is read back and compared with the corresponding block on the input device. When verification is requested, twice the number of output buffers are needed. The /VE switch may appear on either side of the command string. For example:

```
PRE>DK1:/VE=DK0:
PRE>DK1:=DK0:/VE
```

If the verification is successful, PRESRV prints

```
PRE>
```

and will accept another command.

If the comparison fails, PRESRV prints the following error message:

```
VERIFICATION ERROR m,nnnnnn
```

The number printed is the octal logical block number on the output device where the error was detected, where "nnnnnn" is the low-order word and "m" is the high-order word of the two-word LBN number. Thus

```
5,031275
```

indicates LBN 340669 (the last LBN on an RP06). Retry, continue, or abort the operation.

If the volume to be verified is faulty, PRESRV responds with the following error message whenever a verification error occurs:

```
VOLUME FAULTY - "R" TO RETRY, "C" TO CONTINUE>
```

Type R to retry or C to continue the operation.

NOTE

If the user types the ALTMODE key when PRESRV is ready to accept input, the current operation is aborted.

12.3.4 Magnetic Tape Switches

Four switches are provided to specify magnetic tape options. The default recording density is 800 bpi. Seven-channel tapes are always recorded in core dump format.

Switch	Function
/20	Specifies 200 bpi for 7-track tape.
/55	Specifies 556 bpi for 7-track tape.
/80	Specifies 800 bpi for 7-track tape.
/PE	Specifies 1600 bpi (phase encoded) for TU16 tape.

NOTES

1. File labels, bootstraps, and system images are always written at 800 bpi on logical tapes. Therefore, logical tapes written with the /PE switch must be copied /PE specified on both sides of the command. For example:

   ```
   PRE>MM1:/PE=MM0:/PE
   ```

2. Logical tapes written on a 7-track drive are bootable only on a 7-track drive. Logical tapes written on a 9-track drive are bootable only on a 9-track drive.

12.3.4.1 **200 bpi Switch (/20)** - The 200 bpi switch (/20) indicates either that an input magnetic tape volume has been recorded at 200 bits per inch, or that an output magnetic tape volume is to be recorded at 200 bits per inch. The tape drive must be 7-track and an image copy must be performed. Thus:

```
PRE>MT0:/20/VE=MT1:/20
```

produces and verifies an exact copy of MT1: on MT0: at 200 bpi.

12.3.4.2 **556 bpi Switch (/55)** - The 556 bpi switch (/55) indicates either that the input magnetic tape volume has been recorded at 556 bits per inch, or that the output magnetic tape volume is to be recorded at 556 bits per inch. The tape drive must be 7-track and an image copy must be performed.

12.3.4.3 **800 bpi switch (/80)** - The 800 bpi switch (/80) indicates either that an input magnetic tape volume has been recorded at 800 bits per inch or that an output magnetic tape volume is to be recorded at 800 bits per inch. This switch can be specified for image transfers on 7-track magnetic tape only.

12.3.4.4 **Phase Encoded Switch (/PE)** - The phase encoded switch (/PE) indicates either that an input magnetic tape volume is recorded at 1600 bits per inch (phase encoded) or that an output magnetic tape volume is to be recorded at 1600 bits per inch. The volume must be mounted on a TU16 (or equivalent) drive.

NOTE

This switch cannot be used with logical tapes if a TM03 formatter is involved.

12.3.5 Stand-Alone PRESRV Switches

Once a logical tape is booted into memory, PRESRV is running. Seven stand-alone switches are available with PRESRV. Four of the switches specify physical device address information.

Switch	Function
/CSR	Specifies a control status register.
/UNIT	Sets the unit number for a device to 2 through 7.
/VEC	Specifies an interrupt vector for a device.
/TM02	Specifies the unit number of a TM02 formatter on the RH controller.

These four switches can appear together, in any combination, in a single command line. This has no effect on their stand-alone status; that is, the action specified by each switch is executed before another command is accepted.

PRESRV is distributed with the control status register (CSR) and vector addresses shown in Table 12-4.

Two units (0 and 1) are generated for each device type.

If the generated CSR or vector address does not correspond to the actual hardware configuration in use, the /CSR and /VEC switches can be used to correct values generated by PRESRV (see Sections 12.3.5.1 and 12.3.5.3).

The three remaining stand-alone switches are listed below.

Switch	Function
/TIM	Sets the current time and date.
/BO	Boots a monitor from a specified device following completion of PRESRV operations.
/FO	Causes PRESRV to format a disk.

Table 12-4
CSR and Vector Addresses

Device	CSR	Vector[1]
CT	177500	260
DB	176700	300*
DF	177460	204
DK	177404	220
DP	176714	254
DS	172040	310*
DT	177342	214
MM	172440	320*
MT	172522	224
DX	177170	264

[1] An asterisk (*) indicates that this is not a normal vector.

12.3.5.1 **Control Status Register Switch (/CSR)** - The control status register switch (/CSR) specifies the correct CSR for a device type. The switch is typed in response to the PRE> prompt and has the following format.

dev:/CSR=nnnnnn

dev

Specifies the device name and logical unit number of the device whose CSR is to be changed.

nnnnnn

Specifies the octal address of the CSR used by the device.

The new CSR applies to all devices of the same type as the device specified in the command.

12.3.5.2 **Physical Unit Number Switch (/UNIT)** - The physical unit number switch (/UNIT) allows the user to copy to or from a device designated as a unit other than unit 0 or 1 of that device type. Type the /UNIT switch in response to the PRE> prompt, in the following format:

dev:/UNIT=n

dev

Specifies the device name and unit number (0 or 1).

n

Specifies the unit number of the same device type that will be referenced each time "dev" (as specified above) appears in a command line.

For example:

```
PRE>DB1:/UNIT=3
```

causes the RP04 assigned as unit 3 to be referenced each time DB1 appears in a command.

The /UNIT switch can also be used in conjunction with the /TM02 switch, to reference MM2: through MM7: on systems with mixed MASSBUS peripherals.

12.3.5.3 **Vector Address Switch (/VEC)** - The vector address switch (/VEC) changes the interrupt vector for a device type. The switch is typed in response to the PRE> prompt. It has the following format:

```
dev:/VEC=nnnnnn
```

dev

Specifies the device name and unit number for which the vector address is to be changed.

nnnnnn

Specifies the new octal vector address.

NOTE

When a tape created by online PRESRV is bootstrapped, if the vector address is incorrect, the first command issued after the prompt should be to set the appropriate vector address; for example:

```
PRE>MM0:/VEC=224
```

The new vector address applies to all devices of the specified type.

12.3.5.4 **Set TM02 Unit Number Switch (/TM02)** - The /TM02 switch changes the unit number of the TM02/TM03 formatter for a TE16, TU16, TU45, or TU77 attached to an RH11 or RH70 controller. The TM02 formatter for the tape is normally unit 0.

The format is:

```
dev:/TM02=n
```

dev

Specifies the device whose TM02/TM03 formatter position on the RH Controller is to be changed.

n

Specifies the unit number to which the TM02/TM03 formatter is to be changed.

Example

```
PRE>MM0:/TM02=2
```

The new TM02 value applies to only the device specified in the command.

12.3.5.5 **Time Switch (/TIM)** - The time switch (/TIM) sets the time of day for the PRESRV system. It is typed in response to the PRE> prompt and has the following format:

```
/TIM=mm/dd/yy hh:mm:ss
```

mm

Specifies the month.

dd

Specifies the day of the month.

yy

Specifies the year.

hh

Specifies hours.

mm

Specifies minutes.

ss

If present, specifies seconds.

All values are in decimal (no decimal point is specified).

12.3.5.6 **BOOT Switch (/BO)** - The PRESRV program is a stand-alone program that runs independently of any other system. When PRESRV prompts, the user may bootstrap a monitor from a specified device by using the bootstrap (/BO) switch. For example:

```
PRE>DK0:/BO
system monitor identification
```

The above command string causes a monitor to be read from DK0: and loaded into memory. The /BO switch is a stand-alone switch and must be typed as a separate command. Only unit 0 of the specified device type can be bootstrapped in this way, unless the device to be booted is a tape.

12.3.5.7 **Format Disk Switch (/FO)** - RK05, RP02, RP03, RP04, RP05, and RP06 disks delivered from the factory must be prepared to store data before they can be used. This preparation process is called "formatting", and consists of writing sector identification timing and information on the disk. PRESRV performs this operation in response to the /FO switch.

Formatting can be specified as a separate operation or in conjunction with a copy operation. If a copy operation has been specified, the format pass is made before the copy is made. The /FO switch can be specified only on the output side of the command string.

Examples

```
PRE>DK0:/FO

PRE>DP0:/FO/VE=MT0:DISK
```

When formatting starts, the message

```
** BEGIN FORMATTING **
```

is displayed. When formatting has been completed, the message

```
** END FORMATTING **
```

is displayed.

Before RP02 or RP03 disks can be formatted, formatting must be enabled with the format enable switch on the RP11 controller. This switch is located in the third bay from the bottom on the controller. The front cover must be removed to expose the switch. PRESRV prompts the user to enable and disable formatting at the proper times with the following messages:

```
ENABLE FORMATTING WITH SWITCH ON CONTROLLER.  TYPE CR WHEN READY>
DISABLE FORMATTING WITH SWITCH ON CONTROLLER.  TYPE CR WHEN READY>
```

When the switch is set properly, press the RETURN key to tell PRESRV to continue.

Once a disk is formatted, it requires no reformatting unless the control information has been destroyed.

12.4 COMPUTING THE NUMBER OF BLOCKS REQUIRED FOR BUFFERING

When PRESRV is initiated, it prints at the terminal the maximum number of 256-word blocks available for buffering. (This number is a decimal number.) These blocks are required for data transfer and verification operations. The user must refer to the number (nn.) to determine whether a given media copy has sufficient buffering space.

The space required depends on blocking factors and use of the verification switch. Default block sizes for devices supported by PRESRV are provided in Table 12-3. The maximum number of buffer blocks depends on the memory size of the system.

The examples below illustrate the method for determining the blocks required for PRESRV copy operations. The method is to establish the total number of blocks required for input, output, and verification.

Example 1

```
PRE>MT0:ABC=DK0:
```

Thirteen blocks are required to create a logical tape file (labeled ABC) on magtape unit 0, from an RK disk, using default blocking factors.

Operation	Blocks Required
Input	12. (DK default block size)
Output	1. (MT default block size)
Verification	0. (no verification specified)
Total	13.

Example 2

```
PRE>MT0:DEF/BL:12.=DK0
```

Twenty-four blocks are required to create a logical tape file (labeled DEF) on magtape unit 0. The default blocking factor is used for the input device, while physical blocks of 3072 words each are written to MT0: (12 x 256). This results in fewer blocks on MT0:, and requires less execution time because the number of tape write operations is reduced.

The blocking factor (12, decimal) will be recorded in file DEF's file label.

Operation	Blocks Required
Input	12. (DK default block size)
Output	12. (blocking factor specified for MT0:)
Verification	0 (no verification specified)
Total	24.

Example 3

```
PRE>DK0:=MT0:XYZ
```

Twelve blocks, plus the record size of file XYZ, are required to copy file XYZ from MT0: to unit DK0: and to have DK0: formatted as a logical tape.

Operation	Blocks Required
Input	Record size specified in file XYZ's label
Output	12. (DK default block size)
Verification	0
Total	12. plus record size of file XYZ.

Example 4

```
PRE>MT0:IJK/VE/BL:12=DK0:/BL:4
```

Twenty-eight blocks are required to create the file IJK on logical tape MT0: from DK0: with verification specified.

Operation	Blocks Required
Input	4. (specified block size DK)
Output	12. (specified block size MT)
Verification	12. (re-read block from MT)
Total	28.

Example 5

```
PRE>DK0:/VE=MT0:IJK
```

Twenty-four blocks, plus the record size of file IJK, are required to copy file IJK from MT0: to DK0:, with verification specified.

Operation	Blocks Required
Input	Record size of file IJK
Output	12. (DK default block size)
Verification	12. (Re-read block from DK0:)
Total	24. plus record size of file IJK.

Example 6

```
PRE>MM0:=MM1:
```

The above command results in an image copy. The image copy uses all available memory buffer space in creating blocks on the output volume. If any input record is too large, the transfer is aborted. If verification is specified, half of the available memory buffer space is used to form blocks for the output volume. The other half is required for verification.

12.5 PRESRV ERROR MESSAGES

As it executes, PRESRV checks to determine the success of the operation. If an error is detected, PRESRV prints one of the messages listed below.

BITMAP FILE HDR READ ERROR

Explanation: An attempt has been made to perform a copy in Files-11 mode, but the input volume is not in Files-11 format or is not ready.

User Action: Retype the command using either the /IM or /TP switch.

CONFLICTING SWITCHES

Explanation: Two conflicting switches were specified in the command string. For example, /TP/FI.

User Action: Specify the correct switch(es).

DEVICE NOT FORMATTABLE

Explanation: The specified device is not accepted by PRESRV. Devices accepted by PRESRV for formatting are RK03/05, RP02/03, RP04, RP05, and RP06.

User Action: Specify the correct device mnemonic.

DEVICE OFFLINE

Explanation: The specified device is not in the configuration hardware, the Control Status Register (/CSR) switch is improperly set, or the device is not on-line.

User Action: Set the /CSR and vector (/VEC) addresses for the device to the proper addresses or put the device on-line. Retype the command.

DICTIONARY READ ERROR

Explanation: An error occurred in reading a dictionary (storage bitmap equivalent) from logical tape.

User Action: Retry or abort the operation.

DRIVE NOT READY

Explanation: The specified drive is not ready or up to speed.

User Action: Wait until the drive is ready and retype the command.

ERROR IN READING COMMAND

Explanation: An error occurred in input to the terminal or CTRL Z (^Z) was illegally typed.

User Action: Retype the command.

EXPECTED EOF NOT FOUND

Explanation: An End-Of-File (EOF) mark was read from the input volume but not from the output volume during the verification of a tape image copy.

User Action: Retry the operation.

FATAL ERROR ON INPUT DEVICE

Explanation: An error occurred while reading the input volume during a tape image copy.

User Action: Retry the operation.

FATAL ERROR ON OUTPUT DEVICE

Explanation: An error occurred while reading or writing the output volume during a tape image copy.

User Action: Make sure the output device isn't write-protected. Retry the operation.

or

Explanation: An uncorrectable error occurred while formatting a disk. This error usually indicates a media or hardware error.

User Action: Reformat the disk.

HOME BLOCK READ ERROR

Explanation: An attempt has been made to perform a copy in Files-11 mode, but the input volume is not in Files-11 format or is not ready.

User Action: Wait until the device is ready or retype the command using the /IM or /TP switch to specify the proper transfer mode.

ILLEGAL BLOCK COUNT

Explanation: The number of blocks specified with the switch is 0 or is too large for the available buffer space.

User Action: Retry with valid block count. (See Table 12-3 for default block sizes).

INPUT DEVICE ERROR m, nnnnnn

Explanation: An error occurred while attempting to read data. The octal logical block number listed is the location where the error occurred. The LBN is expressed as a 19-bit binary representation (m,nnnnnn). See the verification error message in Section 12.3.3.3.

User Action: None.

INPUT VOLUME nn. OUT OF SEQUENCE

Explanation: The number of the input volume does not match the number of the required volume.

User Action: Mount the proper volume and proceed.

INSUFFICIENT BUFFER SPACE

Explanation: A record on an input tape is too large for the available buffer space. More blocks than are available in the buffer are needed to complete the copy operation.

User Action: Abort; or retry the operation specifying different block sizes.

LABEL OR BOOTSTRAP WRITE ERROR

Explanation: An error occurred while writing the initial part of a logical tape.

User Action: Retry or abort the operation.

LUN ASSIGNMENT ERROR

Explanation: Device or logical unit (LUN) combination specified in the command string is not supported by the current PRESRV system or is not a valid device for PRESRV operations.

User Action: Determine the status of the device and type the proper command. You may be able to redirect the PRESRV operation to another unit by means of the /UNIT switch (Section 12.3.5.2).

NO SUCH FILE

Explanation: File named in the command string was not found on a logical tape, or a magnetic tape was at End-Of-Volume (EOF) for input.

User Action: Retype the command using a proper file label.

OUTPUT DEVICE ERROR m, nnnnnn

Explanation: An error occurred while reading or writing data. The number printed is the octal logical block number where the error was detected. The LBN is expressed as a 19-bit binary representation (m,nnnnnn). See the verification error message in Section 12.3.3.3.

User Action: Retry, continue or abort the operation.

OUTPUT VOLUME OVERFLOW

Explanation: Input has more blocks that the output volume (Files-11) can hold.

User Action: Try an alternate transfer format. Check the blocking factor.

*** nnn: -- SELECT ERROR

Explanation: Device nnn: is off-line.

User Action: Put device on line.

STORAGE BITMAP FILE READ ERROR

Explanation: An attempt has been made to perform a copy in Files-11 mode, but the input volume is not in Files-11 format or is not ready.

User Action: Wait until the device is ready or retype the command specifying the proper transfer mode (/IM or /TP).

SYNTAX ERROR

Explanation: The command string was not a valid command.

User Action: Retype the command string correctly.

TAPE LABEL READ ERROR

Explanation: An error occurred while reading a logical tape label record.

User Action: Retry or abort the operation.

TAPE WAS WRITTEN FOR dd

Explanation: This is a warning message, not an error message. The Files-11 volume being restored from a logical tape is not of the same type as the one from which the tape was originally made. The new volume is given the bitmap of the old volume. The characters dd identify the intended Files-11 device.

User Action: None.

TRANSFER SPECIFICATION MODE ERROR

Explanation: An inconsistency appears in the command string. For example, a magnetic tape has a /FI switch associated with it.

User Action: Retype an acceptable command.

VERIFICATION ERROR nnnnnnn

Explanation: The data read back from the output volume does not match what was written. The number printed is the octal logical block number on the output device where the error was detected.

User Action: Retry, continue or abort the operation. If retry is specified, PRESRV performs the entire operation from the beginning.

VOLUME FAULTY - "R" TO RETRY, "C" TO CONTINUE

Explanation: An I/O error was detected during a copy operation; or a verification error occurred.

User Action: Type R to retry the operation from the beginning. Type C to continue from the point at which the error occurred.

CHAPTER 13

FILE STRUCTURE VERIFICATION UTILITY (VFY)

13.1 INTRODUCTION TO VFY

The File Structure Verification Utility (VFY) program provides the ability to:

- Check the readability and validity of a file-structured volume (default function).
- Print out the number of available blocks on a file-structured volume (/FR).
- Search for files that exist in the index file but that do not exist in any directory, that is, files which are "lost" in the sense that they cannot be accessed by file name (/LO). (See the IAS/RSX-11 I/O Operations Reference Manual for a description of the index file.)
- List all files in the index file, showing the file ID, filename, and owner (/LI).
- Mark as "used" all the blocks that appear to be available but are actually allocated to a file (/UP).
- Rebuild the storage allocation bit map so that it properly reflects the information in the index file (/RE).
- Restore files that are marked for delete (/DE).
- Perform a read check on every allocated block on a file-structured volume (/RC).

There should be no other activity on the volume while VFY is executing. In particular, activities which create new files, extend existing files, or delete files should not be attempted while VFY is executing a function.

VFY must not be aborted while a /UP, /RE, or /DE switch is being processed. Aborting VFY while it is in the process of modifying the storage allocation or index files may seriously endanger the integrity of that volume.

13.2 VFY COMMAND STRING

All VFY functions are invoked by entering a VFY command string through the initiating terminal. The VFY command string is formatted as follows:

```
listfile,scratchdev=indev/switch

          or

indev/switch     (This is a short form of TI:,indev=indev/switch)
```

listfile

Specifies the output listing file in the following format:

```
dev:[ufd]filename.filetype;ver
```

scratchdev

Specifies the device on which the scratch file produced by VFY is to be written. This parameter is in the following format:

```
dev:
```

The scratch file is used by VFY during the verification scan and during the lost file scan. It is created but not entered in a directory. Therefore, it is transparent to the user. The scratch file is automatically deleted upon termination of the VFY program.

If the system disk is faulty or full, use this parameter to force the scratch file to another device. The scratch file should always be assigned to another volume. The scratch file is not used for the /FR and /LI switches.

indev

Specifies the volume to be verified. This parameter is in the following format:

```
dev:
```

/switch

Specifies the function to be performed. This parameter is in the following format:

```
/sw
```

The VFY switches are described in detail in Section 13.3.

VFY normally operates in a read-only mode, assuming that the scratch file, if required, is on another device. VFY requires write access under the following conditions:

1. If the /UP or /RE switch is used, VFY requires write access to the storage allocation map ([0,0]BITMAP.SYS).

2. If the /DE switch is specified, VFY requires write access to the index file ([0,0]INDEXF.SYS).

3. If the /LO switch is specified and lost files are found, VFY requires write access to the [1,3] User File Directory.

VFY may be run under any UIC if only read access is required. If write access is required, VFY must run under a system UIC.

If write access to the volume index or bit map files is required for the desired operation, the user must mount the volume using the /UNLOCK switch.

Default file specifiers for VFY command line elements are listed in Table 13-1.

Table 13-1
VFY Default File Specifiers

Element	Default Value
dev:	Output listing device TI: Scratch file device SY0: Volume to be verified SY0:
[ufd]	The UIC under which VFY is currently running.
filename	No default - must be specified.
.filetype	No default - must be specified.
;ver	Latest version plus 1.

13.3 VFY SWITCHES

VFY functions are specified in the form of switches appended to the VFY command string. Command switches and functions are summarized in Table 13-2.

Table 13-2
VFY Functions and Switches

Function	Switch	Purpose
Delete	/DE	Reset marked-for-delete indicators.
Free	/FR	Print out the available space on a volume.
List	/LI	List entire index file by file identification.
Lost	/LO	Scan entire file structure looking for files which are not in any directory.
Read Check	/RC	Check entire volume to see if every block of every file can be read.
Rebuild	/RE	Recover blocks which appear to be allocated but are not contained in a file.
Update	/UP	Allocate blocks which appear to be available but have been allocated to a file.
Validity Check	Default	Check readability and validity of the volume mounted on specified device.

13.3.1 **Validity Check**

Validity Check (no switch) checks the readability and validity of the volume mounted on the specified device. This function entails reading all the file headers in the index file and ensuring that all the disk blocks referenced in the map area of each file header are marked as allocated to that file in the volume bit map.

Requirements for running the validity check:

1. The volume to be checked must be mounted as a Files-11 structured volume.
2. The volume may be write-protected if:
   a. It is not the system volume; or
   b. The required scratch file is directed to another file-structured volume.

A validity check is specified in the following form:

```
listfile,scratch dev=indev <CR>

     or

indev <CR>
```

Example

```
>VFY DR0:

CONSISTENCY CHECK OF INDEX AND BITMAP ON DR0:


INDEX  INDICATES 114524. BLOCKS FREE, 17156. BLOCKS USED OUT OF 131680.
BITMAP INDICATES 114524. BLOCKS FREE, 17156. BLOCKS USED OUT OF 131680.
```

13.3.1.1 **File Error Reporting** - As VFY verifies a volume, error conditions are reported. All errors for a given file are preceded by a file identification line that identifies the file in error. This line is formatted as follows:

```
FILE ID nn,mm filename.filetype;version OWNER [g,m]
```

nn,mm

Represents the unique file identification number assigned to the file by the system at file-creation time.

filename

Represents the file name.

.filetype

Represents the file type (for example, .OBJ for object file).

;version

Represents the version number of the file.

[g,m]

Represents the UIC that owns the file.

This file identification line is followed by one or more of the following messages:

I/O ERROR READING FILE HEADER-ERROR CODE -32

Explanation: VFY failed to read the file header for the specified file ID. The device is not mounted or is off-line, or the hardware has failed.

BAD FILE HEADER

Explanation: Software checks on the validity of the file header indicate that the header has been corrupted.

MULTIPLE ALLOCATION n,m

Explanation: The specified (double precision) logical block number is allocated to more than one file. If this error occurs, a second pass is automatically taken which will indicate all files that share each multiply-allocated block. The second pass is taken after all file headers have been checked (see Section 13.3.1.3).

BLOCK IS MARKED FREE n,m

Explanation: The specified logical block number is allocated to the indicated file but is not marked as allocated in the storage allocation map (see Section 13.3.1.4).

BAD BLOCK NUMBER n,m

Explanation: The specified block number was found in the header for this file but is illegal for the device (out of range). This indicates a corrupted file header.

FILE IS MARKED FOR DELETE

Explanation: A system failure occurred while the specified file was being deleted. The deletion was not completed and the file header still exists (see Section 13.3.1.2).

HEADER MAP OUT OF SYNC

Explanation: VFY detected an error in the header map area which also indicates a corrupted file header.

The last error message for the file is followed by a summary line for that file, as follows:

SUMMARY: MULT=nn, FREE=nn, BAD=nn.

MULT

Specifies the number of multiple block allocations.

FREE

Specifies the number of blocks marked free that should have been allocated.

BAD

Specifies the number of bad retrieval pointers in the file header.

If the output for VFY is directed to a terminal device, and you do not wish to see all the error messages for a given file, entering <CTRL/O> terminates the listing of all further error messages for that file, that is, all messages but the summary line.

13.3.1.2 **Files Marked-for-Delete** - If a file has been marked for delete but the deletion process was not completed, you can either restore the file, if you still need it, or you can delete the file to recover the space it was occupying. This situation only occurs when the system crashes during file processing.

- Restoring a File

  To restore a file marked-for-delete, the disk volume must be mounted using the MCR Mount command with the /UNL switch specified. For example:

```
>MOU DK0:/UNL
```

Then, run VFY specifying the /DE switch to reset the marked-for-delete indicators in file headers. Once the delete indicator has been reset, run VFY specifying the /LO switch to scan the entire file structure.

The deletion process may have proceeded partially and a portion at the end of the file may be missing. This condition can be detected by a directory listing obtained using the PIP /FU switch.

- Deleting a File

  Files that are marked-for-delete can be deleted directly with PIP, once their unique File ID has been obtained via a validity check. The File ID appears as the first entry in the file identification line which precedes each list of file errors (see Section 13.3.1.1). The following example illustrates how the File ID is used with PIP to delete a file:

```
>PIP /FI:12:20/DE
```

  In this example, the file with File ID 12,20 is deleted from the system device. PIP issues the error message:

```
PIP -- FAILED TO MARK FILE FOR DELETE-NO SUCH FILE
```

  since the file system denies the existence of files already marked-for-delete; however, the file is completely deleted.

Once files have been restored or deleted, run VFY with the /RE switch specified to assure the consistency of the volume's storage allocation bit map.

13.3.1.3 **Deletion of Multiply-Allocated Blocks** - If the file structure contains multiply-allocated blocks, it is necessary to delete files until there are no more such blocks. An automatic rescan of the volume identifies which files share which blocks. This rescan lists the first as well as subsequent files containing the multiply-allocated blocks. Using this information, you can then determine which, if any, of the files can be saved and delete the rest, using the PIP delete function.

Be careful when deleting files containing multiply-allocated blocks. After the files have been deleted, VFY should be run once again to ensure that all of the files containing multiply-allocated blocks have been deleted.

13.3.1.4 **Elimination of Free Blocks** - Once there are no multiply-allocated blocks, the next concern is the elimination of blocks that are marked Free in the storage allocation bit map but are actually allocated to a file. To cause these blocks to be reallocated in the storage allocation bit map, rerun the validity check specifying the /UP switch. This allocates all blocks that should have been marked as allocated. See Section 13.3.8 for a description of the /UP switch.

Once there are no multiply-allocated blocks and no blocks marked free that are actually in use, the file structure is safe for writing new files and extending existing files. However, if there were such errors, there may be files which have had data blocks overwritten as the result of multiple allocation.

13.3.1.5 **Recovering Lost Blocks** - To determine whether any blocks have been lost on a file-structured volume, examine the last two lines of output from the validity check. The last two lines of output give the free space on the volume. The first line of the two tells how much room is available according to the index file (that is, the number of blocks that are not in use by any file in the index file). The last line specifies how much room is available according to the storage allocation bit map. Assuming there are no other errors, these two figures should agree. If the index file indicates that more blocks are free than the storage allocation bit map, then those blocks are "lost" in the sense that they appear to be allocated, but no file contains them. Lost blocks can be recovered by rerunning the validity check specifying the /RE switch. See Section 13.3.7 for a description of the /RE switch.

13.3.2 Delete Switch (/DE)

The Delete switch (/DE) resets the marked-for-delete indicators in the file header area of files that were marked for deletion but never actually deleted. The Delete switch is specified in the following form:

```
listfile,scratchdev=indev/DE

     or

indev/DE
```

The volume must be mounted with the /UNL switch.

VFY must be running under a system UIC.

13.3.3 Free Switch (/FR)

The Free switch (/FR) displays the available space on a specified volume. The Free switch is specified in the following form:

```
listfile=indev/FR

     or

indev/FR
```

Free switch output is the following message:

```
dev: HAS nnnn.  BLOCKS FREE, nnnn.  BLOCKS USED OUT OF nnnn.
```

13.3.4 List Switch (/LI)

The List switch (/LI) lists the entire index file by file identification. The output for each file specifies the file number, file sequence number, file name, and owner. A typical index file listing is illustrated in Figure 13-1. The List switch is specified in the following form:

```
listfile,scratchdev=indev/LI

     or

indev/LI
```

```
VFY>DK:/LI
LISTING OF INDEX ON DK0:

FILE ID 000001,000001 INDEXF.SYS;1    OWNER [1,1]
FILE ID 000002,000002 BITMAP.SYS;1    OWNER [1,1]
FILE ID 000003,000003 BADBLK.SYS;1    OWNER [1,1]
FILE ID 000004,000004 000000.DIR;1    OWNER [1,1]
FILE ID 000005,000005 CORIMG.SYS;1    OWNER [1,1]
FILE ID 000006,000006 001001.DIR;1    OWNER [1,1]
FILE ID 000007,000007 001002.DIR;1    OWNER [1,2]
FILE ID 000010,000010 EXEMC.MLB;1     OWNER [1,1]
FILE ID 000011,000011 RSXMAC.SML;1    OWNER [1,1]
FILE ID 000012,000012 NODES.TBL;1     OWNER [1,1]
FILE ID 000013,000036 QIOSYM.MSG;311 OWNER [1,2]
FILE ID 000014,000037 F4PCOM.MSG;1    OWNER [1,2]
```

Figure 13-1 VFY Listing Sample Using the /LI Switch

13.3.5 Lost Switch (/LO)

The Lost switch (/LO) scans the entire file structure looking for files which are not in any directory and, thus, are lost in that they cannot be referenced by file name. A list of the files is produced, and if the "lost file directory" [1,3] exists on that volume, all the files will be entered in that directory. The Lost switch is specified in the following form:

```
listfile,scratchdev=indev/LO

     or

indev/LO
```

13.3.6 Read Check Switch (/RC)

The Read Check switch (/RC) checks to ensure that every block of every file on a specified volume can be read. The Read Check switch is specified in the following form:

```
listfile=indev/RC[:n]

     or

indev/RC[:n]
```

The optional parameter [:n] is the blocking factor which indicates the number of file blocks to be read at a time. The default value is the maximum number of blocks available in VFY's buffer area.

Since the Read Check function is a read-only operation, the volume can be write-protected.

The buffer area available may be increased by installing VFY in a larger partition. Five blocks are available when VFY is installed in an 8K partition, and four blocks are added for each 1K increment.

For the fastest possible read check, the maximum block factor should be used. Whenever an error is encountered, each block of the portion in error is reread individually to determine which data block(s) cannot be read.

When an error is detected, a file identification line is listed in the following format:

```
FILE ID nn,nn filename.typ;ver.  blocks used/blocks allocated
```

Following this line, an error message is listed. If a blocking factor other than 1 is in use, an error message in the following form will be issued:

```
ERROR STARTING AT VBN n1,n2 LBN n1,n2 - ERROR CODE -n
```

Following the first error message, there should be one or more error messages indicating the exact block(s) in error. The second error message line(s) will be in the following form:

```
ERROR AT VBN n1,n2 LBN n1,n2 - ERROR CODE -n
```

If an ERROR STARTING AT line is displayed without one or more ERROR AT lines, a multiblock read operation on the selected device has failed, but the data blocks appear to be individually readable.

If the VBN of the unreadable block listed in the ERROR AT line is beyond the block-used-count, the data portion of the file is readable.

The negative number printed after the ERROR CODE message is usually -4 to indicate a device parity error. Other error codes are contained in Appendix I of the IAS/RSX-11 I/O Operations Reference Manual.

13.3.7 Rebuild Switch (/RE)

The Rebuild switch (/RE) recovers lost blocks, that is, blocks that appear to be allocated but which are not contained in any file. The Rebuild switch is specified in the form:

```
listfile,scratchdev=indev/RE

    or

indev/RE
```

Multiply-allocated blocks must be removed (deleted) from the file structure before the rebuild can be run.

The volume being updated must be write-enabled.

VFY must be running under a system UIC.

The scratch file should be on another volume. If this is impossible, the volume must be dismounted immediately after VFY terminates. (Failure to do this may result in partial updating of the storage allocation bit map.) Then the volume should be mounted again, and the scratch file must be deleted manually. VFY issues a detailed message in this case, specifying the name of the scratch file to be deleted.

13.3.8 Update Switch (/UP)

The Update switch (/UP) allocates all blocks that appear to be available but are actually allocated to a file. The Update switch is specified in the following form:

```
listfile,scratchdev=indev/UP

    or

indev/UP
```

Files with multiply-allocated blocks must be deleted from the file structure before the update can be run.

The volume being updated must be write-enabled.

VFY must be running under a system UIC.

The scratch file should be on another volume. If this is impossible, the volume must be dismounted immediately after VFY terminates. (Failure to do this may result in partial updating of the storage allocation bit map.) Then the volume should be mounted again, and the scratch file must be deleted manually. VFY issues a detailed message in this case specifying the name of the scratch file to be deleted.

The message is:

```
VFY -- TO COMPLETE THE STORAGE MAP UPDATE DISMOUNT THE VOLUME
       IMMEDIATELY.  THEN MOUNT IT AND DELETE THE FOLLOWING
       FILE: [ufd] filespec
```

13.4 VFY ERROR MESSAGES

VFY -- COMMAND SYNTAX ERROR

Explanation: The command entered does not conform to command syntax rules.

User Action: Reenter the command line with the correct syntax specified.

VFY -- FAILED TO ALLOCATE SPACE FOR TEMP FILE

Explanation: The volume specified for the temporary scratch file is full.

User Action: Use PIP to delete all unnecessary files and rerun VFY.

VFY -- FAILED TO ATTACH DEVICE

or

VFY -- FAILED TO DETACH DEVICE

or

VFY -- ILLEGAL DEVICE

Explanation: The file specifier entered contains an illegal device.

User Action: Reenter the command line with the correct device specified.

VFY -- FAILED TO CLOSE DIRECTORY FILE (See I/O ERROR messages).

VFY -- FAILED TO ENTER FILE

Explanation: One of the following conditions may exist:

- VFY is not running under a system UIC.
- The device is not on-line.
- The device is not mounted.
- The hardware has failed.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

VFY -- FAILED TO OPEN DIRECTORY FILE (See OPEN FAILURE error messages)

VFY -- ILLEGAL SWITCH

Explanation: The switch specified is not a valid VFY switch or a valid switch is used illegally.

User Action: Reenter the command line with the correct switch specified.

VFY -- I/O ERROR ON INPUT FILE

or

VFY -- I/O ERROR ON OUTPUT FILE

or

VFY -- I/O ERROR READING DIRECTORY FILE

or

VFY -- FAILED TO CLOSE DIRECTORY FILE

Explanation: One of the following conditions may exist:

- The device is not on-line.
- The device is not mounted.
- The hardware has failed.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

VFY -- NO DYNAMIC MEMORY AVAILABLE - PARTITION TOO SMALL

Explanation: VFY does not have enough buffer space to run.

User Action: Run VFY in a larger partition (8K minimum).

VFY -- OPEN FAILURE ON BIT MAP

or

VFY -- OPEN FAILURE ON INDEX FILE

or

VFY -- OPEN FAILURE ON LISTING FILE

or

VFY -- OPEN FAILURE ON TEMPORARY FILE

or

VFY -- FAILED TO OPEN DIRECTORY FILE

Explanation: One of the following conditions may exist:

- VFY is not running under a system UIC, but should be.
- The named file does not exist in specified directory.
- The volume is not mounted.
- The specified file directory does not exist.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

VFY -- THEY ARE STILL LOST, COULD NOT FIND DIRECTORY

Explanation: UFD [1,3] did not exist on the volume.

User Action: Use the MCR UFD command to enter UFD [1,3] on the volume.

13.5 VFY ERROR CODES (IAS SYSTEM ONLY)

If VFY cannot access the message file, errors are reported in the following format:

VFY -- ERROR CODE nn.

where:

nn. is one of the error codes contained in Table 13-3.

Refer to Section 13.4 for error descriptions and suggested user actions.

Table 13-3
VFY Error Codes

Error Codes	Corresponding VFY Error Message
1	ILLEGAL DEVICE
2	OPEN FAILURE ON BIT MAP
3	OPEN FAILURE ON TEMPORARY FILE
4	FAILED TO ALLOCATE SPACE FOR TEMP FILE
5	FAILED TO DETACH DEVICE
6	FAILED TO ATTACH DEVICE
7	COMMAND SYNTAX ERROR
8	I/O ERROR ON INPUT FILE
9	I/O ERROR ON OUTPUT FILE
10	ILLEGAL SWITCH
11	OPEN FAILURE ON LISTING FILE
12	OPEN FAILURE ON INDEX FILE
13	NO DYNAMIC MEMORY AVAILABLE - PARTITION TOO SMALL

PART 6
PROGRAMMING UTILITIES

CHAPTER 14

LIBRARIAN UTILITY PROGRAM (LBR)

14.1 INTRODUCTION TO LBR

The Librarian Utility Program (LBR) allows you to create, update, modify, list, and maintain object, macro, and universal library files. A library file is a direct access file which usually contains modules of the same type. Library files are organized for rapid access by the Task Builder, MACRO-11 Assembler, and the Universal Library Access ($ULA) system library routine (see the IAS/RSX-11 System Library Routines Manual, Appendix B.)

LBR and library files, in conjunction with the MACRO-11 Assembler, the Task Builder, and the $ULA system routine, provide fast entry-point search time, easy update with minimal copying of entire files, and the ability to handle multiple module types.

Library files contain two directory tables: an entry point table (EPT) that contains entry point names (global symbols), and a module name table (MNT) that contains module names. Both the EPT and MNT are alphabetically ordered. The EPT applies only to object libraries.

Object module names are derived from .TITLE directives, while entry point names are derived from defined global symbols. Once an entry point is located, its associated module can be accessed directly.

Macro module names are derived from .MACRO directives; macro entry point names are not applicable.

Universal module names are derived from file names at insert time; universal entry point names are not applicable. A module of a universal library can be accessed by the $ULA Routine, found in the IAS/RSX-11 System Library Routines Reference Manual. You may use a universal library to contain modules inserted from any kind of file.

You can invoke LBR using any of the methods for invoking a utility described in Chapter 1.

14.2 LBR COMMAND STRING

LBR accepts command strings in the following general format:

```
outfile[,listfile]=infile1[,infile2,...infilen]
```

LBR allows only one level of indirect command file nesting for RSX-11M and RSX-11M-PLUS. For a complete description of file specifiers, see Chapter 1.

14.3 DEFAULTS IN LBR FILE SPECIFIERS

Defaults in LBR file specifiers are described in Table 14-1.

Table 14-1
Defaults in LBR File Specifiers

Specifier	Default
dev:	Output File SY0: Listing File The device which was specified for the output file; otherwise, the default for the output file. Input File For the first input file specifier, SY0:. For the subsequent input file specifiers, the device specified in the previous input file specifier; otherwise, the default for the previous input file specifier.
[ufd]	Output File The UIC under which LBR is currently running. Listing File The UFD which was specified for the output file; otherwise, the default for the output file specifier. Input File For the first input file specifier, the UIC under which LBR is currently running. For the subsequent input file specifiers, the UFD specified in the previous input file specifier; otherwise, the default for the previous input file specifier.
filename	No default. Must be specified.
.type	Output File Depends on the default in effect (see Section 14.6.4), except when the /CO or /CR switch is specified (see Sections 14.6.1 or 14.6.2, respectively). Listing File .LST Input File Refer to the descriptions of /CO (Section 14.6.1), /IN (Section 14.6.8), and /RP (Section 14.6.11) switches.

(continued on next page)

Table 14-1 (Cont.)
Defaults in LBR File Specifiers

Specifier	Default
;ver	Latest version of the file, or latest version plus 1 for the output file when the /CO, /CR, OR /EX switches are specified.
/switch	Output File /IN (Insert) List File /SP/LI (spool and list module names) Input File None.

14.4 LBR SWITCHES

LBR uses switches appended to file specifiers to invoke functions. These switches are summarized in Table 14-2. For a detailed description of each LBR switch, see Section 14.6.

Table 14-2
LBR Switches

Option	Switch	Function
Compress	/CO	Compress a library file.
Create	/CR	Create a library file.
Delete	/DE	Delete a library module and all of its entry points.
Default	/DF	Specify the default library file type.
Delete Global	/DG	Delete a library module entry point.
Entry Point	/EP	Include entry point elements in the library entry point table.
	/-EP	Exclude entry point elements in the library entry point table.
Extract	/EX	Extract (read) one or more modules from a library file and write them into a specified output file.
Insert	/IN	Insert a module.

(continued on next page)

Table 14-2 (Cont.)
LBR Switches

Option	Switch	Function
List	/LI	List module names.
	/LE	List module names and module entry points.
	/FU	List module names and full module description.
Modify Header	/MH	Modify a universal module header.
Replace	/RP	Replace a module.
	/-RP	Do not replace a module.
Spool	/SP	Spool the listing for printing.
	/-SP	Do not spool the listing.
Selective Search	/SS	Set selective search attribute in module header.
Squeeze	/SZ	Reduce the size of macro source.
	/-SZ	Do not reduce size of a specific macro source.

14.5 FORMAT OF LIBRARY FILES

A library file consists of a library header, an entry point table, a module name table, the library modules and their headers, and (usually) free space. The entry point table has zero length for macro libraries and universal libraries. Figure 14-1 illustrates object and macro library file format and Figure 14-2 illustrates universal library file format.

14.5.1 Library Header

The header section is a full block in which the first 24. words are used to describe the current status of the library. Its contents are updated as the library is modified, so LBR can access the information it needs to perform its functions (Insert, Compress, and so on). The 24th word in the library header is the default insert file type for universal libraries and is undefined for macro and object libraries. See Figure 14-3.

14.5.2 Entry Point Table

The entry point table consists of 4-word elements containing an entry point name (words 0-1), and a pointer to the module header of the module where the entry point is defined (words 2-3). See Figure 14-4.

This table is searched when a library module is referenced by one of its entry points. The table is sequenced in order of ascending entry point names. The entry point table is not used for macro or universal library files.

14.5.3 Module Name Table

The module name table is searched when the library module is referenced by its module name, rather than by one of its entry points. It is made up of 4-word elements; a module name (words 0-1) and a pointer to the module header (words 2-3). See Figure 14-5. The module name table is sequenced in order of ascending module names.

14.5.4 Module Header

Each module starts with a header of 8 words for object and macro modules and 32. words for universal modules, identifying the type and status of the module, its length (number of words), and so forth; see Figure 14-6, and Figure 14-7.

In object modules, the low-order bit of the attributes byte is set if the module has the selective search attribute. Also, in object modules, the 2 words of type-dependent information contain the module identification defined by the .IDENT directive at assembly time. In macro modules, these 2 words are undefined.

For universal modules type dependent identification is derived from the file type and version number of the input file. See Figure 14-7.

Universal libraries allow module header changes (optional descriptive information) via the /MH switch.

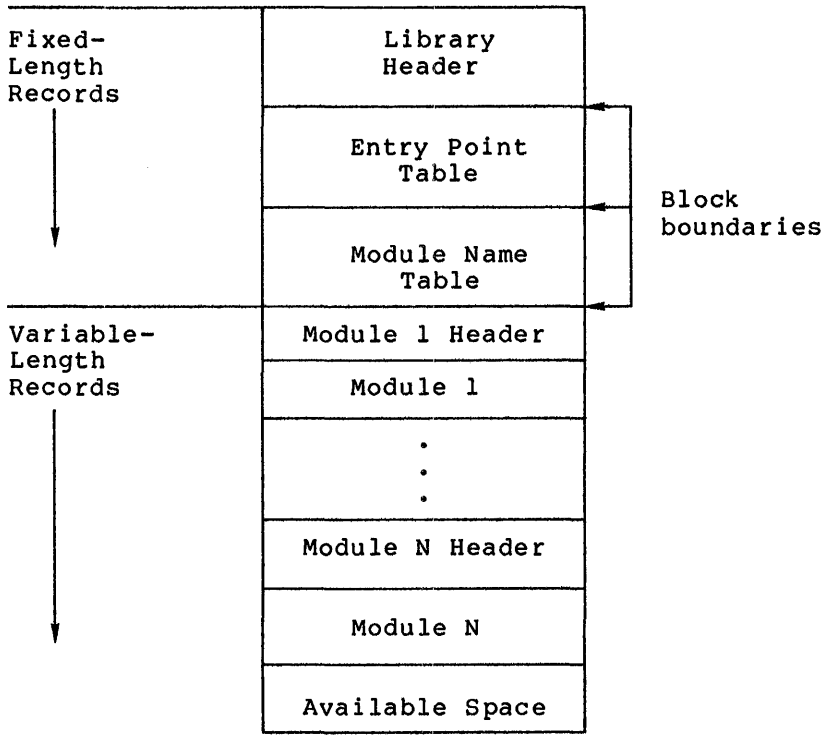
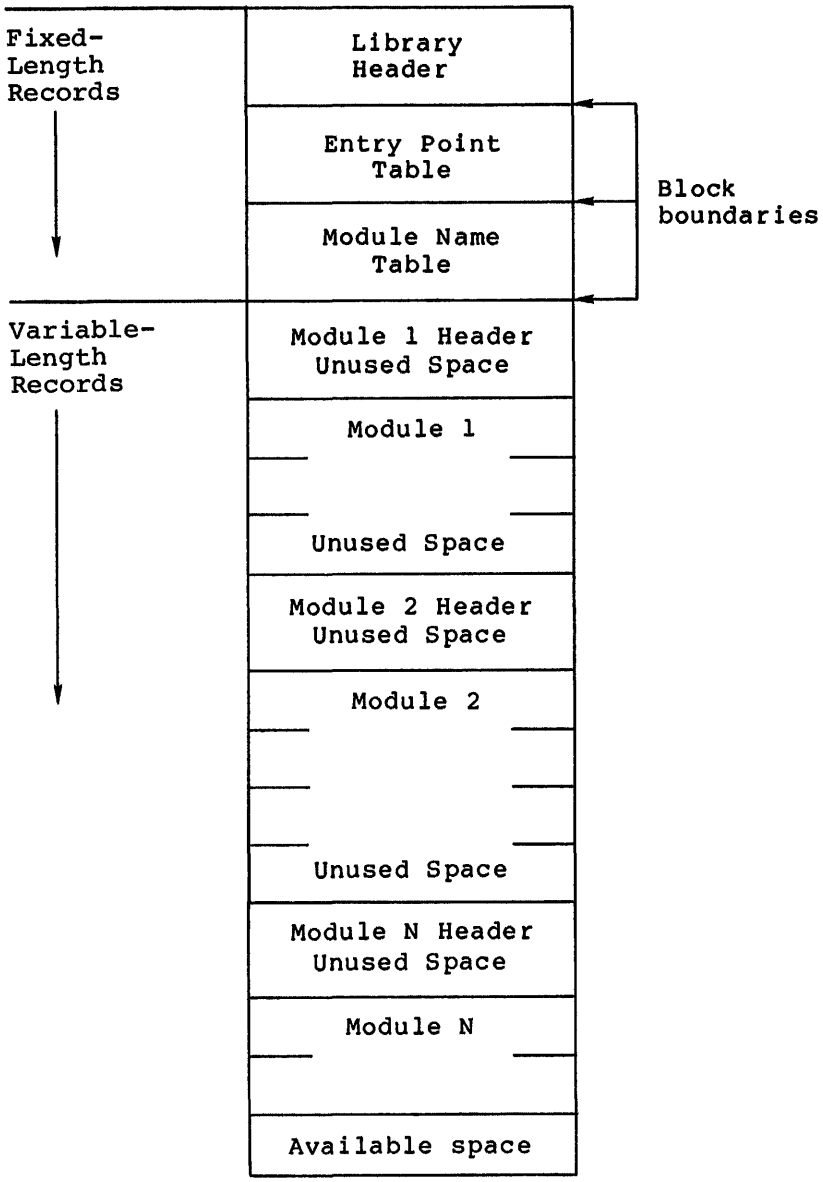


Figure 14-1 General Format for Object and Macro Library Files



NOTE

All universal module headers and the first record of each universal module will start on a block boundary.

Figure 14-2 Universal Library File Format

OFFSET		
WORD 0	NON ZERO ID	LIBRARY TYPE
2	LBR (LIBRARIAN) VERSION	
4	(.IDENT FORMAT)	
6		YEAR
10	DATE AND	MONTH
12	TIME LAST	DAY
14	INSERT	HOUR
16		MINUTE
20		SECOND
22	RESERVED	SIZE EPT ENTR's
24	EPT STARTING RELATIVE BLOCK	
26	NO. EPT ENTRIES ALLOCATED	
30	NO. EPT ENTRIES AVAILABLE	
32	RESERVED	SIZE MNT ENTR'S
34	MNT STARTING REL BLOCK	
36	NO. MNT ENTRIES ALLOCATED	
40	NO. MNT ENTRIES AVAILABLE	
42	LOGICALLY DELETED	
44	AVAILABLE (BYTES)	
46	CONTIGUOUS SPACE	
50	AVAILABLE (BYTES)	
52	NEXT INSERT RELATIVE BLOCK	
54	START BYTE WITHIN BLOCK	
56	UNIVERSAL DEFAULT INSERT TYPE	

(UNDEFINED FOR MACRO AND OBJECT LIBRARIES)

Figure 14-3 Contents of Library Header

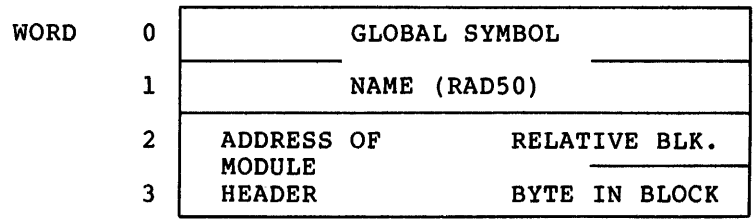


Figure 14-4 Format of Entry Point Table Element

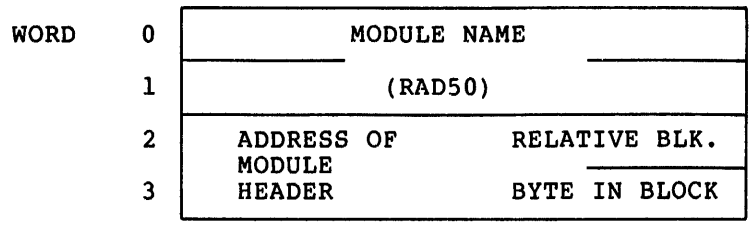


Figure 14-5 Format of Module Name Table Element

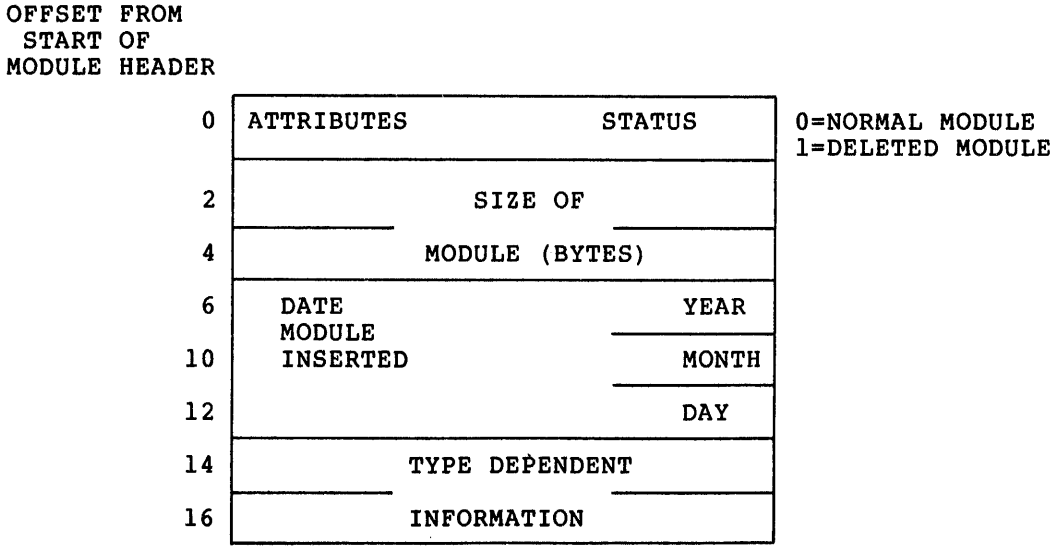


Figure 14-6 Module Header Format for Object and Macro Libraries

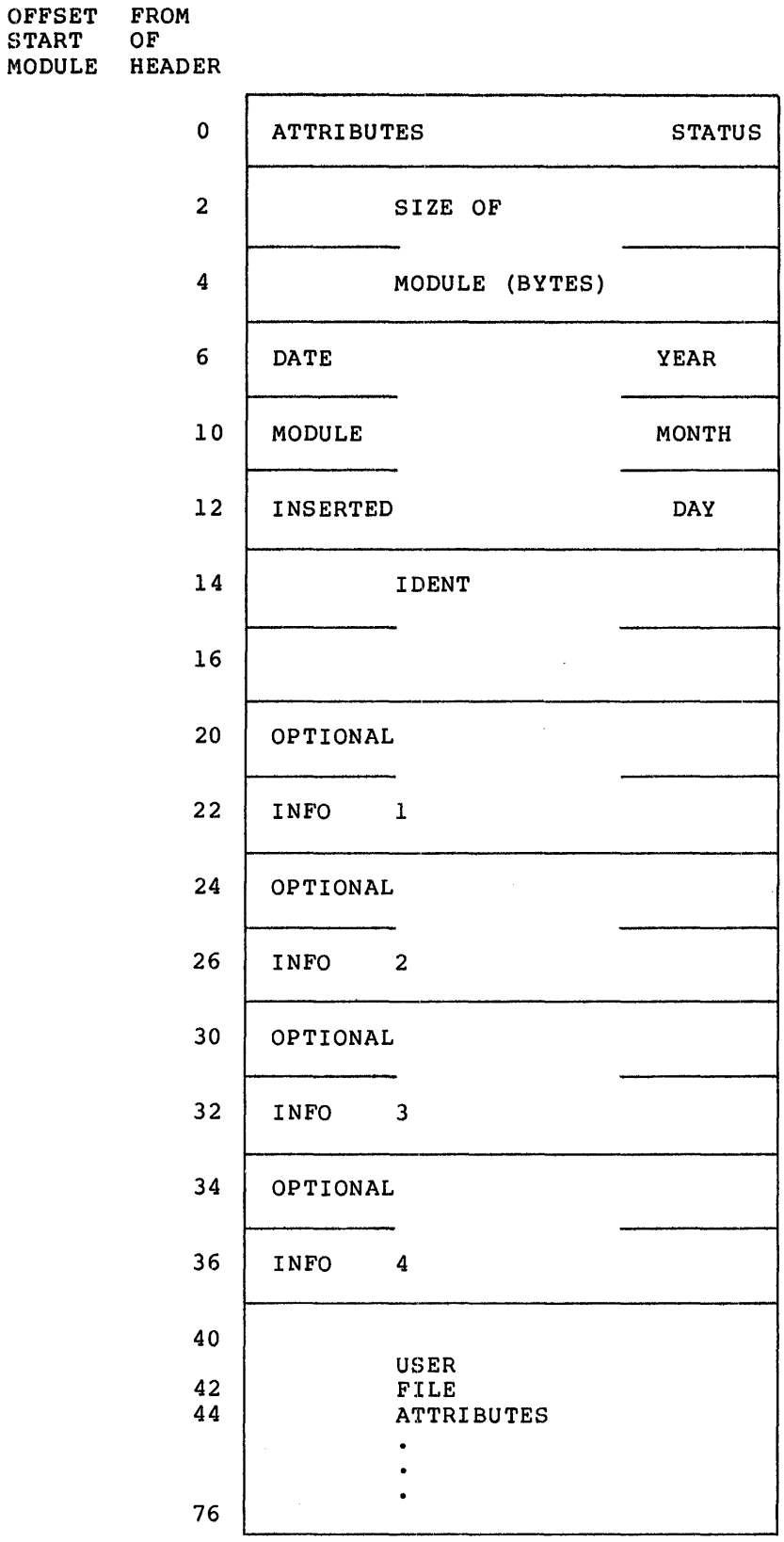


Figure 14-7 Module Header Format for Universal Libraries

14.6 LBR SWITCHES

This section provides a detailed description of each LBR switch.

14.6.1 Compress Switch (/CO)

Use the Compress switch to physically delete all logically deleted records, putting all free space at the end of the file, and making the free space available for new library module inserts. Additionally, the library table specification may be altered for the resulting library. LBR accomplishes this by creating a new file that is a compressed copy of the old library file. The old library file is not deleted after the new file is created.

The /CO switch can be appended only to the output file specifier. The format for specifying the Compress switch is:

```
outfile/CO:size:ept:mnt = infile
```

outfile

Specifies the file that is to become the compressed version of the input file. Default type is .OLB if input file is an object library, .MLB if input file is a macro library, or .ULB if the input file is a universal library.

/CO

Specifies the Compress switch.

:size

Specifies the size of the new library file in 256(10)-word blocks. The size of the old library file is the default size.

:ept

Specifies the number of entry point table (EPT) entries to allocate. If the value specified is not a multiple of 64(10), the next highest multiple of 64(10) is used. The number of EPTs in the old library file is the default value. This parameter is always forced to zero for macro libraries and universal libraries. Maximum number of entries is 4096(10).

:mnt

Specifies the number of module name table (MNT) entries to allocate. If the value specified is not a multiple of 64(10), the next highest multiple of 64(10) is used. The number of MNTs in the old library file is the default value. Maximum number of entries is 4096(10).

infile

Specifies the library file to be compressed. Default file type is .OLB for object libraries, .MLB for macro libraries, and .ULB for universal libraries. The actual default file type is determined by the current default library filetype (see Section 14.6.4).

Example

```
LBR>RICKLIB/CO:100.:128.:64.=SHEILA.OLB
```

In this example, file SHEILA.OLB is compressed, and a new file, RICKLIB.OLB, is created with the following attributes:

```
size = 100(10) blocks
ept = 128(10) entry points
mnt = 64(10) module names
```

The new file, RICKLIB.OLB, receives a version number that is 1 version greater than the latest version for the file.

Both files, RICKLIB.OLB and SHEILA.OLB, reside in the default directory file on SY0:.

14.6.2 Create Switch (/CR)

Use the Create switch to allocate a contiguous library file on a direct access device (for example, a disk). It initializes the library file header, the entry point table, and the module name table.

The /CR switch can be appended only to the output file specification. The format for specifying the Create switch is:

```
outfile/CR:size:ept:mnt:libtype:infiletype
```

outfile

Specifies the library file being created. The default file type is .OLB if an object library is being created, .MLB if a macro library is being created, or .ULB if a universal library is being created.

/CR

Specifies the Create switch.

:size

Specifies the size of the library file in disk (256-word) blocks. The default size is 100(10) blocks.

:ept

Specifies the number of entry point table (EPT) entries to allocate. The default value is 512(10) for object libraries. This parameter is always forced to 0 for macro libraries and universal libraries. Maximum number of entries is 4096(10).

:mnt

Specifies the number of module name table (MNT) entries to allocate. The default value is 256(10). Maximum number of entries is 4096(10).

:libtype

Specifies the type of library to be created. Acceptable values are OBJ for object libraries, MAC for macro libraries, and UNI for universal libraries. The default is the last value specified or implied with the /DF switch (see Section 14.6.4), or OBJ if /DF has not been specified.

:infiletype

Specifies the default input file type for the created universal library. If this value is not specified the default input file type for universal libraries is .UNI. This value is not defined for object or macro libraries.

If the values specified for ept and mnt are not multiples of 64(10), EPT and MNT are automatically filled out to the next disk block boundary.

Example

```
LBR>RICKLIB/CR::128.:64.:OBJ=SHEILA,LAURA,JENNY
```

In this example, a combination of functions is performed. First, the library file RICKLIB.OLB is created in the default directory on SY0:. RICKLIB has the following attributes:

```
size = 100(10) blocks (default size)
ept  = 128(10) entry points
mnt  = 64(10) module names
type = .OBJ
```

Second, object modules from the input files SHEILA.OBJ, LAURA.OBJ, and JENNY.OBJ, which reside in the default directory on SY0:, are inserted into the newly created library file. Insert is the default switch for input files (see Section 14.6.8).

14.6.3 Delete Switch (/DE)

Use the Delete switch to delete library modules and their associated entry points (global symbols) from a library file. Up to 15 library modules and their associated entry points can be deleted with one delete command.

When LBR begins processing the /DE switch, it prints the following message on the initiating terminal:

```
MODULES DELETED:
```

As modules are logically deleted from the library file, the module name is printed on the initiating terminal. See the example at the end of this section.

If a specified library module is not contained in the library file, a message is printed on the initiating terminal, and the processing of the current command is terminated. This message is as follows:

```
LBR -- *FATAL*-NO MODULE NAMED "name"
```

The /DE switch can be appended only to the library file specifier.

When LBR deletes a module from a library file, the module is not physically removed from the file, but is marked for deletion. This means that, although the module is no longer accessible, the file space that the module once occupied is not available for use (unless the deleted module is the last module which was inserted). To physically remove the module from the file and make the freed space available for use, you must compress the library (see Section 14.6.1).

The form for specifying the Delete switch is:

```
outfile/DE:module1[:module2...:modulen]
```

outfile

Specifies the library file.

/DE

Specifies the Delete switch.

:module

Specifies the name of the module to be deleted.

Example

```
LBR> RICKLIB/DE:SHEILA:LAURA:JENNY

MODULES DELETED:

SHEILA

LAURA

JENNY
```

In this example, the modules SHEILA, LAURA, and JENNY and their associated entry points are deleted from the latest version of library file SY0:RICKLIB.OLB.

14.6.4 Default Switch (/DF)

Use the Default switch to specify the default library file type. Acceptable values are OBJ for object libraries, MAC for macro libraries, and UNI for universal libraries. When a default library file type is not specified by the /DF switch, .OBJ is the default library file type.

Specifying a default value:

1. Sets the default file type argument for the Create switch (/CR).

2. Provides a file type default value of .MLB for macro libraries, .ULB for universal libraries, and .OLB for object libraries when opening an output (library) file, except in the cases of /CO and /CR. When /CO is specified, the default applies to the library input file. When /CR is specified, the default file type is .OLB if an object library is being created, .ULB if a universal library is being created, or .MLB if a macro library is being created. The /DF switch only affects the name of the file to be opened; thereafter, the library header record information is used to determine the type of library file being processed.

The /DF switch can be issued alone or appended to a library file specifier. The form for specifying the Default switch is:

```
outfile/DF:filetype...

or

/DF:filetype
```

outfile

Specifies the library file.

/DF

Specifies the Default switch.

filetype

Specifies the default library file type: OBJ for object library files, MAC for macro library files, and UNI for universal library files.

If a value other than OBJ, ULB, or MAC is specified, the current default library type will be set to object libraries, and the following message will be displayed:

```
LBR -- *FATAL*-INVALID LIBRARY TYPE SPECIFIED
```

Examples

```
LBR>/DF:MAC
LBR>RICKLIB=infile
```

File RICKLIB.MLB is opened for insertion.

```
LBR>/DF:MAC
LBR>RICKLIB/DF:OBJ=infile
```

File RICKLIB.OLB is opened for insertion.

```
LBR>/DF:MAC
LBR>RICKLIB/CR
```

Macro library RICKLIB.MLB is created.

```
LBR>/DF:MAC
LBR>RICKLIB/CR::::OBJ
```

Object library RICKLIB.OLB is created.

```
LBR>/DF
LBR>TEMP/CO=RICKLIB
```

RICKLIB.OLB is opened for compression. If RICKLIB.OLB is an object library, the file TEMP.OLB is created to receive the compressed output. If RICKLIB.OLB is a macro library (a nonstandard use of the type .OLB), the file TEMP.MLB is created.

```
LBR>/DF:OBJ
LBR>TEMP/CO=RICKLIB.MLB
```

Assuming that file RICKLIB.MLB is a macro library, the macro library file TEMP.MLB is created to receive the compressed output.

```
LBR>/DF:UNI
LBR>RICHLIB=TEST
```

File RICHLIB.ULB is opened for insertion.

14.6.5 Delete Global Switch (/DG)

Use the Delete Global switch to delete a specified entry point (global symbol) from the EPT. Up to 15 entry points may be deleted with one command. This command does not affect the object module which contains the actual symbol definition. You may wish to delete an entry point if a module to be inserted has the same entry point.

When LBR begins processing the /DG switch, it prints the following message on the initiating terminal:

```
ENTRY POINTS DELETED:
```

As entry points are deleted from the library file, the entry point is printed on the initiating terminal. See the example at the end of this section.

If a specified entry point is not contained in the EPT, a message is printed on the initiating terminal, and the processing of the current command is terminated. This message is as follows:

```
LBR -- *FATAL* - NO ENTRY POINT NAMED "name"
```

The /DG switch can only be appended to the library file specifier.

The format for specifying the Delete Global switch is:

```
outfile/DG:global1[:global2...:globaln]
```

outfile

Specifies the library file.

/DG

Specifies the Delete Global switch.

global

Specifies the name of the entry point to be deleted.

Example

```
LBR>RICKLIB/DG:SHEILA:LAURA:JENNY

ENTRY POINTS DELETED:

SHEILA

LAURA

JENNY
```

In this example, the entry points SHEILA, LAURA, and JENNY are deleted from the latest version of the library file named SY0:RICKLIB.OLB.

14.6.6 Entry Point Switch (/EP)

Use the Entry Point switch to control (include or exclude) the placement of global symbols in a library entry point table. The Entry Point switch can be specified in either a positive or negative form:

/EP	Include entry points in the entry point table.
/-EP	Do not include entry points in the entry point table.
/NOEP	Do not include entry points in the entry point table.

The positive form (/EP) causes all entry points in a module or modules to be entered in the library entry point table.

Either negative form (/-EP or /NOEP) provides for a module to be included in a library while excluding the entry points in that module from being entered in the library entry point table.

/EP is the LBR default; if the switch is not specified, all entry points are entered into the library entry point table.

The Entry Point switch has no effect on macro or universal libraries.

The format for specifying the Entry Point switch is:

```
outfile[ /EP ]=infile,...infilen
       [/-EP ]
       [/NOEP]

                or

outfile=infile[ /EP ][,...infilen[/EP ]]
               [-/EP ]            [-/EP]
               [/NOEP]            [/NOEP]
```

outfile

Specifies the output file. When the entry point switch is applied to this file specification, LBR assumes each of the input files contains modules for which entry points are to be either included or excluded.

infile

Specifies an input file. When the Entry Point switch is applied to an input file specification, LBR assumes only the input files to which the switch is applied contain modules for which entry points are to be either included or excluded.

/-EP is useful for including modules which contain duplicate entry point names in the same library. /-EP provides the means for entering a module in the library without having its entry points included in the library entry point table.

/-EP is also useful in the case where the Task Builder uses only module names to search for modules in an object module library. In this case, entries in the library entry point table are not required. /-EP can be used to exclude entry points from being entered in the library entry point table.

Depending on whether the Entry Point switch is applied to the output specifier or to an input specifier, it has either a global or local effect.

When applied to the output file specifier, the Entry Point switch has a global effect. That is, LBR either includes all entry points in the entry point table or excludes all entry points from being entered in the entry point table.

When applied to an input file specifier, the Entry Point switch has a local effect. That is, LBR either includes entry points in the entry point table or excludes entries from being entered in the entry point table for only those modules to which the switch is applied.

The positive and negative forms of the switch may be applied to both the output and input file specifiers. For example, the effect of /EP applied to the output file can be overridden by applying /-EP to a specific input file.

Entry points in an object module are not affected by the Entry Point switch; the Entry Point switch permits you to either include or exclude entries in the library entry point table.

14.6.7 **Extract Switch (/EX)**

Use the Extract switch to extract (read) one or more modules from an object or macro library file and write them into a specified output file. If more than one module is extracted, the modules are concatenated in the output file. The extract operation has no effect on the library file from which the modules are extracted; that file remains intact. Up to eight modules may be specified in one extract operation for object and macro libraries; however, only one module may be specified in one extract operation for a universal library.

For object and macro libraries, if no modules are specified on the command line, all modules in the library are extracted and concatenated in the output file in alphabetical order.

For universal libraries, RMS fields cannot be extracted to a record oriented device, such as a terminal.

The /EX switch may be applied only to input file specifiers. The format for specifying /EX is:

```
outfile=infile/EX[:modulename1...:modulenamen]
```

outfile

Specifies the file into which extracted modules are to be stored. The default file type for this file is .OBJ if the input modules are object modules; the default file type is .MAC if the input modules are macro modules. If the library is a universal library, the outfile retains the infile type of the module extracted. (However, you are allowed to extract only one universal library module at a time.)

infile

Specifies the library file from which the modules are to be extracted. The default file type for this file is .ULB, .OLB, or .MLB, depending on the current default library type.

/EX

Specifies the Extract switch.

modulename

Specifies the name of the module to be extracted from the library.

Examples

```
LBR>DRIVERS=RSX11M/EX:DXDRV:DKDRV:TTDRV
```

The object modules DXDRV, DKDRV, and TTDRV are concatenated and written into the file DRIVERS.OBJ.

```
LBR>TI:=[1,1]RSXMAC.SML/EX:QIO$S
```

The macro QIO$S is written to the issuing terminal.

```
LBR>TEST.OBS=TEST/EX
```

All of the modules in the library TEST.OLB are written into the file TEST.OBS in alphabetical order.

14.6.8 Insert Switch (/IN) for Object and Macro Libraries

Use the Insert switch to insert library modules into a library file. Any number of input files can be specified. For object libraries and macro libraries, each input file can contain any number of concatenated input modules. For macro libraries, only first-level macro definitions are extracted from the input files. All text outside of the first-level macro definitions is ignored. LBR recognizes only upper-case characters in macro directives. (See Insert Switches for Universal Libraries, Section 14.6.8.1.) The /IN switch is the default library file option, and can be appended only to the library file specifier.

If you attempt to insert an input module which already exists in the library file, the following message is printed on the initiating terminal:

```
LBR -- *FATAL* DUPLICATE MODULE NAME "name" IN filename
```

Likewise, if you attempt to insert a module and a module contains an entry point that duplicates one that is already in the EPT, the following message is printed on the initiating terminal:

```
LBR -- *FATAL* DUPLICATE ENTRY POINT "name" IN filename
```

The format for specifying the Insert switch is:

```
outfile[/IN]=infile1[,infile2,...infilen]
```

outfile

Specifies the library file into which the input modules are to be inserted. The default file type depends on the current default (see Section 14.6.4). It is .OLB if the current default is object libraries, .MLB if the current default is macro libraries.

/IN

Specifies the Insert switch.

infile

Specifies the input file containing modules to be inserted into the library file. The default file type is .OBJ if outfile is an object library and .MAC if outfile is a macro library.

Example

```
LBR>RICKLIB/IN=SHEILA,LAURA,JENNY
```

In this example, the modules contained in the latest versions of files SHEILA, LAURA, and JENNY, which reside in the default directory on SY0:, are inserted into the latest version of the library file RICKLIB, which also resides in the default directory on SY0:. The default filetype for files SHEILA, LAURA, and JENNY is .OBJ if RICKLIB is an object module library, or .MAC if RICKLIB is a macro library.

14.6.8.1 **Insert Switch (/IN) for Universal Libraries** - The Insert switch works basically the same for universal libraries as it does for object libraries and macro libraries. However, when inserting a file into a universal library, the /IN switch is normally applied to the input file. Furthermore, the user can specify module name and descriptive information as switch values in the command line. Also, LBR copies input file attributes to the module header.

The high block indicator (F.HIBK of the file's FDB) and the end of file indicator (F.EFBK of the file's FDB) are included in the input file's user file attributes. LBR makes the high block indicator equal to the end of file indicator in the module header. This means that when a module is extracted to a file, that file will have as many blocks allocated as used.

The Insert switch format for universal libraries is:

```
outfile=infile/IN:name:op:op:op:op
```

outfile

Specifies the universal library into which the file infile is to be inserted.

infile

Specifies the input file to be inserted into outfile. The default for the file type is the value indicated at the universal library's creation time. (See Section 14.6.2.)

/IN

Specifies the Insert switch.

:name

Optionally specifies the module name (up to 6 Radix-50 characters). The default is the first 6 characters of the input file name.

:op

Specifies optional descriptive information (up to 6 Radix-50 characters) to be stored in the module header. The default is null. If only part of the information set is specified, all preceding colons must be supplied.

Example

```
LBR>RICKLIB.ULB=JOE.TXT/IN:MOD1:THIS:IS:JAN2:TEXT
```

In this example, LBR inserts JOE.TXT into the universal library RICKLIB.ULB as MOD1. "THIS", "IS", "JAN2", and "TEXT" are stored in the module header.

You can insert JOE.TXT without the Insert switch and its values. As a result, all the information normally specified by the switch values defaults as described above.

14.6.9 List Switches (/LI, /LE, /FU)

Use the List switches to produce a printed listing of the contents of a library file. Three switches allow you to select the type of listing desired. These switches are as follows:

/LI	Produces a listing of the names of all modules in the library file.
/LE	Produces a listing of the names of all modules in the library file and their corresponding entry points.
/FU	Produces a listing of the names of all modules in the library file and gives a full module description for each: that is, size, date of insertion, and module-dependent information. Appendix B contains sample listings of all three types of library listing.

These switches can be appended only to the output file specifier or the list file specifier.

The /LI switch is the default value; it need not be specified when a listing file has been specified or when any other List switch is included in the command.

The format for specifying List switches is:

```
outfile[,listfile]/switch(es)
```

outfile

Specifies the library file whose contents is to be listed.

listfile

Optionally specifies the listing file. If not specified, the listing is directed to the initiating terminal.

/switch(es)

Specifies the list option(s) selected.

Examples

```
LBR>RICKLIB/LI
```

In this example, a listing of the names of all the modules contained in file SY0:RICKLIB.OLB is printed on the initiating terminal.

```
LBR>RICKLIB/LE
```

In this example, a listing of the names of all the modules and their entry points (contained in file SY0:RICKLIB.OLB) is printed on the initiating terminal.

```
LBR>RICKLIB/FU
```

In this example, a listing of the names of all the modules, and a full description of each module contained in file SY0:RICKLIB.OLB, is printed on the initiating terminal.

```
LBR>DK1:[200,200]RICKLIB,LP.LST:/LE/FU
```

In this example LBR creates file LP.LST in directory [200,200] on DK1: with a listing of all the module names, their entry points, and a full description of each module for file RICKLIB.

14.6.10 **Modify Header Switch (/MH)**

The Modify Header switch pertains only to universal libraries and allows the user to modify the optional user-specified information in the module header.

The format of the switch is:

```
outfile/MH:module:op:op:op:op
```

outfile

Specifies an output file for the universal library. The file type defaults to .ULB.

/MH

Specifies the Modify Header switch.

:module

Specifies the name of the module whose descriptive information is to be modified.

:op

Specifies the optional user information (up to 6 Radix-50 characters) to be stored in the module header. The default is null and indicates that the corresponding information field is not to be changed. Also, entering a pound sign (#) clears the corresponding information field.

Example

The optional descriptive information for module A of RICKLIB.ULB is:

```
"MODA" "FCHCD" "OF" "FCH"
```

The LBR command is:

```
LBR>RICKLIB/MH:A:FCHTS:#::
```

The optional descriptive information for Module A in file RICKLIB is changed to:

```
"FCHTS" "       " "OF" "FCH"
```

14.6.11 Replace Switch (/RP)

Use the Replace switch to replace modules in a library file with input modules of the same name. Any number of input files are allowed, and each file can contain any number of concatenated input modules.

For macro libraries, only first-level macro definitions are extracted from the replacement files. LBR recognizes only upper-case characters in macro directives.

When a match occurs on a module name, the existing module is logically deleted, and all of its entries are removed from the EPT.

As each module in the library file is replaced, a message is printed on the initiating terminal. This message, which contains the name of the module being replaced, is as follows:

```
MODULE "name" REPLACED
```

If the module to be replaced does not exist in the library file, LBR assumes that the input module is to be inserted and automatically inserts it without printing a message.

The /RP switch can be specified in either of the following formats:

- Global format - The /RP switch is appended to the library file specifier, and all of the input files are assumed to contain replacement modules.
- Local format - The /RP switch is appended to an input file specifier, and only the file to which the /RP switch is appended is considered to contain replacement modules.

Global Format

```
outfile/RP=infile1[,infile2,...infilen]
```

outfile

Specifies the library file. The default type depends on the current default (see Section 14.6.4). It is .OLB if the current default is object libraries, or .MLB if the current default is macro libraries.

/RP

Specifies the Replace switch.

infile

Specifies the input file that contains replacement modules for the library file. The default type is .OBJ if outfile is an object library, or .MAC if it is a macro library.

This format of the /RP switch allows you to specify a list of input files without having to append the /RP switch to each of them. To override the global function for a particular input file (that is, to

instruct LBR to process a particular file in a list as a file containing modules to be inserted but not replaced), append /-RP or /NORP to the desired input file specifier.

Local Format

```
outfile=infile1[/RP][,infile2[/RP],...infilen[/RP]]
```

outfile

Specifies the library file. The local format default is the same as the global format default described above.

infile

Specifies the input file which will replace modules in the output library file. The local format default is the same as the global format default described above.

/RP

Specifies the Replace switch. Appending the /RP switch to an input file specifier constitutes the local format of the switch. This overrides the LBR default (Insert) and instructs LBR to treat the module(s) contained in the specified file as modules to be replaced.

Examples

The files used in the following four examples, and the modules contained within each file, are depicted in Figure 14-8. For the examples, these files are assumed to reside in the default directory on the default device, and the initial state of the library file is assumed to be as shown in Figure 14-8.

1. LBR>RICKLIB/RP=SHEILA,LAURA,JENNY

```
MODULE "SHEILA" REPLACED
MODULE "LAURA1" REPLACED
MODULE "LAURA2" REPLACED
MODULE "JENNY1" REPLACED
MODULE "JENNY2" REPLACED
```

In this example, the global format for the /RP switch is used. Object modules from the input files SHEILA, LAURA, and JENNY replace modules by the same names in the library file named RICKLIB and modules JENNY3 and LAURA3 are inserted. The resulting library file is shown in Figure 14-9.

2. LBR>RICKLIB=CHRIS,SHEILA/RP

```
MODULE "SHEILA" REPLACED
```

In this example, the local format of the /RP switch is used. The object module SHEILA from file SHEILA is replaced in the library file RICKLIB. The object modules in the file CHRIS are inserted in the library file. (See Insert switch in Section 14.4.8.) The resulting library file is shown in Figure 14-10.

3. LBR>RICKLIB/RP=SHEILA,LAURA,JENNY,CHRIS/-RP

```
MODULE "SHEILA" REPLACED
MODULE "LAURA1" REPLACED
MODULE "LAURA2" REPLACED
MODULE "JENNY1" REPLACED
MODULE "JENNY2" REPLACED
```

In this example, the /-RP switch is used to override the global format of the command. Object modules in files SHEILA, LAURA, and JENNY are processed as modules to be replaced, and file CHRIS is processed as a file which contains modules to be inserted. The resulting library file is shown in Figure 14-11.

4. LBR>RICKLIB/RP=SHEILA,LAURA/-RP,JENNY

```
MODULE "SHEILA" REPLACED
LBR -- *FATAL* -- DUPLICATE MODULE "LAURA1" IN LAURA.OBJ;1
```

In this example, only module SHEILA from file SHEILA was replaced. The user specified that the modules in file LAURA not be replaced (/-RP), but inserted. One of the modules contained in file LAURA duplicated an already existing module in file RICKLIB (see Figure 14-8). Therefore, LBR issued the fatal error message and terminated the processing of the current command.

	Output Library File	Input Files			
File Name	RICKLIB.OLB;1	SHEILA.OBJ;1	LAURA.OBJ;1	JENNY.OBJ;1	CHRIS.OBJ;1
	JENNY1	SHEILA	LAURA1	JENNY1	CHRIS1
Object	JENNY2		LAURA2	JENNY2	CHRIS2
Modules	LAURA1		LAURA3	JENNY3	
	LAURA2				
	SHEILA				

Figure 14-8 Sample Files Used in LBR Examples 1-4

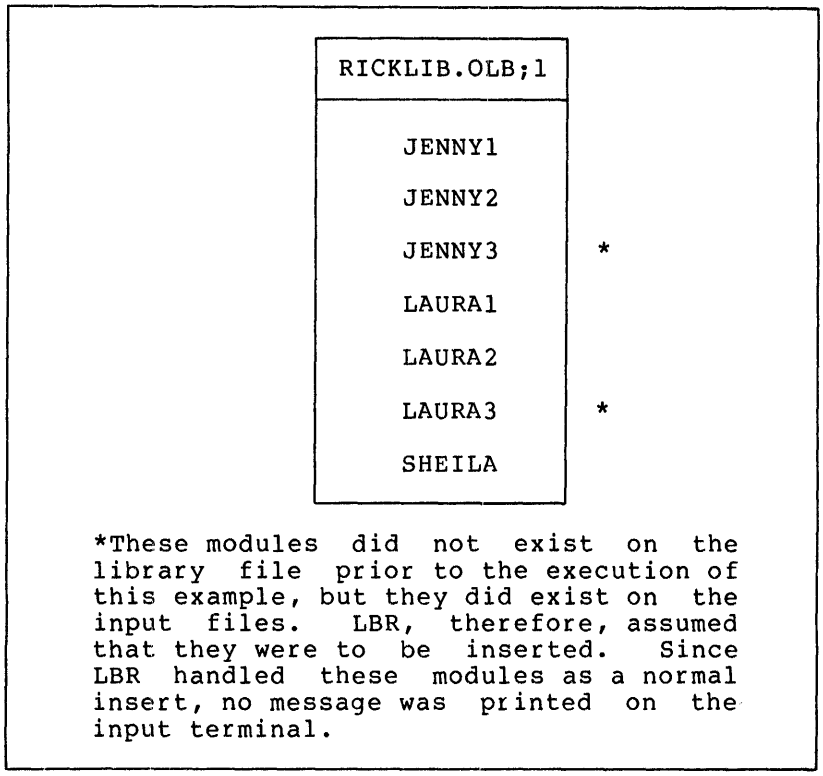


Figure 14-9 Output Library File After Execution of Example 1

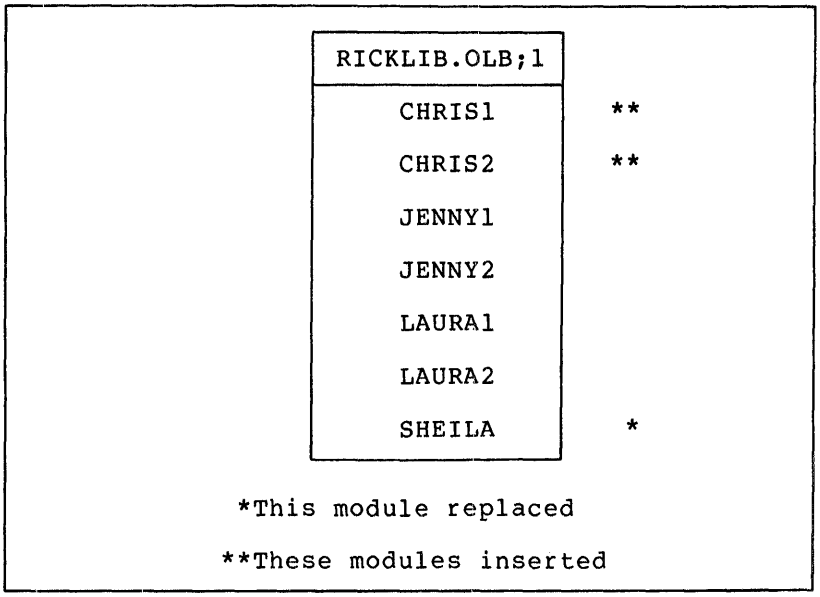


Figure 14-10 Output Library File After Execution of Example 2

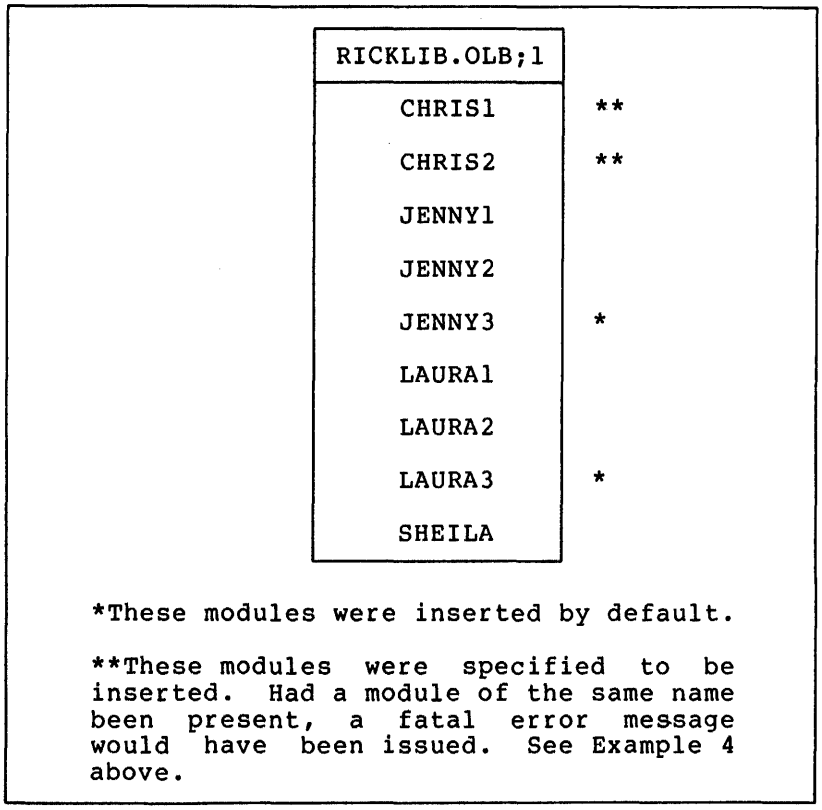


Figure 14-11 Output Library File After Execution of Example 3

14.6.11.1 **Replace Switch (/RP) for Universal Libraries** - Use the /RP switch for universal libraries in the same way as for macro and object libraries. In addition, you can specify the same values for the Replace switch as for the Insert switch for universal libraries. (See Section 14.6.8.1). However, you can specify the /RP switch with either the infile or the outfile. (See the example below).

The global Replace switch format for universal libraries is:

```
outfile/RP:name:op:op:op:op=infile[,infile2,....infilen]
```

The local Replace switch format for universal libraries is:

```
outfile=infile/RP:name:op:op:op:op[,infile2....infilen]
```

outfile

Specifies the universal library file.

infile

Specifies the input file which will replace modules in the library file. The default for the file type is the value indicated at the universal library's creation time. (See Section 14.6.2).

/RP

Specifies the Replace switch.

:name

Optionally specifies the module name to be replaced (up to 6 Radix-50 characters). The default is the first six characters of the infile name.

:op

Specifies optional descriptive information (up to 6 Radix-50 characters) to be stored in the module header. The default is null. If only part of the information set is specified all preceding colons must be supplied.

Example

```
LBR> TEXT.ULB=DEBBIE.TXT/RP::THIS:IS:JAN3:UPDATE

MODULE "DEBBIE" REPLACED
```

In this example, LBR replaces the DEBBIE module in the universal library TEXT.ULB with an updated module from file DEBBIE.TXT. The date of replacement is specified by the user optional information and inserted in the module header. Note that the optional name is omitted.

The initial state of the library file is shown in Figure 14-12. The resulting library file is shown in Figure 14-13.

	Output Library File	Input Files
File Name	TEXT.ULB;1	DEBBIE.TXT
Modules	DEBBIE BERNIE	

Figure 14-12 Sample Files for Universal Library Replace Example

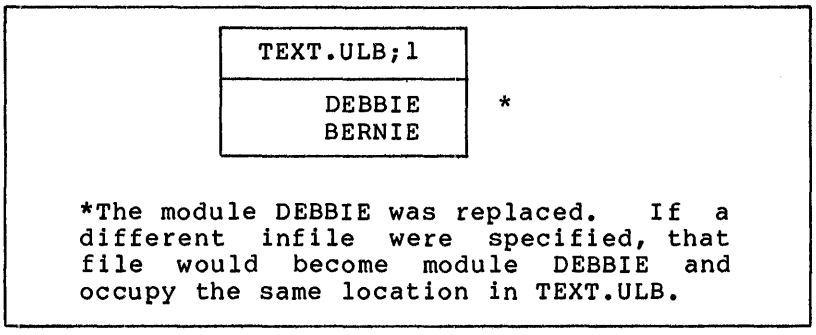


Figure 14-13 Output Library File After Execution of Universal Library Replace Example

14.6.12 Spool Switch (/SP)

The Spool switch is the list file default switch. Whether the switch is specified or not, the results are the same, that is, the listing file is spooled to the line printer.

After the listing file is created, a request is made to the print spooler task to print the spooled file; printing is performed asynchronously (see Appendix C for a description of the spooler task).

The automatic printing of the listing file can be inhibited by specifying a minus sign (-) or the letters NO between the slash (/) and the SP in the spool switch (/-SP or /NOSP). This causes the listing file to be created, but the request to the print spooler task is not issued. Therefore, the file is not automatically printed.

The /SP switch can only be appended to the list file specifier.

The format for specifying the Spool switch is:

outfile,listfile[/SP] or [/-SP]

outfile

Specifies the library file.

listfile

Specifies the listing file.

/SP or /-SP

Specifies the Spool switch.

Example

```
LBR>RICKLIB/DE:SHEILA,RICKLST/-SP
```

In this example, the following occurs:

1. The module SHEILA and its associated entry points are deleted from the library file SY:RICKLIB.

2. The listing of the contents of the resulting library file RICKLIB is written to the list file SY:RICKLST.LST. Since the /-SP switch is specified, the file is not automatically printed.

14.6.13 Selective Search Switch (/SS)

Use the Selective Search switch to set the selective search attribute bit in the module header of object modules as they are inserted into an object library. The switch has no effect when applied to modules being inserted into a macro library. The switch may be specified only on input files for insertion or replacement operations, and it affects all modules in the input file to which it is applied.

Object modules with the selective search attribute are given special treatment by the Task Builder. Global symbols defined in modules with the selective search attribute are only included in the Task Builder's symbol table if they are previously referenced by other modules. Thus, only referenced symbols will be listed with the module in the Task Builder memory allocation file, thereby reducing task build time. The /SS switch should only be applied to object files whose modules contain only absolute (not relocatable) symbol definitions. See the RSX-11M/M-PLUS Task Builder Manual for more information.

The format for specifying the Selective Search switch is:

```
outfile=infile1/SS[,infile2[/SS],...infilen[/SS]]
```

outfile

Specifies the library file.

infile

Specifies the input file that contains modules to be selectively searched.

/SS

Specifies the Selective Search switch.

14.6.14 Squeeze Switch (/SZ)

Use the Squeeze switch to reduce the size of macro definitions by eliminating all trailing blanks and tabs, blank lines, and comments from macro text. The /SZ switch is used to conserve memory in the MACRO-11 Assembler and to reduce the size of macro library files. The Squeeze switch has no effect on object libraries or universal libraries.

The /SZ switch can be specified in either of the following formats:

- Global format - The /SZ switch is appended to the library file specifier, and all of the input files are assumed to contain modules to be squeezed.
- Local format - The /SZ switch is appended to an input file specifier, and only the file to which the /SZ switch is appended is considered to contain modules to be squeezed.

Global Format

```
outfile/SZ=infile1[,infile2,...infilen]
```

outfile

Specifies the library file.

/SZ

Specifies the Squeeze switch.

infile

Specifies the input file that contains modules to be squeezed during insertion into the library file.

Use this format of the /SZ switch to specify a list of input files without having to append the /SZ switch to each of them. To override the global function for a particular input file (that is, to instruct LBR to process a particular file in a list as a file containing modules to be inserted but not squeezed), append /-SZ or /NOSZ to the desired input file specifier.

Local Format

```
outfile=infile1/SZ[,infile2[/SZ],...infilen[/SZ]]
```

outfile

Specifies the library file.

infile

Specifies the file that contains modules to be squeezed during insertion into the library file.

/SZ

Specifies the Squeeze switch.

LBR uses the following algorithm on each line to be squeezed and then inserts the resulting line into the library file:

1. The line is examined for the rightmost semicolon (;).
2. If a semicolon is located, it is deleted, along with all trailing characters in the line.
3. All trailing blanks and tabs in the line are deleted.
4. If the resulting line is null, nothing is transferred to the library file.

If the line contains a semicolon embedded in meaningful (non-comment) text and you want comments squeezed, code a dummy comment for that line. /SZ uses only this rightmost comment during squeeze processing.

Example

Figure 14-14 illustrates the use of the LBR /SZ switch. A file containing input text to be squeezed is illustrated, along with the text actually inserted into the library file after the squeeze operation has been completed.

BEFORE BEING SQUEEZED

```
        .MACRO  MOVSTR RX,RY,?LBL

;***    - - NOTE :                                  ;
;       BOTH ARGUMENTS MUST BE REGISTERS            ;

LBL:    MOVB    (RX)+,(RY)+     ;MOVE A CHARACTER
        BNE     LBL             ;CONTINUE UNTIL NULL SEEN
        DEC     RY              ;BACKUP OUTPUT PTR TO NULL

;END OF MOVSTR
        .ENDM
```

AFTER BEING SQUEEZED

```
        .MACRO  MOVSTR RX,RY,?LBL
;***    - - NOTE :
;       BOTH ARGUMENTS MUST BE REGISTERS
LBL:    MOVB    (RX)+,(RY)+
        BNE     LBL
        DEC     RY
        .ENDM
```

Figure 14-14 MACRO Listing Before and After Running LBR with /SZ Switch

14.7 COMBINING LIBRARY FUNCTIONS

Two or more library functions may be requested in the same command line. The only exceptions are that /CO cannot be requested with anything else except /LI, and /CR and /DE cannot be specified in the same command line.

Functions are performed in the following order:

1. /DF
2. /CR
3. /DE
4. /DG
5. /MH
6. /IN, /RP, /SS, /SZ, /EP
7. /CO
8. /EX
9. /LI, /LE, /FU, /SP

Example

```
LBR>FILE/DE:XYZ:$A,LP.LST:/LE/FU=MODX,MODY/RP
```

Functions are performed in order, as:

1. Delete modules XYZ and $A.
2. Insert all modules from MODX and replace duplicate modules of MODY.
3. Produce a listing of the resultant library file on the line printer with full module descriptions and all entry points.

14.8 LBR RESTRICTIONS

The following restrictions apply when using LBR:

- Limit of 65,536(10) words per module.
- Limit of 65,536(10) blocks per library.
- Tables should be allocated to maximum anticipated size. Expanding table allocations requires using Compress to copy the entire file.
- A fatal error results if an attempt is made to insert a module into a library that contains a module with a different name from, but with the same entry point as, the inserted module. For further information, refer to the discussion of the /IN switch in Section 14.6.8.
- The use of wildcards in file specifiers is not allowed (that is, forms such as *.OBJ, where the * indicates all modules with type .OBJ).

There must be enough space in the library's tables for both the modules being replaced and their replacements, since the new modules are entered and the old modules are logically but not physically deleted.

14.9 LBR ERROR MESSAGES

There are two types of LBR error messages: diagnostic and fatal.

Diagnostic error messages describe a condition that exists that requires consideration, but the nature of the condition does not warrant termination of the command. Diagnostic messages are issued to TI:, in the format:

```
LBR -- *DIAG* - message
```

Fatal error messages describe a condition that exists that caused LBR to terminate the processing of a command. When this occurs, LBR returns to the highest level of command input. For example, if the command is entered in response to the MCR prompt, that is,

```
>LBR command
```

then, LBR issues the fatal error message and exits. If, however, the command is entered in response to the LBR prompt, that is,

```
LBR>command
```

LBR issues the fatal error message and reprompts.

Fatal error messages are issued to TI:, in the format:

```
LBR -- *FATAL* - message
```

If a fatal error occurs during the processing of an indirect command file, the command file is closed, the fatal error message - followed by the command line in error - is issued to TI:, and LBR returns to the highest level of command input.

14.9.1 Effect of Fatal Errors on Library Files

The status of a library file after fatal errors is:

1. In general, output errors leave the library in an indeterminate state.
2. During the deletion process, the library is rewritten prior to the printing of the individual module-/entry-point-deleted messages.
3. During the replacement process, the library is rewritten prior to the printing of the individual module-replaced messages.
4. During the insert process, the library is rewritten after the insertion of all modules in each individual input file, that is, between input files.

14.9.2 LBR Error Messages

LBR -- BAD LIBRARY HEADER

Explanation: Either the file is not a library file or the file is corrupted.

User Action:

- If the file is not a library file, reenter the command line with a proper library file specified.
- If the file is a proper library file, the user should run the file structure verification utility (VFY) against the volume to determine if it is corrupted (see Chapter 13).
- If the volume is corrupted, it must be reconstructed before it can be used.

LBR -- CANNOT MODIFY HEADER

Explanation: An attempt was made to modify the module header of a module in an object library or macro library. No change is made to the module header.

User Action: Reenter the command line, specifying a module in a universal library.

LBR -- COMMAND I/O ERROR

Explanation: One of the following conditions may exist:

- A problem exists on the physical device (for example, not cycled up).
- The file is corrupted or the format is incorrect (for example, record length exceeds 132 bytes).

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

LBR -- COMMAND SYNTAX ERROR
command line

Explanation: A command was entered in a format that does not conform to syntax rules.

User Action: Reenter the command line, using the correct syntax.

LBR -- DUPLICATE ENTRY POINT NAME "name" IN filename

Explanation: An attempt has been made to insert a module into a library file when both contain an identically-named entry point.

User Action: Determine if the specified input file is the correct file. If not, reenter the command line, specifying the correct input file. If the input file is the correct file, the user may delete the duplicate entry point from the library and reenter the command line.

LBR -- DUPLICATE MODULE NAME "name" IN filename

Explanation: An attempt has been made to insert (without replacement) a module into a library that already contains a module with the specified name.

User Action: Determine if the specified input file is the correct file. If the input file is correct, decide whether to delete the duplicate module from the library file and insert the new one, or replace the duplicate module by rerunning LBR with the /RP switch appended to the input file specifier.

LBR -- EPT OR MNT EXCEEDED IN filename

Explanation: The EPT or MNT table limit has been reached during the execution of an Insert or Replace command.

User Action: Copy the library, increasing the table space via the Compress switch. Reenter the command line.

LBR -- EPT OR MNT SPACE EXCEEDED IN COMPRESS

Explanation: An EPT or MNT table size was specified for the output library file that is not large enough to contain the EPT or MNT entries used in the input library file.

User Action: Reenter the command line with a larger EPT or MNT table size specified.

LBR -- ERROR IN LIBRARY TABLES, FILE filename

Explanation: The library file is corrupted or is not a library file.

User Action: If the file is corrupted, no recovery is possible; the file must be reconstructed. If the file is not a library file, reenter the command line with the correct library file specified.

LBR -- EXACTLY ONE INPUT FILE MUST APPEAR WITH /CO

Explanation: No file or more than one input library file was specified when using the /CO switch.

User Action: Reenter the command line with only one input file specified.

LBR -- FATAL COMPRESS ERROR

Explanation: The input library file is corrupted or is not a library file.

User Action: No recovery is possible. The file in question must be reconstructed.

LBR -- GET TIME FAILED

Explanation: This error occurs when LBR attempts to execute a Get Time Parameters directive and fails. The error is caused by a system malfunction.

User Action: Reenter the command line. If the problem persists, submit a Software Performance Report along with the related console dialogue and any other pertinent information.

LBR -- ILLEGAL DEVICE/VOLUME
command line

Explanation: Device specifier entered does not conform to syntax rules. A device specifier consists of 2 ASCII characters, followed by one or two optional octal digits.

User Action: Reenter the command line with the correct device syntax specified.

LBR -- ILLEGAL DIRECTORY
command line

Explanation: UFD entered does not conform to syntax rules. UFD syntax consists of a left square bracket, followed by one to three octal digits, a comma, one to three octal digits, and terminated by a right square bracket.

User Action: Reenter the command line with the correct UFD syntax.

LBR -- ILLEGAL FILENAME
command line

Explanation: One of the following was entered:

- A file specifier that contains a wildcard.
- A file specifier that contains neither a file name nor a file type.

User Action: Reenter the command line correctly.

LBR -- ILLEGAL GET COMMAND LINE ERROR CODE

Explanation: The system, for some reason, is unable to read a command line. This is an internal system failure.

User Action: Reenter the command line. If the problem persists, submit a Software Performance Report along with the related console dialogue and any other pertinent information.

LBR -- ILLEGAL SWITCH
command line

Explanation: A non-LBR switch was specified or a legal switch was specified in an invalid context.

User Action: Reenter the command line with the correct switch specification.

LBR -- ILLEGAL SWITCH COMBINATION

Explanation: Switches were entered that cannot be executed in combination. See Section 14.7.

User Action: Reenter the command line, specifying the switches in the proper sequence.

LBR -- INDIRECT COMMAND SYNTAX ERROR
command line

Explanation: An indirect file was specified in a format that does not conform to syntax rules.

User Action: Reenter the command line with the correct syntax.

LBR -- INDIRECT FILE DEPTH EXCEEDED
command line

Explanation: An attempt has been made to exceed one level of indirect command files.

User Action: Rerun the job with only one level of indirect command file.

LBR -- INDIRECT FILE OPEN FAILURE
command line

Explanation: The requested indirect command file does not exist as specified. One of the following conditions may exist:

- The user directory area is protected against access.
- A problem exists on the physical device (for example, device cycled down).
- The volume is not mounted.
- The specified file directory does not exist.
- The file does not exist as specified.
- Insufficient dynamic memory in Executive.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

LBR -- INPUT ERROR ON filename

Explanation: The file system, while attempting to process an input file, has detected an error. A problem exists with the physical device (for example, the device cycled down).

User Action: Reenter the command line.

LBR -- INSUFFICIENT DYNAMIC MEMORY TO CONTINUE

Explanation: The partition in which LBR is running is too small.

User Action: Remove the task (LBR), install it in a larger partition, and reenter the command line. (See the MCR RUN /INC command description in the RSX-11M/M-PLUS MCR Operations Manual.)

LBR -- INVALID EPT AND/OR MNT SPECIFICATION

Explanation: An EPT or MNT value greater than 4096(10) was entered in a /CR or /CO switch.

User Action: Reenter the command line with the correct value specified.

LBR -- INVALID MODULE FORMAT, insertion module

Explanation: An attempt was made to insert a macro module into an object library.

User Action: Determine if an object file was to be inserted into an object library. If so, reenter the command line with the correct object file. If a macro library was to receive the insertion, reenter the command line with the correct macro library.

LBR -- INVALID FORMAT, INPUT FILE filename

Explanation: The format of the specified input file is not the standard format for a macro source or object file, or the input file is corrupted.

User Action: Reenter the command line with the correct input file specified.

LBR -- INVALID OPERATION FOR OBJECT AND MACRO LIBRARIES

Explanation: Module header information was supplied for an object library or macro library in an Insert or Replace command.

User Action: No action required. The command will be executed as if the information had not been supplied.

LBR -- INVALID LIBRARY TYPE SPECIFIED

Explanation: An illegal library type was specified when using the Create or Default switch. The values OBJ, MAC, and UNI are the only valid specifications. See Sections 14.6.2 and 14.6.4.

User Action: Reenter the command line with OBJ, MAC, or UNI specified.

LBR -- INVALID NAME -- "name"

Explantion: A module name that contains a non-Radix-50 character has been specified for deletion, insertion, or replacemant of a module in a universal library or in a macro module, or a module name has been specified for modification of a universal module header. Radix-50 characters consist of the letters A through Z, the numbers 0 through 9, and the special characters period (.) and dollar sign ($).

User Action: Reenter the command line with a valid name.

LBR -- INVALID RAD50 CHARACTER IN "character string"

Explanation: A character supplied as part of information when using the Insert, Replace, or Modify Header switch for a universal library is not a Radix-50 character.

User Action: Determine which character of the corresponding switch value is not a Radix-50 character. Reenter a Radix-50 character in place of the invalid character.

LBR -- I/O ERROR ON INPUT FILE filename

Explanation: A read error has occurred on an input file. One of the following conditions may exist:

- A problem exists on the physical device (for example, not cycled up).
- The file is corrupted or the format is wrong (record length exceeds 132 bytes).

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter that command line.

LBR -- LIBRARY FILE SPECIFICATION MISSING

Explanation: A command was entered without specifying the library file.

User Action: Reenter the command line with the library file specified.

LBR -- MARK FOR DELETE FAILURE ON LBR WORK FILE

Explanation: When LBR begins processing commands, it automatically creates a work file and marks it for delete. For some reason, this marking for delete failed.

The work file constitutes a lost file, because it does not appear in any file directory.

User Action: The file may be deleted by running the file structure verification utility (VFY) (see Chapter 13).

LBR -- MULTIPLE MODULE EXTRACTIONS NOT PERMITTED FOR UNV MODULES

Explanation: An attempt was made to extract more than one module from a universal library. The first module specified is extracted but others are ignored.

User Action: Reenter the command line for each additional extraction.

LBR -- NO ENTRY POINT NAMED "name"

Explanation: The entry point to be deleted is not in the specified library file.

User Action: Determine if the entry point is misspelled or if the wrong library file is specified. Reenter the command line with the entry point correctly specified.

LBR -- NO MODULE NAMED "module"

Explanation: The module to be deleted is not in the specified library file.

User Action: Determine if the module name is misspelled or if the wrong library file is specified. Reenter the command line with the module name correctly specified.

LBR -- OPEN FAILURE ON FILE filename

Explanation: The file system, while attempting to open a file, has detected an error. One of the following conditions may exist:

- The user directory area is protected against an open.
- A problem exists on the physical device (for example, device cycled down).
- The volume is not mounted.
- The specified file directory does not exist.
- The file does not exist as specified.
- Insufficient contiguous space to allocate the library file (Compress and Create only).
- Insufficient dynamic memory in Executive.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter that command line.

LBR -- OPEN FAILURE ON LBR WORK FILE

Explanation: The file system, while attempting to open the LBR work file, has detected an error. The LBR work file is created on the volume from which LBR was installed. One of the following conditions may exist:

- The volume is full.
- The device is write-protected.
- A problem exists with the physical device.
- Insufficient dynamic memory in Executive.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

LBR -- OUTPUT ERROR ON filename

Explanation: A write error has occurred on the output file. One of the following conditions may exist:

- The volume is full.
- The device is write-protected.
- The hardware has failed.

User Action: If the volume is full, delete all unnecessary files and rerun LBR. If the device is write-protected, write-enable the device, and reenter the command line. If the hardware has failed, swap devices and reenter the command line or wait until the device is repaired and rerun LBR.

LBR -- POSITIONING ERROR ON filename

Explanation: The device is write-locked.

User Action: If the device is write-locked, write-enable it and reenter the command line.

LBR -- RMS FILES CANNOT BE EXTRACTED TO RECORD ORIENTED DEVICE

Explanation: An attempt was made to extract to a record-oriented device a module inserted from a non-sequential RMS file. This is a fatal error message.

User Action: Extract the file to a disk and then use an RMS conversion to make an RMS sequential file.

LBR -- TOO MANY OUTPUT FILES SPECIFIED

Explanation: More than two output files were specified; LBR makes the following assumptions:

- The first output file specified is the output library file.
- The second output file specified is the listing file.
- The third through n files specified to the left of the equal sign are ignored.

User Action: No action is required. LBR continues as though the extra file(s) were not specified.

LBR -- VIRTUAL STORAGE REQUIREMENTS EXCEED 65536 WORDS

Explanation: This error may occur with maximum size libraries in conjunction with a single command line which logically deletes a large number of modules and entry points, and continues to replace them with an equally large number of modules and entry points having highly dissimilar names. Normally, this message indicates some sort of internal system error.

User Action: Rerun the job, but divide the complicated command line into several smaller command lines that do the same operations.

LBR -- WORK FILE I/O ERROR

Explanation: A write error has occurred on the LBR work file. One of the following conditions may exist:

- The volume is full.
- The device is write-protected.
- The hardware has failed.

User Action: If the volume is full, delete all unnecessary files and rerun. If the device is write-protected, write enable the device, and reenter the command line. If the hardware has failed, swap devices and retry the command, or wait until the device is repaired and rerun LBR.

CHAPTER 15

FILE DUMP UTILITY (DMP)

15.1 INTRODUCTION TO DMP

The File Dump utility (DMP) program produces a printed listing of the contents of a file or volume. The listing can be directed to any suitable output device -- line printer, terminal, magnetic tape, DECtape, or disk. DMP runs in two modes, file mode and device mode.

DMP normally handles physical records of up to 2048 bytes in length. If a physical record is larger than 2048 bytes, use the following procedure:

```
>RUN $DMP /INC=n
```

where n is at least equal to the number of words in the record minus 1024. The value of n is interpreted as an octal number unless it is followed by a decimal point.

DMP can be invoked using any of the methods described in Chapter 1.

15.1.1 File Mode

In file mode, one input file is specified and all, or a specified range (see /BL:n:m) of virtual blocks of the named file is dumped. The input device must be a Files-11 structured volume and must be mounted via the MCR Mount command. M-PLUS users must mount the volume via the MCR Mount/Foriegn command. (For a description of MOUNT, see the RSX-11M/M-PLUS MCR Operations Manual.)

A virtual block refers to a block of data in a file. Virtual blocks are numbered sequentially from 1 through n, where n is the total number of virtual blocks in the file.

15.1.2 Device Mode

In device mode, only the input device is specified, and a specified range (/BL:n:m) of logical blocks is dumped. The /BL:n:m switch is a required parameter.

A logical block refers to an actual 512-byte block on disk and DECtape, and physical records on magnetic tape and cassette. Logical blocks are numbered from 0 to n-1, where n is the total number of logical blocks on the device.

When DMP is in device mode, the volume to be dumped must not be mounted via the MCR Mount command.

15.2 DMP COMMAND STRING

Enter DMP command strings in the following format:

```
[outfile][/sw][/sw...][=infile][/sw][/sw...]
```

outfile

Specifies the output file. If the output file name and file type are null, DMP creates the file DMPFIL.DMP. Later defaults for DMPFIL.DMP will take the version number n+1.

/sw

Specifies one of the switches listed in Section 15.3. Unless otherwise indicated in a switch description, all switches can be applied either to the input file or to the output file with equal effect.

infile

Specifies the input file. The equal sign and the input file name and file type are required; DMP does not provide a default for either the file name or type. The input file version number defaults to the latest version.

For a complete description of file specifications see Section 1.3.

15.3 DMP SWITCHES

DMP switch specifications consist of a slash (/) followed by a switch name, optionally followed by a value, which is separated from the switch by a colon (:). DMP functions are implemented by means of the switches described in Table 15-1.

Table 15-1
DMP Switches

Switch	Description
Default	Word mode octal dump.
/AS	Specifies that the data should be dumped in ASCII mode. The control characters (0-37) are printed as ^, followed by the alphabetic character corresponding to the character code plus 100. For example, bell (code 7) is printed as ^G (code 107). Lower case characters (140-177) are printed as %, followed by the corresponding upper case character (character code minus 40).

(continued on next page)

Table 15-1 (Cont.)
DMP Switches

Switch	Description
/BA:n:m	Specifies a 2-word base block address, where n = high-order base block address (octal), and m = low-order base block address (octal). When specified, all future block numbers will be added to this value to obtain an effective block number. This switch is useful for specifying block numbers that exceed 16 bits. For example: DMP>/BA:1:0 specifies that all future block numbers will be relative to 65536(10) (200000(8)). DMP>/BA:0:0 clears the base address. When the /BA switch appears in a command line, no blocks are dumped; the only result of the command is to set the base address.
/BL:n:m	Specifies the range of blocks to be dumped, where n is the first block and m is the last block. In file mode only, the /BL switch is not required. If /BL is not specified, DMP will dump all blocks of the specified file, relative to the current address. If the /BL:n:m switch is specified in file mode, it specifies the range of virtual blocks to be dumped. If the /BL:n:m switch is specified as /BL:0 in file mode, no virtual blocks are dumped. This is useful for dumping only the header portion of the file (see /HD). The /BL:n:m switch is a required parameter in device mode. When used in device mode, it specifies the range of logical blocks to be dumped. Magnetic tapes and cassettes, when dumped in device mode, are always dumped starting with the current tape position; that is, the values given with the /BL switch are ignored. The switch values are used, however, to label the pages of the dump listing, and to determine the number of blocks to dump. When a switch value of /BL:n:m is specified, m-n+1 blocks are dumped, starting at the current tape position.
/BY	Specifies that the data should be dumped in byte octal format.

(continued on next page)

Table 15-1 (Cont.)
DMP Switches

Switch	Description
/DC	Specifies that the data should be dumped in decimal word format.
/DENS:n	Specifies the density of a TU16 input magnetic tape when DMP is in device mode only. The value for n can be 800 or 1600. DMP does not automatically determine the density of an input tape. If the /DENS switch is not specified in a DMP command, DMP attempts to read an input tape at the density currently set in the tape controller. (See the RSX-11M/M-PLUS MCR Operations Manual for a description of the MCR Mount command and its /DENS switch).
/FI:file-number: sequence-number	The file number, used as input file identifier, can be used instead of a file name as a file identifier.
/HD[:F or :U]	This switch is an optional parameter to be used in file mode. If specified, /HD causes the file header as well as the specified portion of the file to be dumped. In addition, this switch has two options. "F", the default, causes a Files-11 formatted dump of the header. "U" specifies an unformatted octal dump. An octal dump also occurs when DMP is used on non-Files-11 headers. If you want only the header portion of the file to be dumped, specify: /HD/BL:0 The file header is described in Appendix F of the IAS/RSX-11 I/O Operations Reference Manual.
/HF	This switch formats data blocks which have Files-11 header structure. Other blocks are output as an unformatted octal dump. Example: DMP> HEAD.LST=[0,0]INDEXF.SYS/HF This example generates a dump of the index file INDEXF.SYS and formats all the headers in the file.

(continued on next page)

Table 15-1 (Cont.)
DMP Switches

Switch	Description
/HX	Specifies that the data be dumped in hexadecimal byte format. Note that a hexadecimal dump reads from right to left.
/ID	Causes DMP's version to be identified. This switch may be specified on a line by itself at any time. Example: DMP>/ID DMP--DMP VERSION M0006.05
/LB	Logical block; specifies that only the starting block number and a contiguous or noncontiguous indication for the file should be displayed. Example: DMP>TI:=DK0:RICKSFILE.DAT;3/LB STARTING BLOCK NUMBER = 0,135163 C File RICKSFILE.DAT, version 3 is a contiguous file starting at block number 0,135163. (See /BA:n:m for block number description.)
/LW	Specifies that the data be dumped in hexadecimal double-word format.
/MD[:n]	Memory dump; allows control of line numbers. Line numbers are normally reset to zero whenever a block boundary is crossed. The /MD [:n] switch allows lines to be numbered sequentially for the full extent of the file; that is, the line numbers are not reset when block boundaries are crossed. The optional value (:n) specifies the value of the first line number. The default is 0. The /MD switch is used with the output file.
/R5	Specifies that data be dumped in Radix-50 format words.
/RC	Specifies that data be dumped a record at a time (rather than a block at a time). The data format is determined by setting one of the format switches, /AS, /DC, /HX, /LW, /R5, or /WD. The largest record DMP can process is limited by the amount of space available to the DMP task; 512 (decimal) bytes are allocated in the task image. To increase the amount of space available, use the MCR Install command /INC switch. For example, to dump a file with 1024-byte records, /INC=256. (at least) must be specified.

(continued on next page)

Table 15-1 (Cont.)
DMP Switches

Switch	Description
/RW	Causes DMP to issue rewind command before referencing a specified tape. This switch may be specified at any time to reposistion a tape at the load point (BOT).
/SP	Causes the dump output file to be spooled to the line printer. /SP may only be specified on the output file specification; it is illegal on the input file specification. Spooled files may be deleted after printing.
/WD	Specifies that the data be dumped in hexadecimal word format.

15.4 DMP ERROR MESSAGES

DMP -- BAD DEVICE NAME

Explanation: An invalid device name was specified in a file specification.

User Action: Reenter the command line, specifying the correct device.

DMP -- BLOCK SWITCH REQUIRED IN LOGICAL BLOCK MODE

Explanation: Self-explanatory -- /BL switch must be specified.

User Action: Reenter the command line with the /BL switch specified.

DMP -- CANNOT FIND INPUT FILE

Explanation: The requested file cannot be located in the specified directory.

User Action: Reenter the command with the correct file name and UIC specified.

DMP -- COMMAND SYNTAX ERROR

Explanation: A command was entered in a format that does not conform to syntax rules.

User Action: Reenter the command line with the correct syntax.

DMP -- FAILED TO ASSIGN LUN

Explanation: An illegal device was entered in a file specification.

User Action: Reenter the command line with the correct device specified.

DMP -- FAILED TO READ ATTRIBUTES

Explanation: You attempted to access a file for which you did not have read access privileges.

User Action: Rerun DMP under a UIC which has read access privileges to the file.

DMP -- ILLEGAL DENSITY VALUE

Explanation: A value other than 800. or 1600. was specified in the DMP /DENS switch.

User Action: Reenter the /DENS switch with the proper value.

DMP -- ILLEGAL SWITCH

Explanation: You specified a switch that is not a valid DMP switch, or used a legal switch in an invalid manner.

User Action: Reenter the command line with the correct switch specified.

DMP -- ILLEGAL VALUE ON /HD SWITCH

Explanation: You entered an option other than F or U for the /HD switch.

User Action: Reenter the command line with the correct option specified.

DMP -- I/O ERROR ON INPUT FILE

or

DMP -- I/O ERROR ON OUTPUT FILE

Explanation: One of the following conditions exists:

- A problem exists on the physical device (for example, device cycled down).
- File is corrupted or the format is incorrect.
- Output volume is full.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

DMP -- NO INPUT FILE SPECIFIED

Explanation: A command line was entered with no input file specification.

User Action: Reenter the command line with an input file specified.

DMP -- NO LISTS OR WILD CARDS ALLOWED

Explanation: Either a command with more than one input or output file name was entered, or a wildcard was entered as a file specification.

User Action: Reenter the command line with only one input file specification and one output file specification. No wildcard specifications are allowed.

DMP -- OPEN FAILURE ON INDIRECT FILE

Explanation: The requested indirect command file does not exist as specified. One of the following conditions may exist:

- The file is protected against access.
- A problem exists on the physical device (for example, device cycled down).
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

DMP -- OPEN FAILURE ON INPUT FILE

or

DMP -- OPEN FAILURE ON OUTPUT FILE

Explanation: One of the following conditions may exist:

- The file is protected against access.
- The named file does not exist in the specified directory.
- The volume is not mounted.
- The specified file directory does not exist.
- A problem exists on the physical device (for example, device cycled down).

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

PART 7
PROGRAM MAINTENANCE UTILITIES

CHAPTER 16

THE FILE COMPARE UTILITY (CMP)

The File Compare Utility (CMP) compares two ASCII text files. The files are compared line by line to determine whether parallel records are identical. Using CMP, you can perform the following file-compare functions:

- Generate a listing showing the differences between the two files. Each difference is listed as a pair; first, the lines from the first file that are being compared to lines in the second file, then the lines from the second file.
- Generate a listing in the form of one list, with differences marked by change bars.
- Generate output suitable for input to the SLP utility. This output contains the SLP commands and input required to make the first input file identical to the second input file. (For more information on SLP, see Chapter 17.)

CMP provides switches that allow you to control compare processing. Using these switches, you can control comparison of blanks, tabs, form-feeds, and comments. You can also control line numbering and the number of lines required for CMP to consider that a match is made between lines in the two files.

The format for specifying the CMP command line is:

```
outfile[/sw...]=infile1,infile2
```

outfile

The file specification for the output file. This file can be in one of three formats, depending on the switch you specify on the command line. The file name of this file must be specified. The default file type is .LST.

/sw...

Switches that you apply to the output file specification. Some of the switches can be negated and some are mutually exclusive. Section 16.1 contains this information.

infile1

The input file specification for the file to be compared to infile2 (described below). The file name of this file must be specified. The default file type is .MAC.

infile2

The file specification for the input file to be compared to infile1 (above). The file name for this file must be specified. The default file type is .MAC.

You can invoke CMP using any of the methods for invoking a utility described in Chapter 1.

16.1 CMP SWITCHES

This section lists the CMP switches, describes the function of each one, and gives the default setting for each one. You specify switches on the output side of the command line.

/BL
/-BL
Specifies that blank lines in both files be included in compare processing. If this switch is specified in the form /-BL, blank lines are not included in compare processing; /-BL is the default setting.

/CB
/-CB
Specifies that CMP list infile2 with change bars, in the form of exclamation marks (!), to denote which lines do not have a corresponding line in infile1. /-CB is the default switch.

You can change the change bar character from the exclamation mark to any character you wish by means of the VB switch, described below.

When a section of lines in infile1 has been deleted in infile2 (the output listing file), the first line not deleted is marked.

/CO
/-CO
Specifies that CMP include comments (that is, text preceded by a semicolon) in compare processing. /CO is the default for this switch.

/DI
/-DI
Specifies that CMP print the differences between the two files (rather than marking the lines in infile2).

/DI is the default for this switch.

/CB and /DI are mutually exclusive switches. If you specify both, /CB overrides /DI.

/FF
/-FF
Specifies that CMP include records consisting of a single form-feed character in compare processing. /-FF is the default for this switch.

/LI:n
Specifies that a number (n) of lines must be identical before CMP recognizes a match. If you do not specify this switch, CMP searches for three identical lines to match.

When it encounters a match, CMP prints all the preceding nonmatching lines, along with the first line of the matched sequence of lines to help you find the location in the code where the match occurred.

/LN
/-LN
Specifies that lines in the output file be preceded by their line number. Line numbers are incremented by one for each record read, including blank lines. /LN is the default switch. If you specify /SL (below), /LN is unnecessary.

/MB
/-MB — Specifies that CMP include all blank and tab characters in a line in compare processing. If you specify /-MB, CMP interprets any sequence of blank and/or tab characters as a single blank character. However, all spaces and tabs are printed in the output listing. /MB is the default switch.

/SL[:au] — Directs CMP to generate an output file suitable for use as SLP command input. When you specify /SL, CMP generates the SLP command input necessary to make infile1 identical to infile2. If a 1- to 8-character alphanumeric symbol is included (:au), an audit trail is specified for SLP input. Section 16.2.3, below, gives an example of how CMP generates SLP command input. (For information on how SLP processes source files, refer to Chapter 17.)

/SP[:n]
/-SP — Specifies that the output file be spooled on the line printer. You can optionally specify the number (in octal or decimal) of files to be spooled. /-SP is the default value for this switch. This switch applies only if you have the Print Spooler task (RSX-11M) or the Queue Manager system (RSX-11M/M-PLUS) installed.

/TB
/-TB — Specifies that CMP include all trailing blanks on a line in compare processing. If you specify /-TB, CMP ignores all blanks following the last nonblank character on a line. When you specify /-CO and /-TB together, blanks that precede a semicolon (;) are considered trailing blanks and are ignored. /TB is the default value for this switch.

/VB:nnn — Specifies an octal character code for use as a change bar. You use this switch with the CB switch. The value nnn specifies the octal character code. For example, you can specify /VB:174 for a vertical bar (if your printer is capable of printing the vertical bar character).

Summary of CMP Default Switch Settings

CMP default switch settings are:

/-BL	Do not compare blanks.
/-CB	Do not generate change bars.
/CO	Compare comments.
/DI	List only the differences between the two files.
/-FF	Do not compare form-feed characters.
/LI:3	Find three identical lines before a match can occur.
/LN	Generate numbered lines.
/MB	Compare all blanks and tab characters.
/-SL	Do not generate an output file suitable for use as SLP command input.
/-SP	Do not spool the output file.
/TB	Compare all trailing blanks.
/VB:041	Set the exclamation mark (ASCII 041) as the change bar character. Used with /CB.

16.2 FORMATS OF CMP OUTPUT FILES

CMP uses the two input files you specify on the command line to create an output file. CMP compares each line in infile1 to its sequential counterpart in infile2. When there are differences between the two files, CMP displays those differences in one of three output formats:

- Differences format (default) (/DI)
- Change bar format (/CB)
- SLP Command Input Format (/SL)

This section gives an example of each of these formats. In all of the examples in the subsequent sections, the following files are used as input files infile1 and infile2:

infile1	infile2
TEST1.DAT	TEST2.DAT
LINE1	LINE1
LINE2	LINE2
LINE3	LINE3
LINE4	LINE4
LINE5	LINE5
12345	45678
23456	56789
34567	67891
LINE9	LINE9
LINE10	LINE10
LINE11	LINE11
EXTRA	EXTRA
	EXTRA
	EXTRA
	EXTRA

16.2.1 Differences Format

If you enter a command line and do not specify any switches, CMP lists the differences between the two files. The differences are listed in pairs; first, the lines from infile1 that do not have counterparts in infile2 are listed, then the lines from infile2 that do not have counterparts in infile1 are listed. Each set of lines is terminated by the first line (or set of lines) for which a match is successful.

The following example shows the two lists along with the CMP command used to generate them. The input files are TEST1.DAT and TEST2.DAT, which are shown in the preceding section. There are two sets of differences separated by a long line of asterisks. (When there are several sets of differences, CMP separates each set from the next set by the long line of asterisks.) The short line of asterisks separates the pair of differences that comprise the set.

```
*****************************************
   1)   TEST1.DAT;1
     6    12345
     7    23456
     8    34567
     9    LINE9
*****************
   2)   TEST2.DAT;1
     6    45678
     7    56789
     8    67891
     9    LINE9
*****************************************
   1)   TEST1.DAT;1
*****************
   2)   TEST2.DAT;1
    13    EXTRA
    14    EXTRA
    15    EXTRA


      2 DIFFERENCES FOUND
TESTDIF.DAT=TEST1.DAT,TEST2.DAT
```

Note that because /LI:n was not specified, the number of lines required for a match defaults to 3. Thus, CMP found two differences.

16.2.2 Change Bar Format

You use the CB switch to generate a listing containing change bars that show the differences between two files. In the CMP command line, infile2 is the listing you want generated.

The following example shows change bars applied to lines from two files that do not match line for line. The command line shown at the bottom of the listing directed CMP to generate the listing.

```
     1           LINE1
     2           LINE2
     3           LINE3
     4           LINE4
     5           LINE5
     6  !        45678
     7  !        56789
     8  !        67891
     9  !        LINE9
    10           LINE10
    11           LINE11
    12           EXTRA
    13  !        EXTRA
    14  !        EXTRA
    15  !        EXTRA

      2 DIFFERENCES FOUND
TESTDIF.DAT=TEST1.DAT,TEST2.DAT/CB
```

Notice that the change bar is applied to the first line of a match (line 9, above).

16.2.3 SLP Command Input Format

Use the SL[:au] switch to generate a file containing records to be used as SLP command input. /SL directs CMP to generate the SLP edit command lines and input lines required to make infile1 identical to infile2.

However, you must enter the command line with SLP command input. CMP does not generate this command line. For a complete description of the SLP utility, refer to Chapter 17 in this manual.

The following example shows the format of output generated using the SL switch. The output file is generated with the CMP command:

```
TESTDIF.DAT/SL:BLS001=TEST1.DAT,TEST2.CMP

-6,8,/;BLS001/
45678
56789
67891
-12,,/;BLS001/
EXTRA
EXTRA
EXTRA
/
```

16.3 CMP MESSAGES

This section lists the CMP error messages, gives a brief description of the condition that causes each message, and suggests a response to the condition.

CMP -- n DIFFERENCES FOUND

Explanation: CMP found n differences between the two files.

User Action: This is an informational message.

CMP -- COMMAND SYNTAX ERROR

Explanation: CMP found an error in the command line specification.

User Action: Check the syntax of the command line specification and reenter the command line using the correct syntax.

CMP -- ERROR READING INPUT FILE

Explanation: An I/O error occurred while CMP was reading an input file.

User Action: Retry the command.

CMP -- ERROR WRITING OUTPUT FILE

Explanation: An I/O error occurred while CMP was writing the output file.

User Action: The output device may be full or bad. Check this, then retry the command.

CMP -- ILLEGAL /LI VALUE

Explanation: You specified a negative value for the number of lines required for a match.

User Action: Reenter the command line with a legal value specified.

CMP -- ILLEGAL SWITCH OR SWITCH VALUE

Explanation: An illegal switch or switch value was entered in the command line.

User Action: Reenter the command using the legal switch or switch value.

CMP -- OPEN FAILURE ON INPUT FILE #1

Explanation: CMP could not open the first input file.

User Action: Ensure that you correctly entered the file specification for the file. Reenter the command.

CMP -- OPEN FAILURE ON INPUT FILE #2

Explanation: CMP could not open the second input file.

User Action: Ensure that you correctly entered the file specification for the file. Reenter the command.

CMP -- OPEN FAILURE ON OUTPUT FILE

Explanation: CMP could not open the specified file.

User Action: Ensure that you correctly entered the file specification for the file. Reenter the command.

CMP -- TOO MANY DIFFERENCES FOR AVAILABLE CORE

Explanation: The files were too dissimilar for CMP to fit all the differences in memory.

User Action: Rerun CMP using the /INC=n switch, or remove and reinstall CMP with a larger increment. (For information on using /INC=n, see the RSX-11M/M-PLUS MCR Operations Manual.)

CHAPTER 17

SOURCE LANGUAGE INPUT PROGRAM (SLP)

The Source Language Input Program (SLP) is a utility program used for source file maintenance. SLP maintenance is usually performed on the latest version of the source file, ensuring that the file contains the most recent updates and corrections. The optional audit trail in the output files allows you to keep a record of these changes.

SLP functions are invoked by means of edit command statements and switches. SLP edit command statements allow you to:

- Update (delete, replace, add) lines in an existing file
- Create source files
- Run indirect files containing SLP edit commands

Input to SLP is an input file that you want updated and command input consisting of text lines and edit command lines that specify the update operations to be performed. To locate lines to be changed, SLP uses locators that you specify as line numbers or character strings. Command input can come directly from your terminal or from an indirect command file containing commands and input lines to be inserted into the file. SLP accepts data from any RSX-11M/M-PLUS file-structured device.

SLP output is a listing file and an updated input file. SLP provides an optional audit trail that helps you keep track of the update status of each line in the file. If an audit trail is not suppressed, it is shown in the listing and permanently applied to the output file.

You can control SLP processing using SLP control switches. These switches allow you to create or suppress audit trails, specify the length and beginning position of the audit trails, calculate the checksum value for the edit commands, generate a double-spaced listing, and spool files to a file-structured volume.

You can invoke SLP by all but one of the methods for invoking a utility described in Chapter 1. You cannot include a command line on the same line on which you invoke SLP. That is, you cannot type SLP filespec.

Also, after you invoke SLP, specifying TI: as your output file specification is not recommended because SLP displays the input file while you are editing it. When you are done editing, you do not have a copy of the output file and the input file is the same as before you began editing.

17.1 SLP INPUT AND OUTPUT FILES

SLP requires two types of input, an input file and command input. The input file is the source file you want to update using SLP. Command input consists of SLP edit commands and, optionally, new lines of text to be placed in the file.

SLP output consists of an output file and a listing file. The output file is the updated input file. The listing file is a copy of the output file with line numbers added. Both show the changes SLP makes to the file.

17.1.1 The Input File

The input file is the file to be updated by SLP. It can contain as many lines of text as are required. When SLP processes the input file, it makes the changes specified by SLP edit commands. If an audit trail is generated, it notes these changes in the output files.

17.1.2 Command Input

SLP uses command input to update files. Command input can be entered interactively after you invoke the SLP utility or indirectly by means of indirect command files.

You enter command input to SLP in two modes: command mode and edit mode. After it is invoked, SLP is in command mode, in which it expects the first line entered to be the command line defining the files to be processed. When SLP accepts this line, it initializes the files you want processed. Once these files are initialized, SLP enters edit mode, in which it interprets the lines you enter as SLP edit commands or new input lines.

You terminate command input with a single slash as the first character of an edit command line.

The following example shows the general form of command input:

```
MYFILE.MAC;2/CS/AU:55:10,MYFILE.LST;1/-SP=MYFILE.MAC;1
-3,,/;BJ007/
INSERT THIS LINE AFTER LINE 3
-4,4
DELETE LINE 4 AND REPLACE IT WITH THIS LINE
/
```

NOTE

Numeric values given for switches default to octal. Decimal values must be followed by a period (.). The default position for the audit trail is 80(10) and its default length is 14(10) characters. (See Section 17.4.2 for more information about the audit trail.)

The first line is the command line, on which SLP expects to find definitions for the output file, the listing file, and the input file. The next four lines comprise the SLP edit commands and input lines.

Note that the input and output files in the example above have the same file name and file type; only the versions are different. To ensure that the correct files are processed, specify the version numbers explicitly when you enter the SLP utility command line. Wildcards cannot be used in any of the file specifications.

You can also have the checksum value for the edit commands calculated. Specify the checksum switch with either the input or output file specification in the format:

```
/CS[:n]
```

The checksum value is calculated for all SLP edit command lines, excluding:

- The command line specifying the input and output files
- Comments in the edit command lines
- Any spaces and/or tabs between characters included in the checksum calculation and those characters excluded from the calculation
- The second comma and anything following it in an edit command line (that is, audit trail and/or comment)
- Comment delimiter (specified by first character of the last audit trail string before the delimiter) and any characters following it in an input line, whether or not it is being used in the line as a delimiter

The value is then reported in a message on your terminal. If you specify a value for the checksum and it is not the same as the calculated checksum, you will get a diagnostic error message. (The messages are described below in Section 17.5.)

17.1.3 The SLP Listing File

The SLP listing file shows the updates made to the source file. Each line in the listing file is numbered. Updates are marked by means of the audit trail if one has been generated. The examples given throughout the chapter contain samples of listing files.

17.1.4 The SLP Output File

The SLP output file is the updated input file. All of the updates specified by means of command input are inserted in this file. The audit trail, if specified, is applied to lines changed by the update. The audit trail is included in the output file. The numbers generated by SLP for the listing file do not appear in the output file.

17.2 HOW SLP PROCESSES FILES

Figure 17-1 shows how SLP processes files when it receives the following command line and edit commands:

```
MYFILE.MAC;2/AU:55:10,MYFILE.LST/-SP=MYFILE.MAC;1
-3
INSERT THIS LINE AFTER LINE 3
-4,4
DELETE LINE 4 AND REPLACE IT WITH THIS LINE
/
```

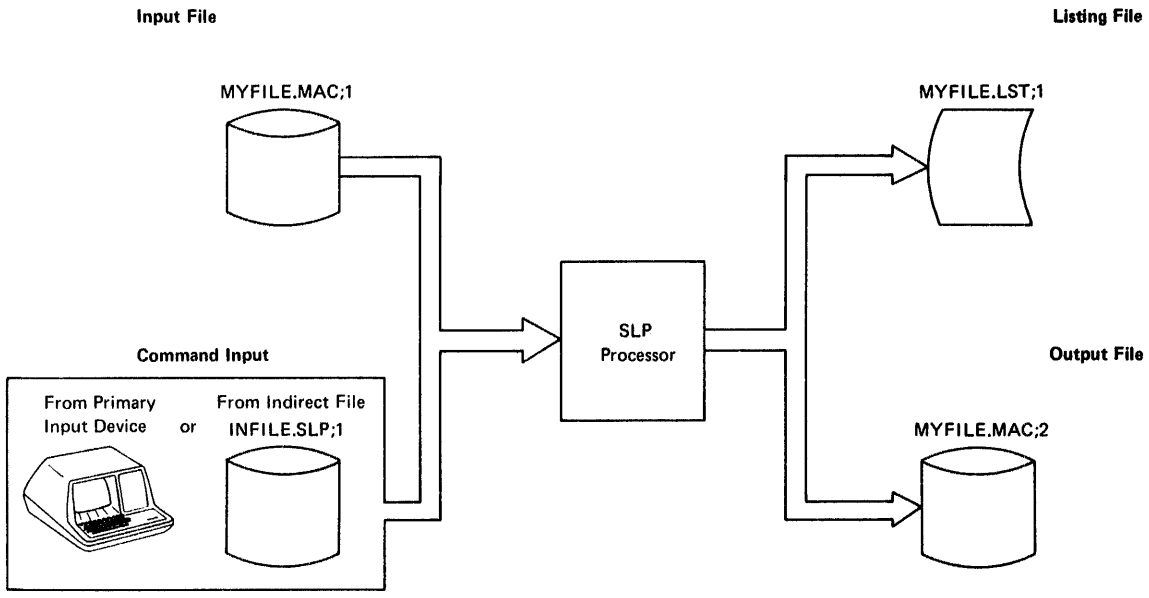


Figure 17-1 Input Files and Output Files Used During SLP Processing

This is the input file (MYFILE.MAC;1) before SLP processes the files:

```
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE
TEN
```

The following is the listing file (MYFILE.LST;1) resulting from SLP processing of these files:

```
 1.  ONE
 2.  TWO
 3.  THREE
 4.  INSERT THIS LINE AFTER LINE 3                    ;**NEW**
 5.  DELETE LINE 4 AND REPLACE IT WITH THIS LINE      ;**NEW**
 6.  FIVE                                             ;**-1
 7.  SIX
 8.  SEVEN
 9.  EIGHT
10.  NINE
11.  TEN
```

The audit trail shows the new lines (;**NEW**) and indicates where lines have been removed (;**-1). (The audit trails ;**NEW** and ;**-n are automatically generated by SLP if you have not suppressed audit trail generation or if you have not specified another audit trail string.) In this case, a line has been added after line 3, and line 4 has been deleted and a new line added in its place.

As shown in Figure 17-1, SLP processes an input file using command input. When processing begins, SLP writes each line from the input file into the output file until it reaches a line to be modified, as requested in the command input. When SLP reaches a line to be modified, it makes the indicated modification, notes the change by means of the audit trail if the trail has been generated, and then continues writing lines to the output file, in sequence, until another command is encountered or until end of file is reached.

17.3 USING SLP

This section describes how to:

- Specify the SLP edit command
- Update files using the SLP edit command
- Enter SLP commands interactively and by means of indirect files
- Create a source file using SLP

17.3.1 Specifying SLP Edit Commands

The SLP edit commands provide the means to update source files by adding, deleting, and replacing lines in a file. You enter SLP edit commands after invoking SLP and specifying an edit command line.

The general format of the SLP edit command line is as follows:

```
-[locator1][,locator2][,/audittrail/][; comment]
inputline
    .
    .
```

- (dash)

Specifies that this is a SLP edit command line.

locator1

A line locator that causes SLP to move the current line pointer to a specified line. If you specify only locator1, the current line pointer is moved to that line and SLP reads the next line in the command input file. This field can be specified using any of the locator forms described below.

locator2

A line locator that defines a range of lines (that is, the range beginning with locator1 and ending with locator2, inclusive) to be deleted or replaced. This field can be specified using any of the locator forms described below.

/audittrail/

A character string used to keep track of the update status of each line in the file. The string must be enclosed within slashes (/). It consists of a comment delimiter as the first character and then a text string. The semicolon (;) is the initial delimiter for audit trails automatically generated by SLP (;**NEW** and ;**-n). The comment delimiter specified in audittrail (usually a semicolon) is the new delimiter for all subsequent audit trails until redefined by a later audittrail.

inputline

A line of new text to be inserted into the file immediately following the current line. You can enter as many input lines as required.

comment

A line of text (delimited by a semicolon) at the end of the SLP edit command line that appears only in the command input file.

All fields in the SLP edit command line are positional and commas must be specified.

The locator fields can take one of the following forms:

```
-/string/[+n]
-/string...string/[+n]
-number[+n]
-.[+n]
```

string

A string of ASCII characters. SLP locates the line in which string exists and moves the current line pointer to that line. If the locator is specified in the form /string...string/, SLP locates the line in which the two character strings delimit a larger character string abbreviated by an ellipsis (...).

number

A decimal line number to which the current line pointer is to be moved. The largest line number that can be specified is 9999.

n

A decimal value used as an offset from the line specified by the locator. You cannot use +n by itself. It must be specified with a number or string locator, or a period. SLP moves the current line pointer n lines beyond the line specified in the locator field.

.(period)

The current line.

Although the values for number and n are taken as decimal, remember that all other SLP values are octal by default.

All forms of the line locator can be specified interchangeably in the SLP edit command line.

17.3.2 Entering SLP Edit Commands

Once you have invoked SLP, you can enter SLP edit commands interactively or by means of indirect command files.

In both cases, the first command you must enter is the command line defining the files to be processed during this SLP session.

This section gives examples of how to use both methods of entering SLP commands.

The following file, BASE.MAC;1, is used as the input file for the examples in this section:

```
AAAA
2222
BBBB
4444
CCCC
6666
DDDD
8888
9999
 TEN
```

17.3.2.1 **Entering SLP Commands Interactively** - To alter the example file interactively, first invoke SLP:

```
>SLP
```

SLP responds to this command with the prompt:

```
SLP>
```

Once you have entered the SLP command mode, SLP does not display prompts. The first line you enter must always be the command line to tell SLP the files you want processed during this session:

```
BASE.MAC;2/AU:48./TR,BASE.LST=BASE.MAC;1
```

Then you enter the edit commands and input lines:

```
-3
INSERT THIS LINE AFTER LINE 3
-4,4
DELETE LINE 4 AND REPLACE IT WITH THIS LINE
-6,,/;JM010/
INSERT THIS LINE AFTER LINE 6 WITH A NEW AUDIT TRAIL VALUE
/
```

Enter only the SLP edit commands and the new input lines. When you have entered all the corrections, enter the slash (/) to terminate the edit session. SLP processes the files and returns control to you with the prompt:

```
SLP>
```

This returns SLP to command mode. You can then enter the command line and begin another editing session.

The listing file resulting from SLP processing appears as follows:

```
 1.  AAAA
 2.  2222
 3.  BBBB
 4.  INSERT THIS LINE AFTER LINE 3                    ;**NEW**
 5.  DELETE LINE 4 AND REPLACE IT WITH THIS LINE      ;**NEW**
 6.  CCCC                                             ;**-1
 7.  6666
 8?  INSERT THIS LINE AFTER LINE 6 WITH A NEW AUDIT T;JM010
 9.  DDDD
10.  8888
11.  9999
12.  TEN
```

The TR switch (/TR in the command line) records the truncation of lines by the audit trail. In the listing file, a question mark (?) replaces the period (.) in the line number of the lines that were truncated. It is possible that audit-trail strings in the input file will be truncated by the new audit-trail string although the commands or text strings will not be truncated.

17.3.2.2 **Entering SLP Commands Using Indirect Command Files** - You can enter the same commands using an indirect command file.

In this example, the SLP edit commands are contained in the indirect command file, BASE.SLP. You invoke SLP and SLP responds with the prompt:

```
SLP>
```

You then enter the file specification for the indirect command file containing the command line, the SLP edit commands, and the input lines:

```
@BASE.SLP
```

SLP processes the files just as if you entered the commands and input lines interactively, returning control to you with the prompt:

```
SLP>
```

You can also specify SLP @BASE.SLP.

The output listing resulting from indirect command file processing is exactly like the output listing resulting from the same changes made interactively.

Indirect command files can be nested to a maximum level of 3. This permits indirect command files to reference a standard editing file when SLP is invoked with an indirect command file reference at the monitor level.

17.3.2.3 **Using SLP Operators** - In addition, you can enter certain special characters called operators, which perform specific functions. Table 17-1 lists the operators and the function each performs. Enter operators, in edit mode, as the first character of an input line.

Table 17-1
SLP Operators

Operator	Function
-	SLP interprets the dash as the first character of a SLP edit command.
\	SLP interprets the backslash as a command to suppress audit-trail processing.
%	SLP interprets the percent sign as a command to reenable audit-trail processing.
@	SLP interprets the at sign as a command to invoke an indirect file for SLP processing.
/	SLP interprets the slash as a command to terminate the SLP edit session, and then return to SLP command mode.
<	SLP interprets the less-than character as an escape character. Escape characters are characters that enable you to enter characters in the input file that SLP otherwise would interpret as operators. For example, </ hides the slash character from SLP, thereby enabling you to enter the slash into the output file without terminating the SLP editing session. The less-than character can be used with all SLP operators.

SLP allows you to enter lines sequentially. Once past a given line in the file, you cannot return the line pointer to that line. To return the line pointer to that line, you must begin another SLP editing session.

To close the file and terminate the editing session, use the slash (/) operator.

17.3.3 Updating Source Files Using SLP

This section describes the procedure for generating a numbered listing for use in editing source files by line number. The section also describes how to use SLP to add, delete, and replace lines in a file.

17.3.3.1 **Generating a Numbered Listing** - SLP processes input by line number. However, line numbers appear only in the listing file; they are not written to the output file.

To use SLP effectively, you should use a numbered listing when you prepare command input. To generate a numbered listing, first invoke SLP, then enter the command line in the format:

```
,listfile=infile
/
```

In this format, listfile is the name you assign to the listing file SLP produces and infile is the name of the input file whose lines are to be numbered. The slash (/) terminates the edit mode. For example, suppose the input file is:

```
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE
TEN
```

SLP processes each line to generate a numbered listing file:

```
 1.  ONE
 2.  TWO
 3.  THREE
 4.  FOUR
 5.  FIVE
 6.  SIX
 7.  SEVEN
 8.  EIGHT
 9.  NINE
10.  TEN
```

17.3.3.2 **Adding Lines to a File** - The three SLP edit command formats for adding lines to a file are:

```
-locator1
inputline
    .
    .
    .

    or

-locator1,,
inputline
    .
    .
    .

    or

locator1,,/audittrail/
inputline
    .
    .
    .
```

See above for the descriptions of these terms.

The following example shows how to add lines to a file. The input file consists of the following lines:

```
ABC
DEF
GHI
KLM
123456789
456
789
CBA
XYZ
987
```

The command input consists of the following commands and text lines:

```
MYFILE.MAC;2/AU:50,LISTING=MYFILE.MAC;1
-/123/
INSERT THIS LINE AFTER LINE 5
/
```

SLP processing generates the following listing file:

```
 1.  ABC
 2.  DEF
 3.  GHI
 4.  KLM
 5.  123456789
 6.  INSERT THIS LINE AFTER LINE 5          ;**NEW**
 7.  456
 8.  789
 9.  CBA
10.  XYZ
11.  987
```

SLP has numbered the lines and applied an audit trail to the line following line 5, where SLP found the first occurrence of the string 123.

The next example uses the same input file and the following new command lines:

```
MYFILE.MAC;2/AU:50,LISTING=MYFILE.MAC;1
-/DEF/+2
THIS IS NEW TEXT
/
```

SLP processing generates the following listing file:

```
 1.  ABC
 2.  DEF
 3.  GHI
 4.  KLM
 5.  THIS IS NEW TEXT                          ;**NEW**
 6.  123456789
 7.  456
 8.  789
 9.  CBA
10.  XYZ
11.  987
```

Again, SLP has numbered the lines and this time the new input line is inserted two lines beyond the line containing the first occurrence of the string DEF.

17.3.3.3 **Deleting Lines from a File** - The SLP edit command format for deleting lines from a file is:

```
-[locator1],[locator2],[/audittrail/][;  comment]
```

In this format, locator1 and locator2 can be any of the forms of the locator fields described in Section 17.3.1; locator1 specifies the line where SLP is to begin deleting lines; locator2 specifies the last line to be deleted. SLP deletes all lines from locator1 through locator2, inclusive.

The following example shows how to delete lines from a file using SLP. The input file consists of the following lines:

```
ABC
DEF
GHI
KLM
123456789
456
789
CBA
XYZ
987
```

The command input for this example is:

```
MYFILE.MAC;2/AU:50,LISTING=MYFILE.MAC;1
-/123...789/,/XYZ/
/
```

SLP processing generates the following listing file:

```
1.  ABC
2.  DEF
3.  GHI
4.  KLM
5.  987                                          ;**-5
```

In this example, the ellipsis (...) abbreviates the larger string 123456789. Assuming the two strings bracket a larger string, SLP searches for the first occurrence of the string 123 and then the first occurrence on the same line of the string 789, in this case the string 123456789. SLP begins deleting lines at this line and continues deleting lines until it deletes the last line of the given range, specified here by the string XYZ. SLP applies the audit-trail count of the lines it deleted to the next line from the input file.

Using the same input file as used in the example above, this example shows how to delete a single line using the period locator. The command input for this example is:

```
MYFILE.MAC;2/AU:50,LISTING=MYFILE.MAC;1
-/DEF/,.
/
```

SLP processing generates the following listing:

```
1.  ABC
2.  GHI                                          ;**-1
3.  KLM
4.  123456789
5.  456
6.  789
7.  CBA
8.  XYZ
9.  987
```

SLP moves the current line pointer to the line containing the string DEF and then finds the period as the second locator field. Since the second locator field is specified as the current line, SLP deletes the current line.

17.3.3.4 **Replacing Lines in a File** - A replacement is the deletion of old text followed by the insertion of new text. The number of lines deleted need not match the number of lines added. To replace lines in a file, use the same SLP edit command format as used in the delete command. The first line locator field specifies the first line to be deleted. The second line locator field defines the last line in the range to be deleted and where the new text is to be inserted. For example:

```
-4,.+4
```

This command instructs SLP to move the line pointer to line 4, and replace line 4 and the next four lines with new input lines.

The following example shows how to delete lines from a file and replace them with new lines. The input file consists of the following lines:

```
ABC
DEF
GHI
123456789
BCN
CRB
BUR
```

The command input is:

```
MYFILE.MAC;2/AU:50,LISTING=MYFILE.MAC;1
-2,.+1
NEW LINE 2
NEW LINE 3
/
```

SLP processing generates the following listing file:

```
1.  ABC
2.  NEW LINE 2                                  ;**NEW**
3.  NEW LINE 3                                  ;**NEW**
4.  123456789                                   ;**-2
5.  BCN
6.  CRB
7.  BUR
```

17.3.4 Creating Source Files Using SLP

Using SLP to create source files is possible, but not recommended. SLP does not have an intraline editing mode, and you cannot return to a line once you have passed it. The interactive editors, EDI and EDT, are better for creating source files.

To create source files using SLP, invoke SLP and enter the command line in the format:

```
outfile/-AU[/sw][,listfile][/sw]=primary input device:[/sw]
```

outfile

The file specification for the output file. The default device is SY0:.

/-AU

Specifies that an audit trail is not to be generated. Otherwise, you will get the ;**NEW** audit trail on every line of the output files.

listfile

The file specification for the listing file (optional).

primary input device:

Specifies that input for the file being created is coming from this device, for example, a terminal or a card reader.

/sw

Specifies any optional SLP switches you want set.

The default device for output files is SY0:; for input files, the default device is your primary input device. For listing files, the default device is implied by the output file specification. The following file specification creates a new file called MYFILE.MAC from the terminal and puts it on SY0:.

```
MYFILE.MAC/-AU=TI:
```

Once you have entered the file specification, SLP accepts each line as a variable-length record of up to 132(10) characters. Trailing blanks and tabs on input lines are deleted. SLP expects input to the file to come from the primary input device. End the SLP session with a slash (/) and then a CTRL/Z.

17.4 CONTROLLING SLP

The SLP switches allow you to calculate the checksum value for the edit commands and to control the generation and format of the listing file and the output file.

17.4.1 SLP Switches

The procedure SLP uses to calculate the checksum value for the edit commands is described above in Section 17.1.2.

SLP output consists of two files -- a listing file and the output file, which is the modified version of the original input file. You can control the audit trail and print options associated with the two files using the SLP switches.

The effects of SLP switches are the same whether you apply them to input or output files (except for the SP switch, which you can specify only with the listing file). Table 17-2 lists the SLP switches and gives a brief description of the functions each performs.

Table 17-2
SLP Switches

Switch	Function
/AU /-AU	Allows you to generate an audit trail or suppress audit trail generation, and specify the beginning field and length of the audit trail. /AU is the default value.
/BF /-BF	Right-justifies the audit trail by inserting spaces instead of tabs at the end of text information. /BF is the default value.
/CM[:n]	Deletes audit trails and any trailing spaces or tabs, and truncates the text at a specified horizontal position. The value given for the beginning position of the audit trail is the default value for this switch.
/CS[:n]	Calculates the checksum value for the edit commands. If you do not specify n, SLP reports the value in a message on your terminal. If you do specify n and the checksum value that SLP calculates is not the same as the one you specified, SLP displays a diagnostic error message.
/DB /-DB	Generates the listing file in double-space format. /-DB is the default value.
/SP /-SP	Spools the listing file to the printer. /SP is the default value. This switch applies only if you have the print spooler task (RSX-11M) or the queue management system (RSX-11M/M-PLUS) on your system.
/TR	Reports truncation of lines by the audit trail. If line truncation occurs, you will get a diagnostic error message. There is no default value for this switch. In the listing file, a question mark (?) replaces the period (.) in the line number of the lines that were truncated.

17.4.2 Controlling the Audit Trail

The [-]AU switch allows you to generate, suppress, and set the length and contents of the audit trail.

To suppress generation of the audit trail, specify the /-AU switch in either the input or output file specification.

For example, either command line below generates an output file with no audit trail:

```
DK1:MYFILE.MAC;3/-AU,LP.LST:=MYFILE.MAC;2
DK1:MYFILE.MAC;3,LP.LST:=MYFILE.MAC;2/-AU
```

By default, SLP automatically generates an audit trail; that is, you need not explicitly specify /AU in your command line (unless you want to specify the beginning position and length of the audit trail).

LP.LST will be spooled automatically.

17.4.3 Setting the Position and Length of the Audit Trail

You can set the beginning position of the audit trail and the length of the audit trail using the AU switch in the format:

```
/AU:position:length
```

position

A number, less than or equal to 132(10), designating the beginning character position of the audit trail on the line. SLP rounds this value to the next highest tab stop (a multiple of 8). The default value for position is 80(10).

length

The length of the audit trail. The default value for length is 8(10) characters; no more than 14(10) characters may be specified.

The example below shows how to specify the beginning position and length of the audit trail. The input file for this example is:

```
ONE
TWO
THREE
FOUR
FIVE
```

The command input is:

```
MYFILE.MAC;2/AU:30.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1
-2,.+1,/;CHANGE001/
NEW LINE 2
NEW LINE 3
/
```

The listing file resulting from SLP processing is:

```
1.  ONE
2.  NEW LINE 2                    ;CHANGE001
3.  NEW LINE 3                    ;CHANGE001
4.  FOUR                          ;**-2
5.  FIVE
```

17.4.4 Changing the Value of the Audit Trail

To change the value of the audit trail, specify:

```
-[locator1],[locator2],/;new value/
```

Audit trail values can change whenever and wherever you want them to. For example,

```
MYFILE.MAC;2/AU:48.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1
-3
INSERT THIS LINE AFTER LINE 3
-4,4
DELETE LINE 4 AND REPLACE IT WITH THIS LINE
-6,,/;JM010/
INSERT THIS LINE AFTER LINE 6 WITH A NEW AUDIT TRAIL VALUE
-9,,/;BJ008/
INSERT THIS LINE AFTER LINE 9 WITH ANOTHER NEW AUDIT TRAIL VALUE
/
```

The listing file resulting from SLP processing appears as follows:

```
 1.  AAAA
 2.  2222
 3.  BBBB
 4.  INSERT THIS LINE AFTER LINE 3                    ;**NEW**
 5.  DELETE LINE 4 AND REPLACE IT WITH THIS LINE      ;**NEW**
 6.  CCCC                                             ;**-1
 7.  6666
 8?  INSERT THIS LINE AFTER LINE 6 WITH A NEW AUDIT T;JM010
 9.  DDDD
10.  8888
11.  9999
12?  INSERT THIS LINE AFTER LINE 9 WITH ANOTHER NEW A;BJ008
13.  TEN
```

17.4.5 Temporarily Suppressing the Audit Trail

You can temporarily suppress the generation of the audit trail by using the backslash (\) operator. You can then reenable audit-trail processing with the percent sign (%) operator. (You cannot enable audit trail processing with this operator if you have specified /-AU in the SLP command line.)

Both operators are entered in the command input. The backslash (\) is specified in column 1 of the line that precedes those commands and/or input files for which you do not want audit-trail processing. The percent sign (%) is specified in column 1 of the line that precedes the lines for which you do want processing. For example:

```
BAK.MAC;26/AU/-BF=BAK.MAC;25
\
-2,2
          .IDENT  /05.03/
-23,23
; VERSION 05.03
-37,,
;  J. MATTHEWS      11-JAN-79
;
;       JM011      CORRECT OUT-OF-BOUNDS CONDITION FOR INPUT-BUFFER
;                  SIZE
;
%
-106,106,/;JM011/
   CMP     #132.,R3          ; IS INPUT-BUFFER SIZE IN RANGE?
   BLT     30$               ; IF LT, NO
   .
   .
   .
/
```

The lines between the backslash (\) and the percent sign (%) are not affected by audit-trail processing. The lines following the percent sign (%) are affected.

17.4.6 Deleting the Audit Trail

The CM switch allows you to delete audit trails and trailing spaces and tabs from a file. The CM switch applied to the output or input file specification accepts a numeric argument that specifies the beginning position of an audit trail or other text string to be deleted. The default for this argument is the position argument given for the AU switch (or its default, decimal 80). This value is rounded to the next highest tab stop before use.

When processing an input line, SLP first truncates the text to the next highest tab stop after the position specified, and then deletes any trailing spaces or tabs. The remaining text is copied to the output file.

The CM switch is specified in the form:

```
/CM:[position]
```

position

A number designating the beginning character position of the audit trail (or other text) to be deleted.

For example:

```
SLP>SLPR11.MAC;12/CM:119.=SLPR11.MAC;11
/
```

In this case, the input lines are truncated to a length of 120(10) characters. The specified length is rounded up to the next highest tab stop (decimal 120) and the audit trail begins at column 121(10). Trailing spaces and tabs are deleted before each line is copied to the output file.

In the next example, SLP truncates input lines to the default position of the audit trail, column 80(10).

```
SLP>SLPR11.MAC;12=SLPR11.MAC;11/CM
/
```

17.5 SLP MESSAGES

SLP messages are divided into two groups: information and error. The messages and suggested responses are given below. Section 17.5.1 describes the information message and Section 17.5.2 describes the error messages.

17.5.1 SLP Information Message

SLP -- COMMAND FILE CHECKSUM IS ######

Explanation: By specifying /CS in the command line, you requested SLP to calculate the checksum value for the edit commands.

User Action: This message is for your information only. No action is required.

17.5.2 SLP Error Messages

SLP error messages are issued in two formats:

- SLP followed by two dashes, the type of error message, and the error message. If applicable, the command line or command line segment that caused the message is printed on the next line. For example:

  ```
  SLP -- *FATAL*-ILLEGAL SWITCH
  SHIRLEY.MAC;2
  ```

- SLP followed by two dashes, the type of error message, the error message, and the name of the file with which the error is associated. For example:

  ```
  SLP -- *FATAL*-OPEN FAILURE LINE LISTING FILE filename
  ```

This section lists the error messages SLP can issue. Following each message is an explanation of the error and recommended user action to correct the error. Note that all but two of the SLP error messages are fatal. The two exceptions are diagnostic messages, which are described at the end of this section.

SLP -- *FATAL*-COMMAND SYNTAX ERROR
command line

Explanation: The command line format did not conform to syntax rules. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line.

SLP -- *FATAL*-ILLEGAL DEVICE NAME
command line

Explanation: The device specified was not a legal device. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line.

SLP -- *FATAL*-ILLEGAL DIRECTORY
command line segment

Explanation: The directory was not legally specified. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line.

SLP -- *FATAL*-ILLEGAL ERROR/SEVERITY CODE p1 p2 p3

Explanation: This error message indicates an error in the SLP program.

User Action: Reenter the command line. If the error persists, submit a Software Performance Report along with the related console dialogue and any other related information, such as programs or listings.

SLP -- *FATAL*-ILLEGAL FILE NAME
command line segment

Explanation: A file specification was greater than 30(8) characters in length or contained a wildcard (that is, an asterisk in place of a file specification element). Open files were closed and SLP was reinitialized.

User Action: Reenter the command line.

SLP -- *FATAL*-ILLEGAL GET COMMAND LINE ERROR

Explanation: The system was unable to read a command line. This indicates an internal system failure or an error in the SLP program.

User Action: Reenter the command line. If the error persists, submit a Software Performance Report along with the related console dialogue and any other pertinent information.

SLP -- *FATAL*-ILLEGAL SWITCH
command line segment

Explanation: The switch was not a valid SLP switch or a legal switch was used in an invalid manner. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line with the correct switch specified.

SLP -- *FATAL*-INDIRECT COMMAND SYNTAX ERROR
command line

Explanation: The command line format specified for the indirect command file did not conform to syntax rules. Open files are closed and SLP was reinitialized.

User Action: Reenter the command line.

SLP -- *FATAL*-INDIRECT FILE DEPTH EXCEEDED
command line

Explanation: More than three levels of indirection were specified in an indirect command file. Open files were closed and SLP was reinitialized.

User Action: Correct the command file and reenter the command line.

SLP -- *FATAL*-I/O ERROR COMMAND INPUT FILE

or

SLP -- *FATAL*-I/O ERROR COMMAND OUTPUT FILE

or

SLP -- *FATAL*-I/O ERROR CORRECTION INPUT FILE filename

or

SLP -- *FATAL*-I/O ERROR LINE LISTING FILE filename

or

SLP -- *FATAL*-I/O ERROR SOURCE OUTPUT FILE filename

Explanation: One of the following conditions may exist:

- A problem exists on the physical device (for example, the disk is not spinning).
- The length of the command line was greater than the specified number of characters.
- The file is corrupted or the format is incorrect.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

SLP -- *FATAL*-INDIRECT FILE OPEN FAILURE
command line

or

SLP -- *FATAL*-OPEN FAILURE CORRECTION INPUT FILE filename

or

SLP -- *FATAL*-OPEN FAILURE LINE LISTING FILE filename

or

SLP -- *FATAL*-OPEN FAILURE SOURCE OUTPUT FILE filename

Explanation: One of the following conditions may exist:

- The file is protected against an access.
- A problem exists with the physical device (for example, the device was not online).
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.
- The available Executive dynamic memory is insufficient for the operation.

These errors cause open files to be closed and SLP to be reinitialized.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

SLP -- *FATAL*-LINE NUMBER ERROR
command line

Explanation: The command line printed contained an illegally-specified numeric line locator.

User Action: Terminate the SLP edit session and refer to the rules for specifying numeric line locators in Section 17.3.1. Correct the error and reenter the command line.

SLP -- *FATAL*-PREMATURE EOF CORRECTION INPUT FILE filename

Explanation: An out-of-range line locator was specified in a correction file or from the terminal; for example, -1000 was specified for an 800-line file.

User Action:

- Terminate the current editing session.
- Restart the editing session, entering the correct line number.

SLP -- *FATAL*-PREMATURE EOF COMMAND INPUT FILE

Explanation: This is caused by not terminating SLP command input with a slash (/) or by inadvertently typing CTRL/Z at the terminal, which sends an end-of-file to SLP before the slash (/) character is read. SLP prints SLP>, indicating that a new file specification is expected.

User Action: Restart the editing session at the point where the CTRL/Z was typed.

SLP -- *DIAG*-ERROR IN COMMAND FILE filespec CHECKSUM

Explanation: An incorrect value was specified for the command file checksum. If you enter the edit command lines directly from the terminal, the command file in the error message is CMI.CMD. Thus, the error message reads:

SLP -- *DIAG* - ERROR IN COMMAND FILE CMI.CMD CHECKSUM

User Action: This is a warning message only. The specified output file is still created, although possibly not as intended.

SLP -- *DIAG*-n LINES TRUNCATED BY AUDIT TRAIL
command line

Explanation: Line truncation by the audit trail was detected.

User Action: This is an informational message only. The specified output file is still created. (In the listing file, a question mark (?) replaces the period (.) in the line number of the lines that were truncated. It is possible that audit-trail strings from the input file will be truncated by the new audit-trail string although text strings will not be truncated.) Determine where the truncation(s) occurred. If necessary, modify the command file so that it contains commands that do not cause truncation.

CHAPTER 18

OBJECT MODULE PATCH UTILITY (PAT)

The Object Module Patch Utility (PAT) allows you to update, or patch, code in a relocatable binary object module.

Input to PAT is two files, an input file and a correction file. The input file consists of one or more concatenated object modules. You can correct only one of these object modules with a single execution of PAT. The correction file consists of object code that, when linked by the Task Builder, either overlays or is appended to the input object module. Unlike the Task Builder and ZAP patching options, PAT allows you to increase the size of the object module because the changes are applied before the module is linked by the Task Builder.

PAT uses the correction file, which contains corrections and/or additional instructions, to update the object module. Correction input is prepared in source form and then assembled by the MACRO-11 assembler.

Output from PAT is the updated input file.

You invoke PAT using any of the methods for invoking a utility described in Chapter 1. PAT can be used interactively or by means of indirect command files. If you use indirect command files, PAT allows a maximum nesting level of 2.

Using PAT to update a file involves several steps. First, you create the correction file using a text editor. Once created, the correction file must be assembled to produce an object module. The correction file and the input file (both in object module format) are then submitted to PAT for processing. Finally, the updated input object module is submitted to the Task Builder to resolve global symbols and to create an executable task. Figure 18-1 shows the processing steps involved in generating an updated task file using PAT.

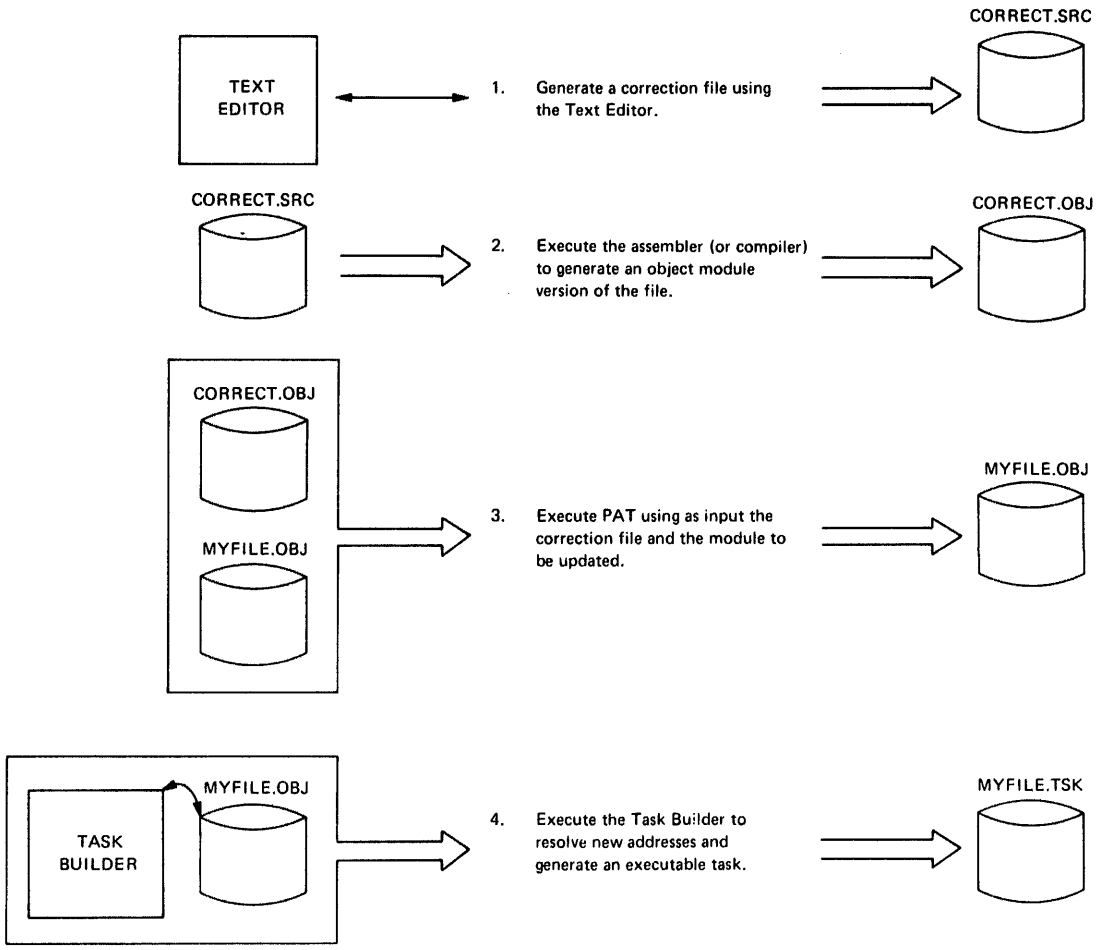


Figure 18-1 Processing Steps Required to Update a Module Using PAT

18.1 SPECIFYING THE PAT COMMAND STRING

Specify the PAT command string in the following format:

```
[outfile]=infile[/CS[:number]],correctfile[/CS:[number]]
```

outfile

The file specification for the output file. If you do not specify an output file, PAT does not generate one.

infile

The file specification for the input file. This file can contain one or more concatenated object modules.

correctfile

The file specification for the correction file. This file contains the updates being applied to one module in the input file.

/CS[:number]

Specifies the Checksum switch. This switch directs PAT to calculate the checksum for all the binary data that constitutes the module. PAT displays this checksum in octal. (Refer to Section 18.2.4 for information on how to use /CS.) You can optionally specify an octal number with /CS. Then, after PAT calculates the checksum value, it compares that value with the number you specified. If the values are not the same, PAT informs you with an error message.

18.2 HOW PAT APPLIES UPDATES

This section describes the PAT input and correction files, gives information on how to create the correction file, and gives examples of how PAT applies the corrections to a module.

18.2.1 The Input File

The input file is the file to be updated; it is the base for the output file. The input file must be in object module format. When PAT executes, the correction file is applied to one of the object modules in the file. PAT assumes a file type of .OBJ for the input file. If you use a file type other than .OBJ, you must specify it explicitly in the command line.

18.2.2 The Correction File

The correction file contains the patches to be applied to the input file. Like the input file, the correction file has a default file type of .OBJ. If you use any other file type. you must specify it explicitly in the command line.

As shown above in Figure 18-1, the first step in using PAT to update an object file is to generate the correction file. Use any text editor to create this source file, which is usually in the following format:

```
.TITLE inputname
.IDENT updatenum
inputline
inputline
    .
    .
    .
```

inputname

The name of the module to be corrected by the PAT update. You must specify inputname to be the same as the name of the module that you are updating.

updatenum

Any value acceptable to the MACRO-11 .IDENT assembler directive. Generally, this value reflects the updated version of the file being processed by PAT (as shown in the examples given in Section 18.2.3).

NOTE

The .IDENT assembler directive is a required part of the correction file. Failure to include a .IDENT directive in the file produces unusable output.

inputline

Lines of input to be used to correct and update the input file.

Once you have created the source version of the correction file, you assemble it to produce an object module that can be processed by PAT.

During PAT execution, new global symbols defined in the correction file are added to the module's symbol table. A symbol definition that is already being used in the input file can be superseded by the definition in the correction file. For a symbol definition to be superseded, both definitions must be either relocatable or absolute.

A duplicate program section supersedes the previous program section, provided:

- Both have the same relocatability attribute (ABS or REL)
- Both are defined with the same directive (.PSECT or .CSECT)

If PAT encounters duplicate program section names, the length attribute for the program section is set to the length of the longer program section, and a new program section is appended to the module.

If you specify a transfer address, it supersedes the transfer address of the module being patched.

18.2.3 How PAT and the Task Builder Update Object Modules

The examples in the following sections show an input file and a correction file (both in object module format) to be processed by PAT and the Task Builder, along with a source-like representation of how the output file looks once PAT and the Task Builder complete processing. Two techniques are described, one for overlaying lines in a module and the other for adding a subroutine to a module.

18.2.3.1 **Overlaying Lines in a Module** - The first example illustrates a technique for overlaying lines in a module using a patch file. First, PAT appends the correction file to the input file. Then, the Task Builder generates a task image from the patched object modules.

The input file for this example is:

```
        .TITLE  ABC
        .IDENT  /01/
ABC::
        MOV     A,C
        CALL    XYZ
        RETURN
        .END
```

To add the instruction ADD A,B after the CALL instruction, you can use the following patch in the correction file:

```
        .TITLE  ABC
        .IDENT /01.01/
.=.+12
        ADD     A,B
        RETURN
        .END
```

You use the MACRO-11 assembler to assemble the correction file. After assembly, PAT processes the resulting object module and the input object module. The result of PAT processing appears as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        MOV     A,C
        CALL    XYZ
        RETURN
.=ABC
.=.+12
        ADD     A,B
        RETURN
        .END
```

You then use the Task Builder to produce the patched object module as a task image. This task image looks the same as the source code would have looked if it had originally been written as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        MOV     A,C
        CALL    XYZ
        ADD     A,B
        RETURN
        .END
```

PAT uses the .=.+12 in the program counter field to determine where to begin overlaying instructions in the program. It overlays the RETURN instruction with the patch code:

```
        ADD     A,B
        RETURN
```

18.2.3.2 **Adding a Subroutine to a Module** - The second example illustrates a technique for adding a subroutine to an object module. A patch often requires that more than a few lines be added to correct the file. A convenient technique for adding new code is to append it to the end of the module as a subroutine. That way, you insert a CALL instruction to the subroutine at an appropriate location. The CALL instruction directs the program to branch to the new code, execute that code, and then return to inline processing.

The input file for the example is:

```
        .TITLE  ABC
        .IDENT  /01/
ABC::
        MOV     A,B
        CALL    XYZ
        MOV     C,R0
        RETURN
         .
         .
         .
        .END
```

The correction file for this example is:

```
        .TITLE  ABC
        .IDENT  /01.01/
        CALL    PATCH
        NOP
        .PSECT  PATCH
PATCH:
        MOV     A,B
        MOV     D,R0
        ASL     R0
        RETURN
        .END
```

PAT merges the correction file with the input file, as in the first example. The Task Builder then processes the files and produces a task image that looks the same as the source file would have looked if it had originally been written as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        CALL    PATCH
        NOP
        CALL    XYZ
        MOV     C,R0
        RETURN
         .
         .
         .
        .PSECT  PATCH
PATCH:
        MOV     A,B
        MOV     D,R0
        ASL     R0
        RETURN
        .END
```

In this example, the CALL PATCH and NOP instructions overlay the 3-word MOV A,B instruction. (The NOP is included because this is a case where a 2-word instruction replaces a 3-word instruction and NOP is required to maintain alignment.) The Task Builder allocates additional storage for .PSECT PATCH, writes the specified code into this program section, and binds the CALL instruction to the first address in this section. The MOV A,B instruction, replaced by the CALL PATCH instruction, is the first instruction executed by the PATCH subroutine.

18.2.4 Determining and Validating the Contents of a File

You use the Checksum switch (/CS) to determine or validate the contents of a module. The switch directs PAT to calculate the checksum (in octal) for all the binary data that constitutes the module and then inform you of the checksum by means of a diagnostic message.

To determine the checksum of a file, enter the PAT command line with the CS switch applied to that file's specification. For example:

```
=MYFILE/CS,CORRECT.POB
```

The command directs PAT to calculate the checksum for the input file, MYFILE. PAT then responds with the message:

```
INPUT MODULE CHECKSUM IS checksum
```

PAT generates a similar message when you request the checksum for the correction file. For example:

```
=MYFILE,CORRECT.POB/CS
```

After calculating the checksum for the correction file, PAT responds with the message:

```
CORRECTION INPUT FILE CHECKSUM IS checksum
```

If you specify /CS:number to validate the size of a file, PAT calculates the checksum for the file and then compares that checksum with the value you specified as number. If the two values do not match, PAT displays the following message to report the checksum error:

```
ERROR IN FILE filename CHECKSUM
```

For example, you might specify:

```
=MYFILE,CORRECT.POB/CS:432163
```

When PAT calculates the checksum for the correction file, the number is different. PAT then displays the message:

```
ERROR IN FILE CORRECT.POB CHECKSUM
```

Checksum processing always results in an octal, nonzero value.

18.3 PATCH MESSAGES

PAT generates messages that state checksum values and messages that describe error conditions. For checksum values and nonfatal error messages, PAT prefixes the messages with:

```
PAT -- *DIAG*-error message
```

For messages the describe errors that caused PAT to terminate, PAT uses the prefix:

```
PAT -- *FATAL*-error message
```

The messages described below are grouped according to message type, as follows:

- Information messages
- Command line errors
- File specification errors
- Input/output errors
- Errors in file contents or format
- Internal software error
- Storage allocation error

18.3.1 Information Messages

The following messages describe results of checksum processing.

CORRECTION INPUT FILE CHECKSUM IS checksum

Explanation: When you specify /CS in the correction file specification, PAT informs you of the file's checksum value. The value is given in octal.

User Action: No response necessary.

INPUT MODULE CHECKSUM IS checksum

Explanation: When you specify /CS in the input file specification, PAT informs you of the file's checksum value. The value is given in octal.

User Action: No response necessary.

18.3.2 Command Line Errors

The following errors result from failure to adhere to the command line syntax rules.

COMMAND LINE ERROR command line

Explanation: The system standard command line processor (.GCML) detected an error in the command line.

User Action: Reenter the command line using the correct syntax.

COMMAND SYNTAX ERROR command line

Explanation: The command line contained a syntax error.

User Action: Reenter the command line using the correct syntax.

ILLEGAL INDIRECT FILE SPECIFICATION command line

Explanation: You used an indirect command file specification that contains one of the following errors:

- A syntax error
- A specification for a nonexistent indirect command file

User Action: Check for file specification syntax errors or ensure that the specified file is contained in the specified User File Directory. Reenter the command line.

MAXIMUM INDIRECT FILE DEPTH EXCEEDED command line

Explanation: In the command line, you specified an indirect command file that exceeds the maximum nesting level of 2 that is permitted by PAT.

User Action: Reorder your files so that they do not exceed PAT's nesting limit.

18.3.3 File Specification Errors

The following messages are caused by errors in the specification of input or output files or related file switches.

CORRECTION INPUT FILE MISSING command line

Explanation: The mandatory correction file was not specified.

User Action: Reenter the command line specifying the correction file.

ILLEGAL DEVICE/VOLUME SPECIFIED device name

Explanation: The device or volume name specification contained a syntax error.

User Action: Check the rules for specifying devices and volumes, then reenter the command line.

ILLEGAL DIRECTORY SPECIFICATION directory name

Explanation: The directory specification contained a syntax error.

User Action: Check the rules for specifying a directory and reenter the command line.

ILLEGAL FILE SPECIFICATION filename

Explanation: The file specification contained a syntax error.

User Action: Reenter the command line using the correct syntax for the file specification.

ILLEGAL SWITCH SPECIFIED filename

Explanation: An unrecognized switch or switch value was specified with the file.

User Action: Check the rules for specifying the switch and reenter the command line.

INVALID FILE SPECIFIED filename

Explanation: The specification for filename is associated with one of the following error conditions:

- Nonexistent device
- Nonexistent directory - the directory in the filename specification does not exist on the specified device (or on the default device if no device was specified)

User Action: Reenter the command line specifying the correct device or directory.

MULTIPLE OUTPUT FILES SPECIFIED command line

Explanation: PAT accepts only one output file specification.

User Action: Reenter the command line specifying only one output file.

REQUIRED INPUT FILE MISSING command line

Explanation: The mandatory input file was not specified in the command line.

User Action: Reenter the command line specifying an input file.

TOO MANY INPUT FILES SPECIFIED command line

Explanation: Too many input files were specified in the command line. PAT accepts only the input and correction file specifications.

User Action: Reenter the command line specifying the correct files.

UNABLE TO FIND FILE filename

Explanation: PAT could not locate the specified input or correction file.

User Action: Check the directory to ensure that the file exists. Reenter the command line.

18.3.4 Input/Output Errors

The error messages listed below are caused by faults detected while PAT was performing I/O to the specified file.

ERROR DURING CLOSE: FILE: filename

Explanation: This error is most likely to occur while PAT is attempting to write the remaining data into the output file before de-accessing it. The most likely causes of this error are the following conditions:

- The device is full
- The device is write-locked
- A hardware error occurred

User Action: Perform the appropriate corrective action and reenter the command line: if the device is full, delete all unnecessary files; if the device is write-locked, write-enable it; if the problem is a hardware error, contact your DIGITAL Field Service representative.

ERROR POSITIONING FILE filename

Explanation: PAT attempted to position the file beyond end-of-file.

User Action: Submit a Software Performance Report along with the related console dialogue and any other pertinent information.

I/O ERROR ON INPUT FILE filename

Explanation: An error was detected while PAT was attempting to read the specified input file. The principal cause of this error is a device hardware error.

User Action: Reenter the command.

I/O ERROR ON OUTPUT FILE filename

Explanation: An error occurred while PAT attempted to write into the named output file. The most likely causes of this error are the following conditions:

- The device is full
- The device is write-locked
- A device hardware error occurred

User Action: Perform the appropriate corrective action and reenter the command line: if the device is full, delete all unnecessary files; if the device is write-locked, write-enable it; if the problem is a hardware error, contact your DIGITAL Field Service representative.

18.3.5 Errors in File Contents or Format

The following errors represent inconsistencies detected by PAT in the format or contents of the input or correction files.

ERROR IN FILE filename CHECKSUM

Explanation: The checksum that PAT calculated for the named file does not match the one that you specified with /CS:number.

User Action: Ensure that you specified the correct checksum. If the checksum is correct, then you specified an invalid version of the file. Rerun PAT specifying the correct version of the file.

FILE filename HAS ILLEGAL FORMAT

Explanation: The format of the named file is not compatible with the object files produced by the standard DIGITAL language processors or accepted by the Task Builder. The principal causes are:

- Truncated input file
- Input file that consists of text

User Action: Ensure that the file is in the correct format and resubmit it for PAT processing.

INCOMPATIBLE REFERENCE TO GLOBAL SYMBOL symbol name

Explanation: The correction file contains a global symbol whose attributes do not match one or more of the following input file symbol attributes:

- Definition or reference
- Relocatable or absolute

User Action: Update the correction file by modifying the symbol attributes. Reassemble the file and resubmit it for PAT processing.

INCOMPATIBLE REFERENCE TO PROGRAM SECTION section name

Explanation: The correction file contains a section name whose attributes do not match one or both of the following input file section attributes:

- Relocatable or absolute
- Defined with the same directive (.PSECT or .CSECT)

User Action: Update the correction file by modifying the section attribute or changing the section type. Reassemble the file and resubmit it to PAT for processing.

UNABLE TO LOCATE MODULE module name

Explanation: PAT could not find the module name that was specified in the correction file in the file of concatenated input modules.

User Action: Update the input file specification to include the missing module. Reenter the command line.

18.3.6 Internal Software Error

This error reflects internal software error conditions.

ILLEGAL ERROR-SEVERITY CODE error data

Explanation: An error message call, containing an illegal parameter, has been generated.

User Action: If these messages persist, submit a Software Performance Report along with related console dialogue and any other pertinent information.

18.3.7 Storage Allocation Error

The following error message indicates that not enough task memory was available for storing global symbol and program section data.

NO DYNAMIC STORAGE AVAILABLE storage-listhead

Explanation: Not enough contiguous task memory was available to satisfy a request for the allocation of storage.

PAT displays the contents of the 2-word dynamic storage listhead in octal.

User Action: If possible, PAT should be reinstalled with a larger increment or in a bigger partition. (See the RSX-11M/M-PLUS MCR Operations Manual.)

CHAPTER 19

TASK/FILE PATCH PROGRAM (ZAP)

The Task/File Patch Program (ZAP) allows you to directly examine and modify files on a Files-11 volume. Using ZAP, you can patch data files or task images interactively without reassembling and rebuilding the task.

ZAP performs many of the functions performed by the RSX-11 On-Line Debugging Technique program (ODT). Thus, a knowledge of ODT is helpful in using ZAP. See the IAS/RSX-11 ODT Reference Manual for information on ODT.

ZAP provides the following features:

- Command line switches that allow you to access specific words and bytes in a file, modify locations in a task image, list the disk block and address boundaries for each overlay segment in a task disk image, and open a file in read-only mode
- A set of internal registers that include eight Relocation Registers
- Single-character commands that, in combination with other command line elements, allow you to display, open, close, and manipulate the values in task images and data files

Except in read-only mode, the results of ZAP commands are permanent. Thus, the safest way to use ZAP is with a hard-copy terminal, so you have a record of the changes you make.

Although the ZAP utility functions are relatively straightforward to use, patching locations in a task image requires knowing how to use the memory allocation file (or map) generated by the Task Builder as well as the listings generated by MACRO-11. These maps and listings provide information you need to access the locations whose contents you want to change. For information on Task Builder maps, see the RSX-11M/M-PLUS Task Builder Manual. For information on MACRO-11 listings, see the IAS/RSX-11 MACRO-11 Reference Manual.

19.1 INVOKING AND TERMINATING ZAP

You cannot include a command line on the same line on which you invoke ZAP. That is, you cannot type ZAP filespec.

To invoke ZAP, you enter only the utility name, followed by a carriage return, and wait for ZAP to prompt you. Then, enter the file specification for the file you want to change. You enter the file specification in the format:

```
dev:[group,member]filename.filetype;version[/sw...]
```

The default file type is .TSK. After you enter the file specification, ZAP prompts with an underscore (_).

You terminate ZAP by entering the X command. This command exits from ZAP and returns control to MCR.

Section 19.2 describes the ZAP switches and their functions.

19.2 ZAP SWITCHES

Switches set the mode in which ZAP operates. There are three modes: task image (default) and absolute, both of which are addressing modes, and read-only. The modes and their switches are described below:

- Task image mode is the default mode for ZAP. In this mode, addresses in ZAP command lines refer to addresses in the task image file as they are shown in the Task Builder map for the file. (ZAP processing is identical to ODT run-time processing in this case.)

- When you specify absolute mode (/AB), ZAP processes the addresses you enter in ZAP command lines as absolute byte addresses within the file. You should specify /AB for files that are not in task image format, but you can also specify the switch for files that are.

- When you specify read-only mode (/RO), ZAP opens the file in read-only mode. That is, you can execute ZAP functions that change the contents of locations, but these changes are not permanent. When ZAP exits, the original values in the task image file are restored.

In addition, you can specify /LI in task image and read-only modes (but not in absolute mode). This switch displays the starting disk block and address boundaries for each overlay segment in the file in the format:

```
ssssss:  aaaaaa-bbbbbb
```

ssssss:

The starting block in octal.

aaaaaa

The lower address boundary in octal.

bbbbbb

The upper address boundary in octal.

19.3 ADDRESSING LOCATIONS IN A TASK IMAGE

To address locations in a task image, ZAP provides two addressing modes and a set of internal registers, which include eight Relocation Registers. This section first introduces the concept of relocation biases and the use of the Relocation Registers, then explains the addressing modes.

19.3.1 Relocation Biases

When MACRO-11 generates a relocatable object module, the base address of each program section of the module is 000000. In the assembly listing, all locations in the program section are shown relative to this base address.

The Task Builder links program sections to other program sections by mapping the relative addresses applied by the assembler to the physical addresses in memory (for unmapped systems) or to virtual addresses (for mapped systems).

Many values within the resulting task image are biased by a constant whose value is the absolute base address of the program section after the section has been relocated. This bias is called the relocation bias for the program section.

ZAP's eight Relocation Registers, 0R through 7R, are generally set to the relocation biases of the modules to be examined. This allows you to refer to a location in a module by the same relative address that appears in the MACRO-11 listing. The addressing modes help you calculate the relocation biases.

19.3.2 ZAP Addressing Modes: Absolute and Task Image

The two modes for addressing locations in a task image are absolute mode and task image mode. Task image mode is the default mode for ZAP.

You can examine locations in a file in either absolute or task image mode using the RO switch. /RO allows you to open and temporarily change the contents of locations. When ZAP exits, the file is still in its original form.

If you specify /LI to display block/segment information, ZAP is in task image mode after the information is displayed.

19.3.2.1 **Using the Absolute Addressing Mode** - To use ZAP in absolute mode, enter /AB with the file specification after you invoke ZAP.

In absolute mode, ZAP interprets the first address in the file you are changing as segment 1, location 000000. All other addresses you enter are interpreted using this address as the base location. Absolute mode allows you to access all the bytes in a file as well as the label and header blocks of a task disk image. However, to modify a task disk image in absolute mode, you must know the disk layout of the task disk image. Generally, this mode is practical only for data files or for task image files that are not overlaid.

19.3.2.2 **Using the Task Image Addressing Mode** - In task image mode, ZAP allows you to address locations using the block number and relative offset listed in the Task Builder map. Task image mode is useful for changing locations in a file constructed of overlay segments because the Task Builder and ZAP perform the calculations necessary to relate the task's disk structure to its run-time memory structure.

The Task Builder adds blocks containing system information to the beginning of a task image file. The map generated by the Task Builder gives you the starting block and byte offset of the file you want to change.

In task image mode, ZAP allows you to address locations in a task image using the block number and byte offset listed in the Task Builder map and addresses that MACRO-11 prints in an assembler output listing. The following excerpts from a MACRO-11 listing and a map generated by the Task Builder show how to use ZAP in task image mode.

The following lines represent assembled instructions from a MACRO-11 source listing:

```
71 000574  032767   000000G 000000G        BIT     #FE.MUP,$FMASK
72 000602  001002                         BNE     2$
73 000604  000167   000406                 JMP     30$
74 000610  016700   000000G          2$:   MOV     $TKTCB,R0
75 000614  016000   000000G                MOV     T.UCB(R0),R0
76 000620  010067   177534                 MOV     R0,UCB
77 000624  032760   000000G 000000G        BIT     #U2.HLD,U.CW2(R0)
```

The following excerpt from a Task Builder map gives the information you need to address locations in the task image file as they appear in the above MACRO-11 listing:

```
R/W MEM   LIMITS: 120000 123023 003024 01556.
DISK BLK  LIMITS: 000002 000005 000004 00004.

MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE IDENT     FILE

  BLK.:(RW,I,LCL,REL,CON)   120232 002546 01382.
                            120232 002244 01188.   MYFILE  01    MCR.OLB;1
                            122476 000064 00052.   FMTDV   01    MCR.OLB;1
$$RESL:(RW,I,LCL,REL,CON)   123000 000024 00020.
```

Using information in the map above, you can determine the block number and byte offset for the beginning of the file you want to change. The disk-block-limits line lists block 2 as the block where the code segment begins. The synopsis lists byte offset 120232 as the beginning of the file, MYFILE. To address location 574 in the MACRO-11 listing in task image mode, specify the command:

```
2:120232+574/<CR>
```

ZAP responds by opening the location and displaying its contents:

```
002:121026/ 032767
```

19.4 THE ZAP COMMAND LINE AND COMMAND LINE ELEMENTS

ZAP commands perform functions that allow you to examine and modify the contents of locations in a task image file. Command lines comprise combinations of the following elements:

- Commands
- Internal registers
- Arithmetic operators
- Command line element separators
- The current location symbol
- Location-specifier formats

The command elements can be combined with each other to perform multiple functions. The function of a given command line depends not only on which elements you use, but also on the position of one element in relation to the next.

The following sections describe the ZAP command line elements. Sections 19.5 and 19.6 describe how to combine the command line elements to execute ZAP functions.

19.4.1 ZAP Commands

There are three types of ZAP commands:

- Open/close location commands
- General-purpose commands
- Carriage return

19.4.1.1 **Open/Close Location Commands** - Open/close location commands are nonalphanumeric ASCII characters that direct ZAP to perform a sequence of functions. Open/close commands specify two general sequences of operations:

- Open a location, display its contents, and store the contents in the Quantity Register (see Section 19.4.2)
- Close the location after (optionally) modifying it and open another location as specified by the command

Section 19.5 describes the format for specifying open/close location commands.

19.4.1.2 **General-Purpose Commands** - ZAP provides six single-character, general-purpose commands. You use these commands for calculating displacements, verifying location contents, and exiting from ZAP. You can enter some of the commands on the command line with no other parameters. Section 19.6 describes the format for specifying these commands.

19.4.1.3 **Carriage Return** - Unless there is another value or command on the line, the carriage return closes the current location as modified and opens the next sequential location. Unlike ODT, ZAP commands take effect only after you enter a carriage return.

19.4.2 ZAP Internal Registers

ZAP internal registers are fixed storage locations that ZAP uses as registers. These registers contain values set by both you and ZAP. ZAP provides the following internal registers:

- Relocation Registers 0 through 7 (0R through 7R). These registers provide a means for indexing into a module to change the contents of locations in the module. You load the registers with the base address of the modules that have been relocated by the Task Builder.
- The Constant Register (C). You set this register to contain a 16-bit value, which can be specified as an expression.
- The Format Register (F). This register controls the format of the displayed address. If the value of the F Register is 0 (the initial value), ZAP displays addresses relative to the largest value of any Relocation Register whose value is less than or equal to the address to be displayed. If the value of the Format Register is not 0, ZAP displays addresses in absolute format.
- The Quantity Register (Q). ZAP sets the value in the register to be the last value displayed at your terminal.

To access the contents of the registers, specify a dollar sign ($) preceding the register when you enter a command. For example:

```
$C/
```

This command directs ZAP to display the contents of the Constant Register. (The slash, /, is an open command, described in Table 19-3.)

19.4.3 ZAP Arithmetic Operators

Operators are single-character command line elements that define an arithmetic operation in a command line expression. Generally, ZAP evaluates these expressions as addresses. Table 19-1 describes the operators.

You use the operators in expressions in command lines. For example, rather than manually adding all the displacements listed in the Task Builder map, you can specify a location using the following notation:

```
2:120000+170/
```

This method for calculating such a displacement is faster and more accurate than doing it manually.

Table 19-1
ZAP Arithmetic Operators

Operator	Function
+	Adds a value to another value. Used in an expression that ZAP then evaluates to be a command line element.
-	Subtracts a value from another value. Used in an expression that ZAP then evaluates to be a command line element.
*	Multiplies a value by 50(8) and adds it to another value. Used to form a Radix-50 string.

The following example shows how to use the asterisk (*) to form Radix-50 strings. (Section 19.4.4 explains the use of the colon and comma; the percent sign, %, is an open command, described in Table 19-3.)

```
_0,40/<CR>
002:0,000040/  000000
_0*1<CR>
_/<CR>
002:0,000040/  000001
_1*33<CR>
_/<CR>
002:0,000040/  000103
_%<CR>
002:0,000040%  A$
```

19.4.4 ZAP Command Line Element Separators

ZAP provides separators to delimit one command line element from another. Different separators are required depending on the type of ZAP command being executed.

Table 19-2
ZAP Command Line Element Separators

Separator	Function
'	Separates a Relocation Register specification from another command line element.
;	Separates an address from an internal register specification. Used in expressions that set values for Relocation Registers.
:	Separates a block number base value from an offset into the block. Used in most of the references to locations in a file.

19.4.5 The Current Location Symbol

In command line expressions that ZAP evaluates as addresses, a period (.) represents the last open location.

19.4.6 ZAP Command Line Location-Specifier Formats

ZAP has three formats for specifying locations in a command line. Each provides a means of indexing into the task image file. The formats are:

- Byte offset
- Block number:byte offset
- Relocation register, byte offset

19.4.6.1 Byte Offset Format - You specify the byte offset format as follows:

```
location
```

If you are using ZAP in absolute mode, ZAP interprets this specification as a byte offset from block 1, location 000000. If you are using ZAP in task image mode, ZAP interprets this specification as a byte offset from block 0, location 000000.

This format is generally useful only when you are using ZAP in absolute mode. For example, the following ZAP command opens absolute location 664:

```
664/<CR>
```

19.4.6.2 Block Number:Byte Offset Format - This format allows you to specify a byte offset from a specific block in the task image file. Specify the format as follows:

```
blocknum:byteoffset
```

You can use this format for addressing locations whether or not you enter /AB with the file specification.

The Task Builder prints a map that gives information on overlay segments:

```
R/W MEM   LIMITS:  120000 123023 003024 01556.
DISK BLK  LIMITS:  000002 000005 000004 00004.

MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE  IDENT  FILE

. BLK.:(RW,I,LCL,REL,CON)    120232 002546 01382.
                             120232 002244 01188. MYFILE  01  MCR.OLB;1
                             122476 000064 00052. FMTDV   01  MCR.OLB;1
$$RESL:(RW,I,LCL,REL,CON)    123000 000024 00020.
```

In task image mode, ZAP allows you to enter the block number and byte offset displayed in the Task Builder map. In the example above, the disk-block-limits line shows MYFILE beginning on block 2. The synopsis shows that MYFILE has an offset of 120232.

19.4.6.3 **Relocation Register:Byte Offset Format** - This format allows you to load a Relocation Register with the value of a location. The value is then used as a relocation bias. You specify this format for addressing locations in a task image file as follows:

```
relocreg,byteoffset
```

Specify relocreg in the form nR, where n is the number of the Relocation Register. You can then address byte offsets from the value loaded in the Relocation Register. For example:

```
_2:120232;3R<CR>
_3,574/<CR>
```

The first command sets the value of Relocation Register 3, then the second command opens the location that is 574 bytes offset from block (segment) 2, location 120232.

19.5 USING ZAP OPEN/CLOSE COMMANDS

This section is an overview of how to use the ZAP open/close commands, which allow you to open locations in a task image file, modify those locations, and close the locations.

Table 19-3 summarizes the open/close commands.

Table 19-3
ZAP Open/Close Commands

Character	Designation	Function
/	Slash	Opens a location, displays its contents in octal, and stores the contents of the location in the Quantity Register (Q). If the location is odd, it is opened as a byte.
"	Quotation marks	Opens a location, displays the contents of the location as two ASCII characters, and stores the contents of the location in the Quantity Register (Q).
%	Percent sign	Opens a location, displays the contents of the location in Radix-50 format, and stores the contents of the location in the Quantity Register (Q).

(continued on next page)

Table 19-3 (Cont.)
ZAP Open/Close Commands

Character	Designation	Function
\	Backslash	Opens a location as a byte, displays the contents of the location in octal, and stores the contents of the location in the Quantity Register (Q).
'	Apostrophe	Opens a location, displays the contents as one ASCII character, and stores the contents of the location in the Quantity Register (Q).
^ ↑	Circumflex or Up-arrow	Closes the currently open location as modified and opens the preceding location.
_ ←	Underscore or Back-arrow	Closes the currently open location as modified, uses the contents of the location as an offset from the current location value, and opens that location.
@	At sign	Closes the currently open location as modified, uses the contents of the location as an absolute address, and opens that location.
>	Right angle bracket	Closes the currently open location as modified, interprets the low-order byte of the contents of the location as the relative branch offset and opens the target location of the branch.
<	Left angle bracket	Closes the currently open location as modified, returns to the location from which the last series of underscore (_), at sign (@), and/or right angle bracket (>) commands began, and opens the next sequential location.

19.5.1 Opening Locations in a Task Image File

Use any of the five ZAP open commands -- slash (/), quotation marks ("), percent sign (%), backslash (\), or apostrophe (') -- to open a location in a task image file. The format in which ZAP displays the contents of the open location depends on which operator you use.

Once you open a location in a given format, ZAP displays in that format any other locations you open. For example, if you enter the percent sign (%) command, the contents of the open location are displayed in Radix-50 format. If you enter consecutive carriage returns, consecutive locations are displayed in Radix-50 format until you change the format by entering a different special-character open command.

19.5.2 Changing the Contents of a Location

When you open a location with a special-character open command, you can change the contents of that location by entering the new value and a carriage return. The following example is a sequence of commands and ZAP responses that shows how to open a location, change the value of the location, and close the location.

```
 <CR>
002:120000/ 000000
 44444<CR>
_/<CR>
002:120000/ 44444
```

The first command (_) displays the contents (000000) of a word location. The contents are changed by entering the value 44444, and then the location is closed as modified by entering a carriage return. The slash (/) followed by a carriage return displays the new contents of the location (last line of example). The slash indicates that word locations are being opened and their contents displayed in octal format.

19.5.3 Closing Task Image Locations

ZAP has five special-character commands for closing a location in a task image. (The carriage return also closes locations.) The close commands perform three functions:

- Close the current location
- Direct ZAP to another location (such as the preceding location or a location refered to by the current location)
- Open the new location

The following sections give examples of how each command works.

19.5.3.1 **Close a Location and Open the Preceding Location** - Use the circumflex (^) or up-arrow (↑) command (depending on the type of terminal you are using) to close the current location, to direct ZAP to the preceding location, and to open that location. The ASCII representation of both the circumflex and up-arrow is 136(8). The following sequence of ZAP commands and responses shows how this command works:

```
 2:120100/<CR>
002:120100/ 000000
 <CR>
002:120102/ 000111
 <CR>
002:120104/ 000222
 <CR>
002:120106/ 000333
 ^<CR>
002:120104/ 000222
```

The carriage return closes the first three open locations and then opens the next location. The circumflex closes location 120106 and directs ZAP to open the preceding location, 120104.

19.5.3.2 **Close a Location and Open a Location at an Offset from the Location Counter** - Use the underscore (_) or back-arrow (←) command to close the current location, to direct ZAP to use the contents of the current location as an offset from the current location, and to open that location. The ASCII representation of both the underscore and back-arrow is 137(8). The following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/<CR>
002:120100/ 000000
_<CR>
002:120102/ 111111
_<CR>
002:120104/ 222222
_<CR>
002:120106/ 000022
_<CR>
002:120132/ 123456
```

The carriage return closes the first three open locations. The underscore closes location 120106, directs ZAP to use the contents (22) of the current location as the offset from the current location (120110), and then opens that offset location (120132).

19.5.3.3 **Close a Location and Open a Location Offset from the Value of the Just-Closed Location** - Use the at sign (@) command to close a location, to direct ZAP to use the contents of the just-closed location as the absolute address of a location, and to open that location. The following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/<CR>
002:120100/ 000000
_<CR>
002:120102/ 111111
_<CR>
002:120104/ 120114
_@<CR>
002:120114/ 114114
```

The carriage return closes the first three open locations. The at sign closes 120104, directs ZAP to use the contents (120114) of that location as the absolute address of the next location to open, and then opens that location.

19.5.3.4 **Close a Location and Open the Target Location of a Branch Instruction** - Use the right angle bracket (>) command to close the current location, to direct ZAP to use the low-order byte of the contents of the just-closed location as a branch offset, and then to open that location. The following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/<CR>
002:120100/ 005000
_<CR>
002:120102/ 005301
_<CR>
002:120104/ 001020
_><CR>
002:120146/ 052712
```

The carriage return closes the first three open locations. The right angle bracket command closes location 120104, directs ZAP to use the low-order byte of its contents (020) as the branch offset for the address of the next location (120146), and then opens that location.

19.5.3.5 **Close a Location and Open the Location Where the Current Series of Commands Began** - Use the left angle bracket (<) command to close the current location; to direct ZAP to the location where the current series of underscore (_), at sign (@), and/or right angle bracket (>) commands began; and then to open that location. The following sequence of ZAP commands and responses shows how this command works:

```
_1202;0R<CR>
_0,10/<CR>
002:0,000010/ 005212
_ <CR>
002:0,005224/ 001020
_><CR>
002:0,005266/ 000000
_@<CR>
002:0,000000/ 000000
_<<CR>
002:0,000012/ 000430
```

The underscore command directs ZAP to location 005224. The right angle bracket command directs ZAP to location 005266, and the at sign command directs ZAP to location 000000. The left angle bracket command then directs ZAP to location 000012, which is the next sequential address after the location where the sequence of commands began.

19.6 USING ZAP GENERAL-PURPOSE COMMANDS

This section explains the functions of ZAP general-purpose commands and shows the formats for specifying them. Table 19-4 describes the commands. (Table 19-4 is on the following page.)

19.6.1 Exit from ZAP

Use the X command to terminate ZAP and then return control to MCR.

Specify the X command in the format:

```
_X
```

The underscore (_) is the ZAP prompt.

19.6.2 Compute an Offset and Store It in the Quantity Register

Use the K command to compute the offset between the value of the nearest (less than or equal to) Relocation Register and the currently open location, to display the offset value, and to store it in the Quantity Register (Q).

Table 19-4
ZAP General-Purpose Commands

Command	Function
X	Exits from ZAP, returns to MCR
K	Computes the offset between the value of the nearest (less than or equal to) Relocation Register and the currently open location, displays the offset value, and stores it in the Quantity Register (Q)
O	Displays the jump and branch displacements from the current location to a target location
=	Displays in octal the value of the expression to the left of the equal sign
V	Verifies the contents of the current location
R	Sets the value of a Relocation Register

You can enter the K command in the following formats:

K — Calculates the displacement in bytes from the address of the last open location and the value of the Relocation Register whose contents are closest to but less than the value of that address

nK — Calculates the displacement in bytes from the last open location and Relocation Register n

a;nK — Calculates the displacement between address a and Relocation Register n

ZAP responds to the K command by displaying the Relocation Register it used and the offset value it computed in the format:

```
=reg,offset
```

The following example shows how to use the K command:

```
_2:1172;0R<CR>
_2:1232;1R<CR>
_2:1202/<CR>
002:000020/ 000111
_K<CR>
=0,000010
_0,100;1K<CR>
=1,000040
```

19.6.3 Display the Branch and Jump Displacements from the Current Location

Use the O command to display the branch and jump displacements from the current location to a target location. A branch displacement is the low-order byte of a branch instruction which, when executed, branches to the target location. A jump displacement is the offset between the open location and the target location. The jump displacement is used in the second word of a jump instruction if the instruction uses relative addressing.

You can enter the O command in the following formats:

aO — Displays the jump and branch displacements from the current location to the target of the branch (a)

a;rO — Displays the jump and branch displacements from location a to target location r

The following example shows how to use the O command:

```
 0,4534/<CR>
0,4534/ 1234
 4566O<CR>
_000030> 000014
_4534;4566O<CR>
_000030> 000014
_
```

The first number (000030) is the jump displacement, the second number (000014) is the branch displacement.

19.6.4 Display the Value of an Expression

Use the equal sign command (=) to display (in octal) the value of an expression to the left of the equal sign.

Specify the equal sign command in the format:

```
expression=
```

The following example shows how to use the equal sign command:

```
 2:30/<CR>
002:000030/ 000000
 .+177756=<CR>
000006
```

19.6.5 Verify the Contents of a Location

Use the V command to verify that a location contains a specified value.

Specify the V command in the format:

```
contentsV
```

You use V to ensure that, before you have ZAP change them, the contents being changed are what they should be. The V command is mainly useful in indirect command files because ZAP reports the error and exits if the contents do not match. That way, the contents are not changed incorrectly.

The following example shows how to use the V command; if you were using an indirect command file, you would include this sequence of ZAP commands in it.

```
0,1200
6V
10
```

ZAP opens the location that is 1200 offset from the value of Relocation Register 0 and ensures that the value contained at the location is 6. If so, ZAP changes the 6 to 10. If the value is not 6, ZAP terminates.

19.6.6 Set The Value For A Relocation Register

Use the R command to specify the value for a Relocation Register. As explained above in Sections 19.3 and 19.4.2, ZAP uses these registers to index into a module so that you can change the contents of locations in the module.

Specify the R command in the format:

```
nR
```

The variable n is the number of the Relocation Register (0 through 7).

19.7 ZAP ERROR MESSAGES

This section lists the messages generated by ZAP, explains the condition that causes each message, and suggests a response to the message.

ZAP -- ADDRESS NOT WITHIN SEGMENT

Explanation: The address specified was not within the overlay segment specified.

User Action: Reenter the command using the correct address or overlay segment number.

ZAP -- CANNOT BE USED IN BYTE MODE

Explanation: The at sign (@), underscore (_), and right angle bracket (>) commands cannot be used when a location is opened as a byte.

User Action: If the location is an even address, open the location as a word.

ZAP -- ERROR IN FILE SPECIFICATION

Explanation: The file specification was entered incorrectly.

User Action: Check the file specification and reenter it.

ZAP -- ERROR ON COMMAND INPUT

Explanation: An I/O error occurred while a command was being read. This could be a hardware error.

User Action: Ensure that the hardware is functioning properly. If it is, retry the command. If not, call your Field Service representative.

ZAP -- I/O ERROR ON TASK IMAGE FILE

Explanation: An I/O error occurred while the file being modified was being read or written. This could be a hardware error.

User Action: Ensure that the hardware is functioning properly. If it is, retry the command. If not, call your Field Service representative.

ZAP -- NO OPEN LOCATION

Explanation: You attempted to modify data in a closed location.

User Action: Open the location to perform the modification.

ZAP -- NO SUCH INTERNAL REGISTER

Explanation: The character following a dollar sign was not a valid specification for the internal register.

User Action: Reenter the command using the correct specification for the internal register.

ZAP -- NO SUCH RELOCATION REGISTER

Explanation: An invalid number was specified for a Relocation Register.

User Action: Relocation Registers are numbered 0 through 7. Any other numbers are illegal. Reenter the command using a valid Relocation Register number.

ZAP -- NO SUCH SEGMENT

Explanation: The starting disk block was not the start of any segment in the task disk image.

User Action: Reenter the command using the correct disk block address.

ZAP-- NOT A TASK IMAGE OR NO TASK HEADER

Explanation: An error occurred while the segment description tables were being constructed. Possibly, the file is not a task image, /AB was not specified, or the task image is defective.

User Action: Terminate the ZAP session, and try invoking ZAP with /AB specified.

ZAP -- NOT IMPLEMENTED

Explanation: You entered a command that is recognized by ZAP, but not implemented.

User Action: Ensure that you entered the command correctly.

ZAP -- OPEN FAILURE FOR TASK IMAGE FILE

Explanation: The file to be modified could not be opened. Possibly, the file does not exist, the file is locked, the device is not mounted, or you do not have write-access to the file.

User Action: Check the file specifier for errors; ensure that the volume is properly mounted; or use PIP to check your file access privileges (see Chapter 4).

ZAP -- SEGMENT TABLE OVERFLOW

Explanation: ZAP does not have enough room in its partition to construct a segment table.

User Action: Install ZAP in a larger partition, or install ZAP using a larger /INC value. (See the RSX-11M/M-PLUS MCR Operations Manual for information on /INC.)

ZAP -- TOO MANY ARGUMENTS

Explanation: You entered more arguments on the command line than are allowed.

User Action: Reenter the command line using the correct syntax.

ZAP -- UNRECOGNIZED COMMAND

Explanation: ZAP did not recognize the command as entered.

User Action: Check the syntax of the command you are trying to execute and reenter the command using the correct syntax.

ZAP -- VERIFY FAILURE

Explanation: The command for verification (V) determined that the contents of a location did not match the expected value. ZAP terminates.

User Action: If applicable, check for errors in the indirect command file. Ensure that the contents of the file are what they should be. Locate the cause of the error and retry the command. Ensure that you are correcting the right file or file version.

APPENDIXES

APPENDIX A

COMMANDS AND SWITCHES

A.1 INTRODUCTION

This appendix presents a summary of the commands and/or switches used by the RSX-11 utilities described in this manual. Each of the numbered sections of this appendix corresponds, in number, to the chapter discussing that utility. For example, Chapter 4 and Section A.4 both deal with PIP.

Commands and switches are presented alphabetically within the sections of this appendix, regardless of their presentation in the various chapters.

A.2 EDT COMMAND SUMMARY

Change [range] [/NL]

Invokes Character Mode.

COpy range-1 %TO range-2 [/Q] [/SEQ] [/UN]

Copies the lines in range 1 to a position ahead of the first line in range 2.

Delete [range] [/Q]

Deletes lines from the buffer.

EXit [/RE:filespec]

Terminates EDT; writes contents of the main text buffer to specified output file.

Find range

Moves the line pointer to the first line in the range.

INClude [range] /FI:filespec [/SEQ] [/UN]

Locates a file and copies it into a text buffer.

Insert [range] [/SEQ] [/UN]

Inserts text typed at a terminal in the buffer ahead of the first line in the range.

```
                              [/Q]
Move range-1 %TO range-2      [/SEQ]
                              [/UN]
```

Transfers the lines in range 1 to a position ahead of the first line in range 2.

```
PRint [range] /FI:filespec
```

Generates an output file from the contents of the range. The output file contains EDT line numbers as part of the text.

```
QUIT
```

Terminates EDT; saves no edits or text buffers; generates no files.

```
Replace [range] [/SEQ]
                [/UN]
```

First deletes the lines in the range, then inserts text typed at the terminal into the buffer in place of the deleted text.

```
RESequence [range] [/SEQ]
                   [/UN]
```

Assigns new line numbers to the lines in the range.

```
RESTore /FI:filespec
```

Locates the specified file created by a Save command; uses the file's contents to restore the status and contents of the text buffers.

```
SAve /FI:filespec
```

Creates a file that contains the status and contents of the text buffers currently in use.

```
                 ⎧ CASE     ⎧ UPPER ⎫ ⎫
                 ⎪          ⎨ LOWER ⎬ ⎪
                 ⎪          ⎩ NONE  ⎭ ⎪
                 ⎪ EXACT    ⎧ CASE  ⎫ ⎪
                 ⎪          ⎩ NONE  ⎭ ⎪
SET              ⎪                    ⎪
                 ⎪            ⎧ HCPY ⎫ ⎪
                 ⎪            ⎪ VT05 ⎪ ⎪
                 ⎪            ⎪ VT50 ⎪ ⎪
                 ⎪ TErminal   ⎪ VT52 ⎪ ⎪
                 ⎪            ⎪ VT55 ⎪ ⎪
                 ⎪            ⎪ VT61 ⎪ ⎪
                 ⎪            ⎪ LA30 ⎪ ⎪
                 ⎪            ⎩ LA36 ⎭ ⎪
                 ⎩                    ⎭
```

Establishes criteria that other EDT commands use in their operation; flags uppercase or lowercase characters; and establishes proper terminal parameters.

```
      (BUffers  )
      |CAse    |
SHow  |EXACT   |
      |TErminal|
      (VErsion  )
```

Displays the values established by the Set command, as well as current buffer status and software version information.

```
                        [range]
Substitute/str-1/str-2/[/BR]
                        [/Q]
                        [/-T]
```

Changes string 1 to string 2. If range is specified, changes all string 1s in the range. If no range, changes only the first string 1 encountered.

```
Substitute Next
```

Repeats the operation of the Substitute command that it must immediately follow.

```
type [range]
     or
[Type] [range] [/BRief]
```

Displays the contents of the range on your terminal.

```
WRite [range] /FI:filespec
```

Creates an output file from the contents of the range.

```
Xeq range
```

Executes the EDT commands contained in the range.

A.3 EDI COMMAND SUMMARY

```
Add (string)
```

Adds the text specified by string to the end of the current line.

```
AP (string)
```

Same as ADD, except that the new current line is printed.

```
Begin
```

Sets the current line pointer to the top of the block buffer or input file.

```
BLock[ON] or [OFF]
```

Switches text access modes.

```
BO[TTOM]
```

Sets the current line pointer to the bottom of block buffer or input file.

[n]Change /string1/string2[/]

Searches for string1 in the current line and replaces it with the text specified in string2. The integer n allows you to change the first n occurrences of string1 to string2.

CLose filespec

Transfers the remaining lines in the block buffer and the input file into the output file, then closes both the input file and the output file.

CLOSES

Closes secondary input file and begins selecting lines from the primary input file.

CDL [filespec]

Same as the CLOSE command, except that the input file is deleted.

CC [character]

Changes the command concatenation character to the specified character (the default is &).

CTRL/Z

Same as EXIT if in Edit mode; otherwise, causes an immediate exit of EDI.

Delete [n] or [-n]

Deletes the current and next n-1 lines, if n is positive; deletes n lines preceding the current line, but not the current line, if n is negative.

DP [n] or [-n]

Same as DELETE, except that the new current line is printed out.

End

Same as the BOTTOM command.

ERASE [n]

Erases the entire block buffer, the current line, and the next n blocks.

EXit [filespec]

Same as CLOSE command, except that, when files are closed, EDI exits.

EDx [filespec]

Exits from the editing session, closes the output file, and deletes the input file.

FF

Inserts a form feed into the block buffer after the current line.

FIle filespec

Transfers lines from the input file to the file specified by filespec.

[n]Find [string]

Finds the line starting with string or, if n is specified, the nth line starting with string.

Insert [string]

Inserts string immediately following the current line. If string is null, EDI enters Input mode.

KILL

Terminates this editing session; closes the input and output files; and deletes the output file.

[n]LC /string1/string2

Same as CHANGE, except that all occurrences of string1 in the current line are changed to string2.

LIST

Prints on the user terminal all lines in the block buffer or all remaining lines in the input file, starting with current line.

LP

Lists the text in the block buffer or input file on the pseudo device CL:, starting with the current line.

[n]Locate string

Searches the block buffer for string or, if n is specified, the nth occurrence of string.

MAcro x definition

Defines macro x to be definition.

MCall

Retrieves macros from the latest version of file MCALL;n.

[n]Mx [a]

Executes macro x for n executions, passing it the numeric argument a.

[n]<definition>

Allows you to define and execute a macro n times in one step.

Next [n] or [-n]

Establishes a new current line plus or minus n lines from the current line.

NP [n] or [-n]

Same as Next command, except that the new current line is printed.

OPens filespec

Opens the secondary input file.

OUtput [ON] or [OFF]

Turns the output on or off.

Overlay [n]

Deletes the current line and the next n-1 lines, and enters Input mode.

PAGe [n]

Enters block mode, if not already in block mode, and reads page n into the block buffer.

[n]PFind (string)

Identical to FIND command, except that it searches successive block buffers until the nth occurrence of string is found.

[n]PLocate (string)

Same as LOCATE command, except that successive block buffers are searched for the value specified by string.

PAste /string1/string2

Same as the LINE CHANGE command, except that all lines in the remainder of the block buffer or input file are searched for string1. Wherever found, string1 is replaced with string2.

Print [n]

Prints the current line, and the next n-1 lines, on the terminal. The last line printed becomes the current line.

REAd [n]

Reads the next n pages into the block buffer.

RENew [n]

Writes the current buffer and reads in the next. If n is specified, repeats n-1 times.

Retype (string)

Replaces the current line with the text of string. If string is null, the line is deleted.

SAve [n] [filespec]

Saves the current line, and the next n-1 lines, in the file specified by filespec.

SC /string1/string2

Searches for string1, in the block buffer or input file starting with the line following the current line. When string1 is found, replaces all occurrences in the line with string2.

SP

Selects the primary input file.

SS

Selects the secondary input file.

SIZE n

Specifies the maximum number of lines to be read into the block buffer on a single read operation.

TAb [ON] or [OFF]

Turns automatic tabbing on or off.

Top

Same as BEGIN command.

TOF

Returns to the top of the input file, in block mode, and saves all pages previously edited.

TYpe [n]

Same as PRINT command, except that the current line pointer does not change unless EOB is reached.

UNSave [filespec]

Retrieves the lines that were previously saved on filespec and inserts them immediately following the current line.

UC [ON] or [OFF]

Turns uppercase conversion on or off.

Verify [ON] or [OFF]

Displays or suppresses display of lines located or changed.

Write

Writes the current block to the output file, and erases the contents of the buffer.

A.4 PIP COMMAND SUMMARY

outfile[/FO]=infilel [,infile2,...,infilen]/AP[/FO]

Opens an existing file (outfile) and appends the input file(s) onto the end of it. (/FO is the File Owner switch.)

outfile[/BS:n]=infilel[/BS:n][,...,infilen[/BS:n]]

Defines block sizes for 7- and 9-track magnetic tape.

outfile[/switch]=infilel[,infile2,...,infilen][/switch]

Creates a copy of a file on the same or another device. Switch can be:

/BL:n[.]	Block allocated
/CO	Contiguous output
/-CO	Output may be noncontiguous
/FO	File Owner
/NV	New Version
/SU	Supersede

outfile/CD=infile

Gives the output file the creation date of the input file rather than the date of transfer. (This switch cannot be used with the Merge switch, /ME.)

dev:[group,member]/DF

Changes the default device and/or UFD.

infilel[,infile2,...,infilen]/DE[/LD]

Deletes files. (/LD is the List Deleted files subswitch.) See the sections on Delete and Purge in Chapter 4 for a complete description of the List Deleted files subswitch.

outfile=infilel[,infile2,...,infilen]/EN[/NV]

Enters a synonym for a file in a directory with an option to force the version number of outfile to one greater than the latest version for the file. (/NV is the New Version switch.)

infilel/EOF[:block:byte][,...infilen/EOF[:block:byte]]

Specifies the end-of-file pointer for a file.

outfile=/FI:filenum:seqnum

Accesses a file by its file identification number.

dev:/FR

Prints out the available space on a volume and the largest contiguous space on that volume.

/ID

Causes the version of PIP currently in use to be displayed on the terminal.

[listfile]=infile1[,...,infilen]/LI

Lists one or more directories with an option to specify directory listing formats ([listfile] defaults to TI: if not specified). In place of /LI, you can specify one of the alternate mode switches:

```
/BR       Brief format
/FU[:n]   Full format
/TB       Total blocks format
```

infile1[,infile2,...infilen]/sw/NM

Deletes certain PIP error messages, for example, NO SUCH FILE(S); /sw can be /LI (List directory), /DE (Delete files), or /PU (Purge files), or any of their respective subswitches.

infile1/PR[/SY[:RWED][/OW[:RWED]][/GR[:RWED]][/WO[:RWED]][/FO]

Alters file protection. The switches are:

```
/SY -- system access rights
/OW -- owner access rights
/GR -- group access rights
/WO -- world access rights

:/RWED -- read, write, extend, delete privilege
/FO -- File Owner subswitch
```

infile1[,infile2,...,infilen]/PU[:n][/LD]

Deletes a specified range of obsolete versions of a file. (Never deletes latest version.) (/LD is the List Deleted files subswitch.)

infile1[,infile2,...,infilen]/RM

Removes an entry from a directory file. (Opposite of ENTER.)

outfile=infile1[,infile2,...,infilen]/RE[/NV]

Changes the name of a file with an option to force the version number of outfile to one greater than the latest version for the file. (/NV is the New Version switch.)

outfile[/RW]=infile[/RW]

Rewinds a tape.

outdsk:outfile/SB=inmag:infile

File copied to disk from magnetic tape may have records crossing block boundaries. (/SB is the default.)

infilel[,infile2,...infilen]/SD

Prompts for user response before deleting files.

outfile=infile/SR

Allows shared reading of a file that has already been opened for writing.

infilel[,infile2,...,infilen]/SP[:n]

Specifies a list of files to be printed (n is the number of copies). This switch applies only if you have the serial despooler task (RSX-11M) or the queue management system (RSX-11M/M-PLUS) installed.

infilel[,infile2,...infilen]/TR

Truncates file(s) to logical end-of-file.

outfile/UF[/FO]=infile,...,infilen

Creates a User File Directory entry on a volume.

infilel[,infile2,...,infilen]/UN

Unlocks a file that was locked as a result of being improperly closed.

outfile=infilel[,infile2,...,infilen]/UP[/FO]

Opens existing file (infile) and writes it, from the beginning, into outfile. (/FO is the File Owner switch.)

A.5 FLX COMMAND SUMMARY

The FLX commands generally have the form:

outfile=infilel[,infile2,...,infilen]/switch

/switch can be:

/BL:n

Indicates the number of contiguous blocks to be allocated to the output file.

/BS:n

Specifies the block size for cassette tape output.

/CO

Indicates that the output file is to be contiguous.

/DE

Deletes files from a DOS-11 or RT-11 volume.

/DI

Causes a directory listing of DOS-11 or RT-11 volumes; or DOS-11 or RSX-11 cassette tape volumes to be listed.

/DNS:n

Specifies either an 800- or 1600-bpi density for the volume.

/DO

Identifies the file as a DOS-11 formatted file.

/FA:n

Specifies formatted ASCII.

/FB:n

Specifies formatted binary.

/FC

Indicates that FORTRAN carriage control conventions are to be used.

/ID

Requests the current version number of FLX.

/IM:n

Specifies Image mode (n is in decimal bytes).

/LI

Same as DI.

/NU:n

Used with /ZE and /RT switches; specifies the number of directory blocks to allocate.

/RS

Indicates that file is a Files-11 formatted file.

/RT

Indicates that file is an RT-11 formatted file.

/RW and /-RW

Specifies whether the tape will rewind before FLX processes the next command.

/SP

Indicates that the converted file is to be spooled by means of the serial despooler (RSX-11M) or the queue management system (RSX-11M/M-PLUS).

/UI

Indicates that the output file is to have the same UIC as the input file.

/VE

Verifies after write (for cassette only).

/ZE

Initializes DOS-11 and RT-11 volumes and cassettes for DOS-11 or RSX-11 files.

A.6 PRINT AND QUE COMMAND SUMMARY

The format of the PRInt command is:

```
>PRInt [[ddnn:]jobname /jsw=]filespec [/fsw [/fsw]][,filespec [/fsw]]
```

/jsw can be one of the following:

/AFter:hh:mm

Specifies that the job is time-blocked in its queue until after

/switch can be:

/BL:n

Indicates the number of contiguous blocks to be allocated to the output file.

/BS:n

Specifies the block size for cassette tape output.

/CO

Indicates that the output file is to be contiguous.

/DE

Deletes files from a DOS-11 or RT-11 volume.

/DI

Causes a directory listing of DOS-11 or RT-11 volumes; or DOS-11 or RSX-11 cassette tape volumes to be listed.

/DNS:n

Specifies either an 800- or 1600-bpi density for the volume.

/DO

Identifies the file as a DOS-11 formatted file.

/FA:n

Specifies formatted ASCII.

/FB:n

Specifies formatted binary.

/FC

Indicates that FORTRAN carriage control conventions are to be used.

/ID

Requests the current version number of FLX.

/IM:n

Specifies Image mode (n is in decimal bytes).

/LI

Same as DI.

/NU:n

Used with /ZE and /RT switches; specifies the number of directory blocks to allocate.

/RS

Indicates that file is a Files-11 formatted file.

/RT

Indicates that file is an RT-11 formatted file.

/RW and /-RW

Specifies whether the tape will rewind before FLX processes the next command.

/SP

Indicates that the converted file is to be spooled by means of the serial despooler (RSX-11M) or the queue management system (RSX-11M/M-PLUS).

/UI

Indicates that the output file is to have the same UIC as the input file.

/VE

Verifies after write (for cassette only).

/ZE

Initializes DOS-11 and RT-11 volumes and cassettes for DOS-11 or RSX-11 files.

A.6 PRINT AND QUE COMMAND SUMMARY

The format of the PRInt command is:

>PRInt [[ddnn:]jobname /jsw=]filespec [/fsw [/fsw]][,filespec [/fsw]]

/jsw can be one of the following:

/AFter:hh:mm

Specifies that the job is time-blocked in its queue until after the stated time; hh can be from 0 to 23; mm can be from 0 to 59.

/[NO]FLag

Specifies that each file in the job is preceded by one or more file flag pages. The default is /NOFLAG.

/FOrm:n

Sets the forms setting that an output device must have to accept the job. The default for n is 0, which is the standard line printer setup in most installations.

/HOld

Holds the job in its queue until it is released by the QUE jobname /RELease command.

/LEngth:n

Sets the number of lines per page. Length can be set from 1 through 65535. The default is a page of infinite length.

/PRiority:n

Sets the priority of the job in its queue. Priority can be from 0 through 250. Highest priority jobs run first. The default priority is 50.

/[NO]REstart

Makes the job restartable or not. If /RE is specified and the job does not complete its run for some reason, when the system or print processor or Queue Manager is restarted, the job is restarted from the beginning of the first file in the job. /NOREstart is the default.

/fsw can be one of the following:

/COpies:n

Sets the number of copies of the file to be despooled. The default for n is 1.

/[NO]DELete

Directs the Queue Manager to delete the file from its UFD after it is despooled. /NODELete is the default.

The format of the QUE command is:

>QUE job-ID /function [/sw]

/function can be one of the following:

/BRief[:option]

Displays only the job names, the UIC from which the Print command was issued, the queue entry number, and the status of the job.

/FULL[:option]

Displays job names, the UIC from which the Print command was issued, the queue entry number, the status of the job, the file specifications of all files necessary to complete the job, and all attributes of the job, whether set explicitly or by default.

/LIst [:option]

Displays job names, the UIC from which the Print command was issued, the queue entry number, the status of the job, and the file specifications of all files necessary to complete the job. This is the standard format.

:option can be one of the following:

:QUE

Displays entries in the default queue PRINT. This is the default option.

:ALLQUE

Displays all entries in all queues.

:DEV

Displays characteristics of a device. If you have not specified a device in the job-ID field of the command line, you will get an error message.

:ALLDEV

Displays characteristics of all spooled devices.

/DELete

Deletes the specified queue entry. If the job is active at the time the command is issued, the output ceases and the queue entry is deleted.

/HOld

Holds job in its queue. The job remains in its queue, but is not eligible for processing. The effect is the same as with the /HOLD qualifier to the Print command. You cannot hold an active job. (An active job is a job currently being printed.)

/MODify /sw

Modifies the attributes given to queue entries by the Print command. /sw is one of the following:

/AFter:hh:mm

Modifies the time after which the job is to be despooled.

/[NO]FLag

Modifies the FLAG attribute of the job

/FOrm:n

Modifies the FORM attribute of the job.

/LEngth:n

Modifies the LENGTH attribute of the job, which is the number of lines per page.

/PRiority:n

Modifies the queue priority of the job.

/[NO]REStart

Modifies the RESTART attribute of the job.

/RELease

Releases a held job, making it eligible for processing.

A.7 QUEUE MANAGER COMMAND SUMMARY

The following forms of the QUEue command are privileged.

The format of the Queue Manager setup commands is:

>QUE ddnn:/function[:option[s]][/sw[s]]

/function can be one of the following:

/ASsign:queuename

Normally, assigns the default queue PRINT to one or more despooler tasks, but you can assign any queue to any despooler.

/DEAssign:queuename

Eliminates the path between a queue and a despooler, but the queue still exists with all its jobs. Jobs can be added to the queue, but none will be dequeued while the queue is not assigned.

/SPool/sw

Creates queue ddQn and starts the despool task ddPn which must be installed previously for device ddn. This attaches device ddnn: for despooling by the despool task ddPn which is open to jobs from queue ddQn. /sw is one or more of the following.

/FLag:n

Sets the number of flag pages to precede all jobs printed on the device. Note that if you are using special forms, you may want to set this attribute to 0 to avoid printing banner pages on your special forms. This also sets the number of file flag pages to be printed when users specify the FLAGS switch in their Print commands.

/FOrm:n

Sets the device despooler to accept jobs with forms type n. The default for n is 0; it can be between 0 and 255, inclusive.

/STArt[:option] or [/sw]

If you do not enter an option or switch, the device starts printing from the point at which printing stopped. This, in effect, resumes printing after a pause. Option and switch are mutually exclusive fields; that is, you can only use one or the other.

:option can be one of the following:

:BAck:n

Printing starts n pages back from the point at which it was stopped.

:FOrward:n

Printing starts n pages forward from the point at which it was stopped.

:NExt

Printing starts at the top of the next job.

:PAge:n

Printing starts at page n of the current file.

:QMG

Starts the Queue Manager and initializes the default queue PRINT. It creates the file SY:[1,7]QUEUE.SYS if it is not there. If the queue file was intact, all queues still contain their jobs, but are not assigned to processors. (Processors will have to be spooled again.)

:TOP

Printing starts at the top of the current file.

/sw can be one or more of the following:

/FLag:n

Resets the flag setting of the despooler task from the number of flag pages set with SPOOL. The value for n can be 0, 1, 2, or 3.

/FOrm:n

Resets the forms type of the despooler task from the type set with SPOOL.

/STOp [:option]

Stops the device immediately if you do not specify an option. :option can be one of the following:

:EOF

Device stops at the end of the current file.

:EOJ

Device stops at the end of the current job.

:QMG

The Queue Manager stops at the end of the current job. All assignments of queues to devices are broken (deassigned). All despooler tasks are detached from their devices. The Queue Manager stops.

/UNSPool

Frees a device used in despooling for general use. The despool task for the device exits but its entries in queues assigned to it remain in the general file.

A.8 FMT COMMAND SUMMARY

The general format of a FMT command line is:

FMT dev:[/switch1.../switchn]

/switch can be one or more of the following:

/BAD

Runs the Bad BLock Locator task if it is installed. Note that this switch can only be used with operating systems that allow spawning of tasks. All RSX-11M-PLUS and IAS systems permit spawned tasks and RSX-11M systems permit spawned tasks if the option was included at system generation.

/DENS

Selects high (double) or low (single) density for RX02 floppy diskettes.

/ERL

Determines the maximum number of errors FMT will allow on the volume.

/MAN

Enters manual mode and formats the sector or track you specify.

/OVR

Overrides or ignores the manufacturer's bad block sector file (MDBSF).

/VE

Verifies that FMT operation was successfully completed.

/WLT

Rewrites the MDBSF.

/@Y

Informs FMT that it is executing from an indirect command file and does not allow operator intervention in the process.

A.9 BAD COMMAND SUMMARY

The format for executing BAD is:

dev:/swl.../swn

/sw can be one or more of the following:

/CSR=nnnnnn

Alters the CSR address of the device to conform to CSR address of user's system (stand-alone version of BAD only).

/LI

Causes all bad block numbers to be printed on the user's terminal.

/MAN

Allows the user to enter bad blocks.

/NOWCHK

Negates the effect of /WCHK.

/OVR

Causes BAD to ignore the last track of a last-track device.

/UPDATE

Reads bad blocks and prompts for user entries.

/VEC=nnn

Alters the interrupt vector address of the device to conform to the vector address of the device in the user's system (stand-alone version of BAD only).

/WCHK

Causes a write-check operation to take place after each write operation (stand-alone version of BAD only). Not valid for devices DX, DL, or DT.

A.10 BRU COMMAND SUMMARY

The format for executing BRU is:

[/qualifiers] indevl:,...[filespec,...] outdevl:,...[filespec,...]

/qualifiers can be one or more of the following:

/APPEND

Appends new back-up data to a tape with one or more back-up sets.

BACKUP_SET:name

Specifies the name of the back-up set to be placed on tape.

/BAD [:MANUAL | :AUTOMATIC | :OVERRIDE]

Enters the locations of bad blocks on volumes. The default is BAD:AUTOMATIC.

/BUFFERS:number

Specifies the number of directory File Control Blocks (FCBs) per volume kept by the ACP.

/COMPARE

Compares the data on the output volume to the data on the input volume and reports any differences.

/CREATED [:dd-mmm-yy | :(hh:mm:ss) | :(dd-mmm-yy hh:mm:ss) | :BEFORE:dd-mmm-yy | :BEFORE:(hh:mm:ss) | :BEFORE(dd-mmm-yy hh:mm:ss) | :AFTER:dd-mmm-yy | :AFTER:(hh:mm:ss) | :AFTER:(dd-mmm-yy hh:mm:ss)]

Directs BRU to process files created at or before or after a specified date and/or time. If you do not specify a date or time, BRU defaults to the current date and time.

/DENSITY:number

Specifies the data density at which BRU writes to tape. The default is /DENSITY:800.

/DIRECTORY

Displays information (such as back-up set names, file names, or volume number of a tape) on a specified tape volume.

/DISPLAY

Displays at your terminal the UFD and file name of each file being backed up.

/ERRORS:number

Specifies the number of nonfatal I/O errors BRU tolerates on tape reads before automatically terminating execution. The default is 25(10) errors.

/EXCLUDE

Excludes selectively from a back-up or restore operation all files specified on the command line.

/EXTEND:number

Specifies the number of blocks by which to extend a file when that file has exhausted its allocated space.

/HEADERS:number

Specifies the number of file headers to allocate initially to the index file.

/INITIALIZE

Directs BRU to initialize a disk before proceeding with the data transfer.

/INVOLUME:name

Specifies the volume label of the input disk.

/LENGTH:number

Specifies the length of the output tape in decimal feet.

/MAXIMUM:number

Specifies the maximum number of files that can be placed on a volume.

/MOUNTED

Allows you to back up files from a disk that is mounted as a Files-11 volume (by means of the MCR command MOUNT).

/NEW_VERSION

Directs BRU to resolve conflicts resulting from files with identical file specifications by creating a new version of the file. Both versions are kept on the output volume.

/NOINITIALIZE

Specifies that you do not want BRU to initialize the output disk; it already has a Files-11 structure and is mounted as a Files-11 volume.

/NOPRESERVE

Specifies that you do not want BRU to preserve file identifiers.

/NOSUPERSEDE

Resolves the conflict of files on the output volume having file specifications identical to files on the input volume by keeping the file on the output volume rather than the one on the input volume. (The default is /NOSUPERSEDE.)

/OUTVOLUME:name

Specifies the volume label of the output disk.

```
/POSITION:{BEGINNING  }
          {MIDDLE     }
          {END        }
          {BLOCK:number}
```

Specifies the location of the index file, the Master File Directory, and the storage allocation file on a disk.

```
/PROTECTION:{SYSTEM:value}
            {OWNER:value }
            {GROUP:value }
            {WORLD:value }
```

Specifies the global default protection status of files created on a disk after it has been mounted.

```
/REVISED [:dd-mmm-yy                    ]
         [:(hh:mm:ss)                   ]
         [:(dd-mmm-yy hh:mm:ss)         ]
         [:BEFORE:dd-mmm-yy             ]
         [:BEFORE:(hh:mm:ss)            ]
         [:BEFORE:(dd-mmm-yy hh:mm:ss)  ]
         [:AFTER:dd-mmm-yy              ]
         [:AFTER:(hh:mm:ss)             ]
         [:AFTER:(dd-mmm-yy hh:mm:ss)   ]
```

Directs BRU to process files revised at or before or after a specified date and/or time. If you do not specify a date or time, BRU defaults to the current date and time.

/REWIND

Rewinds the first tape of a tape set before execution of a command line.

/SUPERSEDE

Restores files to an existing disk; resolves file specification conflicts by deleting the old file on the output disk and replacing it with the file being restored from the input disk. (The default is /NOSUPERSEDE.)

/TAPE_LABEL:label

Specifies a 6-character ANSI volume identifier for identifying the tape volume.

/VERIFY

Copies data from the input volume to the output volume, compares the volumes, and reports differences.

/WINDOWS:number

Specifies for the output disk the default number of retrieval pointers allocated for use with file windows. The default number is specified when the disk is initialized.

A.11 DSC COMMAND SUMMARY

The format for executing DSC is:

outdev:[label][/switch]=indev:[label][/switch]

/switch can be one or more of the following:

/AP

Appends a DSC file to a magnetic tape that already contains a DSC file.

/BAD= { MAN | NOAUTO | MAN:NOAUTO }

Allows manual entry of bad block locations; can supplement, override, or ignore the disk's own bad block descriptor.

/BL

Sets the number of blocks DSC will provide in each of its buffers.

/CMP

Compares an input file and an output file for differences.

/CSR=xxxx

Alters control status addresses for specific device types.

/DENS=1600
/DENS=800:1600

Overrides the DSC default storage density for magnetic tapes of 800 bits per inch; the first form of the switch creates magnetic tapes at 1600 bits per inch density; the second form (the split density qualifier) creates magnetic tapes with volume header information at 800 bits per inch and the rest of the tape at 1600 bits per inch.

/RW

Rewinds all magnetic tapes before execution of the current command.

/TM02=x

Alters the physical unit formatter.

/UNIT=x

Specifies the physical unit that a LUN accesses.

/VE

Copies data from the input volume and compares it with the output volume following the data transfer.

/VEC=xxx

Alters the vector address of a unit.

A.12 PRESRV COMMAND SUMMARY

The format for executing PRESRV is:

outdev:[label]/switch1.../switchn=indev:[label]/switch1.../switchn

/switch can be one or more of the following:

/BO

Bootstraps the system from the specified device (unit 0 only).

/BL:nnn

Specifies a blocking factor for I/O. Octal is the default for nnn.

/CSR

Changes the control status register for a device.

/ER

Indicates that the PRESRV operation is to be terminated if an I/O error occurs.

/FI

On input, indicates Files-11 format for reading a volume, which overrides implied image mode (/IM). On output, overrides logical tape format (/TP) or implied image mode (/IM).

/FO

Indicates that PRESRV is to format the output volume.

/HE

Displays a summary of PRESRV commands, switches, and defaults.

/IM

Indicates image mode: a transfer of all physical blocks of an input volume.

/LI

Displays a list of all files on a logical tape.

Magnetic tape switches:

/20, /55, /80, /PE

Indicate recording densities of 200, 556, 800, and 1600, respectively.

/RW

Rewinds a cassette or magnetic tape.

/TIM

Specifies the correct time of day to the system.

/TM02

Sets the unit number of the TM02 formatter on the RH controller for a specific TU16 magnetic tape unit.

/TP

Specifies that an output volume be in logical tape format. /TP also overrides the default Files-11 mode on an input volume.

/VE

Verifies the output volume after file transfer.

/VEC

Changes the interrupt vector address for a device.

A.13 VFY COMMAND SUMMARY

listfile,scratchdev=indev/DE
or
indev/DE

Resets the marked-for-delete indicators in the file header area of those files marked for deletion, but never actually deleted.

listfile=indev/FR
or
indev/FR

Prints out the available space on a volume.

listfile,scratchdev=indev/LI
or
indev/LI

Lists the entire index file by file identification.

listfile,scratchdev=indev/LO
or
indev/LO

Scans the entire file structure looking for files that are not in any directory.

listfile=indev/RC[:n]
or
indev/RC[:n]

Checks that every block of every file on the specified volume can be read.

```
listfile,scratchdev=indev/RE
     or
indev/RE
```

Recovers blocks that appear to be allocated, but are not contained in any file.

```
listfile,scratchdev=indev/UP
     or
indev/UP
```

Allocates blocks that appear to be available, but are actually allocated to a file.

A.14 LBR COMMAND SUMMARY

```
outfile/CO:size:ept:mnt:=infile
```

Creates a new library file and transfers contents, but physically deletes logically deleted records in the file and puts all free space at the end of the file.

```
outfile/CR:size:ept:mnt:type
```

Allocates a contiguous library file on a direct access device.

```
outfile/DE:module1[:module-2:...:module-n]
```

Deletes library modules and their associated entry points from a file.

```
outfile/DF:type...
     or
/DF:type
```

Specifies default library file type.

```
outfile/DG:global1[:global2:...:globaln]
```

Deletes specified library module entry points from a file.

```
outfile[/EP]=infile,...,infilen
     or
outfile=infile[/EP],...,infilen[/EP]
```

Includes or excludes entries in the entry point table.

```
outfile=infile/EX[:modulename:...modulenamen]
```

Reads a module from a library; writes it to an output file.

```
outfile/IN=infile1[,infile2,...,infilen]
```

Inserts library modules into a library file.

outfile[,listfile]/switch(es)

Lists all modules in the library file plus additional information depending on which form of the switch you use:

/LI Lists all modules in the library file

/LE Lists all modules in the library file and all their entry points

/FU Lists all modules in the library file and provides a full module description including size, date of insertion, and version

outfile/RP=infilel[,infile2,...,infilen]
or
outfile=infilel/RP[,infile2[/RP],...,infilen[/RP]]

Inserts, and in certain cases, replaces library modules in a library file.

outfile,listfile/SP

Spools the listing file out for printing. This switch applies only if the serial despooler task (RSX-11M) or the queue management system (RSX-11M/M-PLUS) is installed.

outfile=infilel/SS[,infile2[/SS],...,infilen[/SS]]

Sets the selective search attribute bit in the object module header.

outfile/SZ=infilel[,infile2,...,infilen]
or
outfile=infilel/SZ[,infile2[/SZ],...,infilen[/SZ]]

Reduces the size of macro sources by removing comments.

A.15 DMP COMMAND SUMMARY

The format for executing DMP is:

outfile=infile/switch

/switch can be one of the following:

/AS

Specifies that data be dumped in ASCII mode.

/BA:n:m

Specifies a base block address.

/BL:n:m

Specifies the first and last logical blocks to be dumped.

/BY

Specifies that data be dumped in byte octal format.

/DC

Dumps in decimal format.

/DENS:n

Specifies density of a TU16 input magnetic tape when DMP is in device mode only.

/HD

Includes the file header in the data dumped.

/HX

Specifies that data be dumped in hexadecimal byte format.

/ID

Causes the current version of DMP to be printed on the listing.

/LB

Causes starting (logical) block number and a contiguous or noncontiguous indication for the file to be printed.

/LW

Specifies that data be dumped in hexadecimal double-word format.

/MD[:n]

Controls line number sequencing during a memory image dump.

/RC

Dumps one record at a time in the specified format.

/RW

Issues a rewind command to the tape driver before referencing a specified tape. /RW can be used at any time to reposition a tape at the load point (BOT).

/R5

Dumps in Radix-50 format.

/SP

Spools the dump file to the line printer. This switch applies only if you have the serial despooler task (RSX-11M) or the queue management system (RSX-11M/M-PLUS) installed.

/WD

Specifies that data be dumped in hexadecimal word format.

A.16 CMP COMMAND SUMMARY

The CMP command takes the following form:

```
outfile [/sw...]=infile1[/sw...],infile2[/sw...]
```

/sw can be any one of the following:

/BL and /-BL

Specifies that blank lines in both files be included in compare processing. If specified in the form /-BL, blank lines are not included in compare processing; /-BL is the default setting.

/CB and /-CB

Specifies that CMP list infile2 with change bars, in the form of exclamation marks (!), applied to each line that does not have a corresponding line in infile1. /-CB is the default switch.

You can change the change bar character from the exclamation mark to any character you wish by means of the VB switch, described below.

When a section of lines in infile1 has been deleted in infile2 (the output listing file), the first line not deleted is marked.

/CO and /-CO

Specifies that CMP include comments (that is, text preceded by a semicolon) in compare processing. /CO is the default for this switch.

/DI and /-DI

Specifies that CMP print the differences between the two files (rather than marking the lines in infile2).

/DI is the default for this switch.

/CB and /DI are mutually exclusive switches. If both are specified, /CB overrides /DI.

/FF and /-FF

Specifies that CMP include records consisting of a single form-feed character in compare processing. /-FF is the default for this switch.

/LI:n

Specifies that a number (n) of lines must be identical before CMP recognizes a match. If you do not specify this switch, CMP searches for three identical lines to match.

When it encounters a match, CMP prints all the preceding nonmatching lines, along with the first line of the matched sequence of lines to help you find the location in the code where the match occurred.

/LN and /-LN

Specifies that lines in the output file be preceded by their line number. Line numbers are incremented by one for each record read, including blank lines. /LN is the default for this switch. If you specify /SL (below), /LN is unnecessary.

/MB and /-MB

Specifies that CMP include all blank and tab characters in a line in compare processing. If you specify /-MB, CMP interprets any sequence of blank and/or tab characters as a single blank character in compare processing. However, all spaces and tabs are printed in the output listing. /MB is the default switch.

/SL[:au]

Directs CMP to generate an output file suitable for use as SLP command input. When you specify /SL, CMP generates the SLP command input necessary to make infile1 identical to infile2. If a 1- to 8-character alphanumeric symbol is included (:au), an audit trail is specified for SLP input.

/SP[:n] and /-SP

Specifies that the output file be spooled on the line printer. You can optionally specify the number (in octal or decimal) of files to be spooled. /-SP is the default value for this switch.

This switch applies only if you have the serial despooler task (RSX-11M) or the queue management system (RSX-11M/M-PLUS) installed.

/TB and /-TB

Specifies that CMP include all trailing blanks on a line in compare processing. If you specify /-TB, CMP ignores all blanks following the last nonblank character on a line. When you specify /-CO and /-TB together, blanks that precede a semicolon (;) are considered trailing blanks and are ignored. /TB is the default.

/VB:nnn

Specifies an octal character code for use as a change bar. You use this switch with the CB switch. The value nnn specifies the octal character code. For example, you can specify /VB:174 for a vertical bar (if your printer is capable of printing the vertical bar character).

A.17 SLP COMMAND SUMMARY

The SLP utility has only one command:

outfile[/switch,listfile/SP or/-SP]=infile[/switch]

/switch can be any one of the following:

/AU and /-AU

Enables and disables the editing audit trail, which indicates the changes made during the most recent editing session.

/BF and /-BF

Enables and disables blank fill when an audit trail is being produced.

/CM:n

Deletes audit trail, and trailing spaces and tabs beginning at line position n.

/CS[:n]

Calculates the checksum value of the edit commands.

/DB and /-DB

Enables and disables double-spaced listing.

/SP and /-SP

Enables and disables the spooling of listing files to a file structured volume. This switch applies only if the serial despooler task (RSX-11M) or the queue management system (RSX-11M/M-PLUS) is installed.

/TR

Specifies that a diagnostic error message occur when lines are truncated by the audit trail.

A.18 PAT COMMAND SUMMARY

The format for specifying execution of PAT is:

[outfile]=infile[/CS[:number]],correctfile[/CS[:number]]

The CS switch provides the facility to compute the checksum of a specified module.

A.19 ZAP COMMAND SUMMARY

ZAP provides the following commands:

Open/Close Command

/ (slash)

Opens a location, displays its contents in octal, and stores the contents of the location in the Quantity Register (Q). If the location is odd, it is opened as a byte.

" (quotation marks)

Opens a location, displays the contents of the location as two ASCII characters, and stores the contents of the location in the Quantity Register (Q).

% (percent sign)

Opens a location, displays the contents of the location in Radix-50 format, and stores the contents of the location in the Quantity Register (Q).

\ (backslash)

Opens a location as a byte, displays the contents of the location in octal, and stores the contents of the location in the Quantity Register (Q).

' (apostrophe)

Opens a location, displays the contents as one ASCII character, and stores the contents of the location in the Quantity Register (Q).

^ or ↑ (circumflex or up arrow)

Closes the currently open location as modified and opens the preceding location.

_ (underscore)

Closes the currently open location as modified, uses the contents of the location as an offset from the current location value and opens that location.

@ (at sign)

Closes the currently open location as modified, uses the contents of the location as an absolute address, and opens that location.

> (right angle bracket)

Closes the currently open location as modified, interprets the low-order byte of the location as the relative branch offset and opens the target location of the branch.

< (left angle bracket)

Closes the currently open location as modified, returns to the location from which the last series of underscore (_), at sign (@), and/or right angle bracket (>) commands began, and opens the next sequential location.

General-Purpose Commands

X

Exits from ZAP, returns to MCR.

K

Computes the offset between the value of the nearest (less than or equal to) Relocation Register and the currently open location, displays the offset value, and stores it into the Quantity Register (Q).

O

Displays the jump and branch displacements from the current location to a target location.

=

Displays in octal the value of the expression to the left of the equal sign.

V

Verifies the contents of the current location.

R

Sets the value of a Relocation Register.

APPENDIX B

LBR, EDI AND DMP EXAMPLES

B.1 SAMPLE LISTINGS FOR LBR LIST SWITCHES (OBJECT LIBRARY)

B.1.1 List Module Names

```
LBR>MAC,LP:
    or
LBR>MAC,LP:/LI

     DIRECTORY OF FILE MAC.OLB;1
     OBJECT MODULE LIBRARY CREATED BY:  LBR V%02VM
     LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50
     MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
     EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
     FILE SPACE AVAILABLE: 00015 WORDS

     ASGMT
     ASSEM
     CNDTL
     CODHD
     DATDR
     ENBDS
     ENDLN
     ENDPS
     EXPRS
     FLOAT
     GETLN
     GMARG
     INFIL
     INIFL
     INOFL
     LABEL
     LISTC
     LSTNG
     MACRO
     MACRS
     MCALL
     MLIBS
     MSCDR
     NDRCT
     PROCSI
     PROPC
     PROSW
     PST
     READ
     REPT
     ROLHD
     RSDAT
     RSEXEC
```

```
R5UNP
SECTR
SETDIR
SETDN
SETIMM
SETMX
SPACE
STMNT
SYMBL
WORDR
WRITE
```

B.1.2 List Module Names and Full Module Information

```
LBR>MAC,LP:/FU
```

or

```
LBR>MAC,LP:/LI/FU

DIRECTORY OF FILE MAC.OLB;1
OBJECT MODULE LIBRARY CREATED BY:  LBR VX02VM
LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
FILE SPACE AVAILABLE: 00015 WORDS

ASGMT   SIZE:00264  INSERTED:17-JUL-74  IDENT:02
ASSEM   SIZE:00749  INSERTED:1-AUG-74   IDENT:05M
CNDTL   SIZE:00727  INSERTED:31-JUL-74  IDENT:04
CODHD   SIZE:00923  INSERTED:17-JUL-74  IDENT:06
DATDR   SIZE:00414  INSERTED:17-JUL-74  IDENT:07
ENBDS   SIZE:00248  INSERTED:1-AUG-74   IDENT:06
ENDLN   SIZE:00812  INSERTED:31-JUL-74  IDENT:06
ENDPS   SIZE:01060  INSERTED:17-JUL-74  IDENT:04
EXPRS   SIZE:01211  INSERTED:31-JUL-74  IDENT:06
FLOAT   SIZE:00035  INSERTED:17-JUL-74  IDENT:02
GETLN   SIZE:00670  INSERTED:17-JUL-74  IDENT:05
GMARG   SIZE:00290  INSERTED:17-JUL-74  IDENT:01
INFIL   SIZE:00941  INSERTED:9-SEP-74   IDENT:12
INIFL   SIZE:00493  INSERTED:31-JUL-74  IDENT:01
INOFL   SIZE:00960  INSERTED:9-SEP-74   IDENT:01
LABEL   SIZE:00400  INSERTED:1-AUG-74   IDENT:04
LISTC   SIZE:00284  INSERTED:17-JUL-74  IDENT:04
LSTNG   SIZE:00566  INSERTED:17-JUL-74  IDENT:07
MACRO   SIZE:02019  INSERTED:17-JUL-74  IDENT:013
MACRS   SIZE:01540  INSERTED:9-SEP-74   IDENT:09
MCALL   SIZE:00264  INSERTED:31-JUL-74  IDENT:01
MLIBS   SIZE:00807  INSERTED:31-JUL-74  IDENT:06
MSCDR   SIZE:00843  INSERTED:31-JUL-74  IDENT:08
NDRCT   SIZE:00258  INSERTED:31-JUL-74  IDENT:02
PROCSI  SIZE:00216  INSERTED:17-JUL-74  IDENT:01
PROPC   SIZE:00865  INSERTED:17-JUL-74  IDENT:02
PROSW   SIZE:00258  INSERTED:17-JUL-74  IDENT:03
PST     SIZE:01307  INSERTED:17-JUL-74  IDENT:04
READ    SIZE:00198  INSERTED:17-JUL-74  IDENT:01
REPT    SIZE:00473  INSERTED:31-JUL-74  IDENT:01
ROLHD   SIZE:00685  INSERTED:1-AUG-74   IDENT:05
RSDAT   SIZE:00374  INSERTED:17-JUL-74  IDENT:06
RSEXEC  SIZE:00974  INSERTED:1-AUG-74   IDENT:17M
R5UNP   SIZE:00117  INSERTED:17-JUL-74  IDENT:01
```

```
SECTR    SIZE:00551   INSERTED:1-AUG-74   IDENT:04
SETDIR   SIZE:00126   INSERTED:17-JUL-74   IDENT:02
SETDN    SIZE:00670   INSERTED:31-JUL-74   IDENT:06
SETIMM   SIZE:00292   INSERTED:17-JUL-74   IDENT:02
SETMX    SIZE:00131   INSERTED:17-JUL-74   IDENT:01
SPACE    SIZE:00449   INSERTED:22-SEP-74   IDENT:04
STMNT    SIZE:00315   INSERTED:17-JUL-74   IDENT:03
SYMBL    SIZE:00732   INSERTED:17-JUL-74   IDENT:04
WORDB    SIZE:00141   INSERTED:17-JUL-74   IDENT:02
WRITE    SIZE:00189   INSERTED:17-JUL-74   IDENT:01
```

B.1.3 List Module Names, Full Module Information and Module Entry Points (Global Symbols)

```
LBR>MAC,LP:/FU/LE
```

or

```
LBR>MAC,LP:/LI/FU/LE

DIRECTORY OF FILE MAC.OLB;1
OBJECT MODULE LIBRARY CREATED BY:  LBR VX02VM
LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
FILE SPACE AVAILABLE: 00015 WORDS

** MODULE:ASGMT   SIZE:00264   INSERTED:17-JUL-74   IDENT:02

  ASGMT   ASGMTF

** MODULE:ASSEM   SIZE:00749   INSERTED:1-AUG-74   IDENT:05M

  ALLOC$  ASSEM   CLSALL  EDBITS  LCBITS  MACP1   XCTPAS  XCTPRG

** MODULE:CNDTL   SIZE:00727   INSERTED:31-JUL-74   IDENT:04

  CNDBAS  CNDTOP  ENDC    IF      IFF     IFT     IFTF    IIF

** MODULE:CODHD   SIZE:00923   INSERTED:17-JUL-74   IDENT:06

  CPXSTL  INSIZE  OBJDMP  OBJINI  OBJLOC  OBJPNT  OBJSEC  PCRCNT
  PCROLL  PCRTBL  RLDDMP  RLDPNT  STCODE  TSTRLD  ZAPCPX

** MODULE:DATDR   SIZE:00414   INSERTED:17-JUL-74   IDENT:07

  BLKB    IDENT   RADIX   RAD50

** MODULE:ENBDS   SIZE:00248   INSERTED:1-AUG-74   IDENT:06

  EDTBAS  EDTTOP  ENABL
```

```
** MODULE:ENDLN   SIZE:00812  INSERTED:31-JUL-74  IDENT:06

  ENDLIN  ERRBTS  ERRCNT  LINBUF  LINEND  LSTBUF

** MODULE:ENDPS   SIZE:01060  INSERTED:17-JUL-74  IDENT:04

  ENDP1   ENDP2

** MODULE:EXPRS   SIZE:01211  INSERTED:31-JUL-74  IDENT:06

  ABSERR  ABSEXP  ABSTRM  ABSTST  EXPR    GLBEXP  GLBTRM  RELEXP
  RELTRM  RELTST  TERM

** MODULE:FLOAT   SIZE:00035  INSERTED:17-JUL-74  IDENT:02

** MODULE:GETLN   SIZE:00670  INSERTED:17-JUL-74  IDENT:05

  FFCNT   GETLIN  LINNUM  LPPCNT  PAGEXT  PAGNUM  SEGEND
  GMARG   GMARGF  RMARG

** MODULE:INFIL   SIZE:00941  INSERTED:9-SEP-74  IDENT:12

  CMLM2   CMLM3   CMLM4   CMLM5   CSIM2   CSIM5   FINP1   INPM1
  OPENCH  OPNSRC  OPSWT1  OPSWT2  OUTERM  OUTM1   STKM1   $OPSWT

** MODULE:INIFL   SIZE:00493  INSERTED:31-JUL-74  IDENT:01

  SRCNAM  $INIFL

** MODULE:INOFL   SIZE:00960  INSERTED:9-SEP-74  IDENT:01

  LSTNAM  OBJNAM  $INOFL

** MODULE:LABEL   SIZE:00400  INSERTED:1-AUG-74  IDENT:04

  LABEL   LABELF

** MODULE:LISTC   SIZE:00284  INSERTED:17-JUL-74  IDENT:04

  LCTBAS  LCTTOP  LIST    PAGE

** MODULE:LSTNG   SIZE:00566  INSERTED:17-JUL-74  IDENT:07

  CRLF    LINPPG  LSTDEV  LSTREQ  PAGMNE  PF0     PF1     PUTKB
  PUTKBL  PUTLIN  PUTLP   SETBYT  SETPF0  SETPF1  SETWDB  SETWRD

** MODULE:MACRO   SIZE:02019  INSERTED:17-JUL-74  IDENT:013

  ALTSAV  ASCII   ASCIZ   BASCND  BASCOD  BASCPX  BASDMA  BASDUM
  BASEDT  BASLCD  BASLIB  BASLSY  BASMAA  BASMAB  BASMAC  BASPST
  BASREG  BASSAT  BASSEC  BASSRC  BASSST  BASSTK  BASSWT  BASSYM
```

```
  BLKW    BYTMOD  CHRPNT  CLCFGS  CLCLOC  CLCMAX  CLCNAM  CLCSEC
  CNDROL  CODROL  CPXROL  DMAROL  DSABL   DUMROL  EDTROL  ENDFLG
  EOT     ERRMNE  ERR.    ERR.A   ERR.B   ERR.D   ERR.E   ERR.I
  ERR.L   ERR.M   ERR.N   ERR.O   ERR.P   ERR.Q   ERR.R   ERR.T
  ERR.U   ERR.Z   EVEN    FLAGS   IMPPAS  IMPPAT  IMPURE  IMPURT
  IRPC    LCDROL  LIBROL  LSYROL  MAAROL  MABROL  MACP2   MACP2F
  MACROL  MEXIT   MODE    MOVBYT  NLIST   ODD     OPCERR  OPCLAS
  OVMACR  OVSTMT  PASS    PSTROL  REGBAS  REGROL  REGTOP  RELLVL
  ROLBAS  ROLSIZ  ROLTOP  RS.CND  RS.COD  RS.CPX  RS.DMA  RS.DUM
  RS.EDT  RS.LCD  RS.LIB  RS.LSY  RS.MAA  RS.MAB  RS.MAC  RS.PST
  RS.REG  RS.SAT  RS.SEC  RS.SRC  RS.SST  RS.STK  RS.SWT  RS.SYM
  R50ABS  R50DOT  SATROL  SAVREG  SECROL  SECTOR  SETXPR  SIZCND
  SIZCOD  SIZCPX  SIZDMA  SIZDUM  SIZEDT  SIZLCD  SIZLIB  SIZLSY
  SIZMAA  SIZMAB  SIZMAC  SIZPST  SIZREG  SIZSAT  SIZSEC  SIZSRC
  SIZSST  SIZSTK  SIZSWT  SIZSYM  SRCROL  SSTROL  STKROL  SWTROL
  SYMBEG  SYMBOL  SYMROL  TOPCND  TOPCOD  TOPCPX  TOPDMA  TOPDUM
  TOPEDT  TOPLCD  TOPLIB  TOPLSY  TOPMAA  TOPMAB  TOPMAC  TOPPST
  TOPREG  TOPSAT  TOPSEC  TOPSRC  TOPSST  TOPSTK  TOPSWT  TOPSYM
  VALUE   WORD    XCTLIN  XMIT0   XMIT1   XMIT2   XMIT3   XMIT4
  XMIT5   XMIT6   XMIT7

** MODULE:MACRS   SIZE:01540  INSERTED:9-SEP-74  IDENT:09

  ENDLOA  GETBLK  MACR    MACROC  MACROF  MT.MAC  MT.MAX  PROMA
  PROMCF  PROMT   SETMAC  WCIMT

** MODULE:MCALL   SIZE:00264  INSERTED:31-JUL-74  IDENT:01

  MCALL

** MODULE:MLIBS   SIZE:00807  INSERTED:31-JUL-74  IDENT:06

  CPYMAC  FINSML  GETFID  INISML  SMLFDB

** MODULE:MSCDR   SIZE:00843  INSERTED:31-JUL-74  IDENT:08

  END     ERROR   GLOBL   PRINT   SBTTL   SETHDR  TITLE

** MODULE:NDRCT   SIZE:00258  INSERTED:31-JUL-74  IDENT:02

  NARG    NCHR    NTYPE

** MODULE:PROCSI  SIZE:00216  INSERTED:17-JUL-74  IDENT:01

  DSADDR  DSMSK   ENADDR  ENMSK   LIADDR  LIMSK   MLMSK   NLADDR
  NLMSK   PAMSK   PROCSI  SPMSK

** MODULE:PROPC   SIZE:00865  INSERTED:17-JUL-74  IDENT:02

  AEXP    OPCL00  OPCL01  OPCL02  OPCL03  OPCL04  OPCL05  OPCL06
  OPCL07  OPCL08  OPCL09  OPCL10  PROPC
```

```
** MODULE:PROSW    SIZE:00258  INSERTED:17-JUL-74  IDENT:03

  PROSW   SWTBAS  SWTTOP


** MODULE:PST      SIZE:01307  INSERTED:17-JUL-74  IDENT:04

  BSYTOP  DFLCND  DFLGBM  DFLGEV  DFLMAC  DFLSMC  PSTBAS  PSTTOP
  SSTBAS  SSTTOP  WRDSYM


** MODULE:READ     SIZE:00198  INSERTED:17-JUL-74  IDENT:01

  GETVBN  $READ


** MODULE:REPT     SIZE:00473  INSERTED:31-JUL-74  IDENT:01

  ENDMAC  IRP     MPUSH   REPT


** MODULE:ROLHD    SIZE:00685  INSERTED:1-AUG-74   IDENT:05

  APPEND  INSERT  LSREGS  LSFLAG  LSGBAS  LSRCH   LSYBKN  MSRCH
  NEXT    OSRCH   ROLNDX  ROLUPD  SCAN    SCANW   SEARCH  SSRCH
  ZAP


** MODULE:RSDAT    SIZE:00374  INSERTED:17-JUL-74  IDENT:06

  ARG.MAX CNDLVL  CNDMBX  CNDMSK  CNDWRD  CONCNT  CRADTA  EDINIT
  EDMASK  EDMBAK  EDMCSI  ED.AMA  ED.GBL  ED.LSB  ED.REG  ENDVEC
  EXMFLG  GMABLK  GMAPNT  LBLEND  LCBEGL  LCENDL  LCFLAG  LCINIT
  LCLVL   LCMASK  LCMCSI  LCSAVE  LCSAVL  LCSBAK  LC.     LC.BEX
  LC.BIN  LC.CND  LC.COM  LC.LD   LC.LOC  LC.MC   LC.MD   LC.ME
  LC.MEB  LC.SEQ  LC.SRC  LC.SYM  LC.TOC  LC.TTM  LIBNUM  MACGSB
  MACLVL  MACNAM  MACNXT  MACTXT  MACWRT  MSBARG  MSBBLK  MSBCNT
  MSBEND  MSBLGH  MSBMBP  MSBPBP  MSHTXT  MSBTYP  PRGIDN  PRGTTL
  SMLLVL  SRCNUM  STARS   STLBUF  TTLBRK  TTLBUF


** MODULE:RSEXEC   SIZE:00974  INSERTED:1-AUG-74   IDENT:17M

  BUFTBL  CLOSRC  CMIBUF  CMLBLK  CNTTBL  CONT    CSIBLK  DATTIM
  DEFMC   FDBTBL  FDB1    FDB2    GETFLG  GETPLI  HDRTTL  IOFTBL
  IOSEOF  IO.ERR  IO.NNU  IO.OPN  IO.OUT  IO.TTY  LOAMAC  LSTFIL
  MACLDG  OBJBUF  OBJFIL  PASSSW  PURGMC  RESTRT  RLDBUF  SPSAV
  SRCCLO  SRCMRK  SRCPNT  SRCSAV  TSTSTK  VBNSAV  $LIMIT  $LSTVZ
  $SWTCH


** MODULE:R50UNP   SIZE:00117  INSERTED:17-JUL-74  IDENT:01

  R50UNP


** MODULE:SECTR    SIZE:00551  INSERTED:1-AUG-74   IDENT:04

  ASECT   CSECT   LIMIT   PSECT   SATBAS  SATTOP  SECINI
```

```
** MODULE:SETDIR  SIZE:00126  INSERTED:17-JUL-74  IDENT:02
  SETDIR

** MODULE:SETDN   SIZE:00670  INSERTED:31-JUL-74  IDENT:06
  SETDN   SETTIM

** MODULE:SETIMM  SIZE:00292  INSERTED:17-JUL-74  IDENT:02
  SETDSP  SETIMM

** MODULE:SETMX   SIZE:00131  INSERTED:17-JUL-74  IDENT:01
  SETMAX

** MODULE:SPACE   SIZE:00449  INSERTED:22-SEP-74  IDENT:04
  MRKOUT  REMMAC  SHFMSB  SQZSTK

** MODULE:STMNT   SIZE:00315  INSERTED:17-JUL-74  IDENT:03
  STMNT

** MODULE:SYMBL   SIZE:00732  INSERTED:17-JUL-74  IDENT:04
  ARGCNT  ARGPNT  CHSCAN  CTTBL   CT.ALP  CT.COM  CT.EOL  CT.LC
  CT.NUM  CT.PC   CT.PCX  CT.SMC  CT.SP   CT.SPT  CT.TAB  CVTNUM
  DIV     DNC     DNCF    EXPFLG  GETCHR  GETNB  GETR50  GETSYM
  GSARG   GSARGF  MUL     MULR50  SETCHR  SETNB  SETR50  SETSYM
  TSTARG  TSTR50

** MODULE:WORDB   SIZE:00141  INSERTED:17-JUL-74  IDENT:02
  BYTE

** MODULE:WRITE   SIZE:00189  INSERTED:17-JUL-74  IDENT:01
  SQCMD   $WRITE
```

B.1.4 List Module Names and Module Entry Points (Global Symbols)

```
LBR>MAC,LP:/LE
     or
LBR>MAC,LP:/LI/LE

     DIRECTORY OF FILE MAC.OLB;1
     OBJECT MODULE LIBRARY CREATED BY:  LBR VX02VM
     LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50
```

```
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
FILE SPACE AVAILABLE: 00015 WORDS

** MODULE:ASGMT

  ASGMT   ASGMTF

** MODULE:ASSEM

  ALLOC$  ASSEM   CLSALL  EDRITS  LCBITS  MACP1   XCTPAS  XCTPRG

** MODULE:CNDTL

  CNDBAS  CNDTOP  ENDC    IF      IFF     IFT     IFTF    IIF

** MODULE:CODHD

  CPXSTL  INSIZE  OBJDMP  OBJINI  OBJLOC  OBJPNT  OBJSEC  PCRCNT
  PCROLL  PCRTBL  RLDDMP  RLDPNT  STCODE  TSTHLD  ZAPCPX

** MODULE:DATDR

  BLKB    IDENT   RADIX   RAD50

** MODULE:ENBDS

  EDTBAS  EDTTOP  ENABL

** MODULE:ENDLN

  ENDLIN  ERRBTS  ERRCNT  LINBUF  LINEND  LSTBUF

** MODULE:ENDPS

  ENDP1   ENDP2

** MODULE:EXPRS

  ABSERR  ABSEXP  ABSTRM  ABSTST  EXPR    GLBEXP  GLBTRM  RELEXP
  RELTRM  RELTST  TERM

** MODULE:FLOAT

** MODULE:GETLN

  FFCNT   GETLIN  LINNUM  LPPCNT  PAGEXT  PAGNUM  SEQEND
```

```
  GMARG   GMARGF  RMARG

** MODULE:INFIL

  CMLM2   CMLM3   CMLM4   CMLM5   CSIM2   CSIM5   FINP1   INPM1
  OPENCH  OPNSRC  OPSWT1  OPSWT2  OUTERM  OUTM1   STKM1   $OPSWT

** MODULE:INIFL

  SRCNAM  $INIFL

** MODULE:INOFL

  LSTNAM  OBJNAM  $INOFL

** MODULE:LABEL

  LABEL   LABELF

** MODULE:LISTC

  LCTBAS  LCTTOP  LIST    PAGE

** MODULE:LSTNG

  CRLF    LINPPG  LSTDEV  LSTREQ  PAGMNE  PFW     PF1     PUTKB
  PUTKBL  PUTLIN  PUTLP   SETBYT  SETPF0  SETPF1  SETWDB  SETWRD

** MODULE:MACRO

  ALTSAV  ASCII   ASCIZ   BASCND  BASCOD  BASCPX  BASDMA  BASDUM
  BASEDT  BASLCD  BASLIB  BASLSY  BASMAA  BASMAB  BASMAC  BASPST
  BASREG  BASSAT  BASSEC  BASSRC  BASSST  BASSTK  BASSWT  BASSYM
  BLKW    BYTMOD  CHRPNT  CLCFGS  CLCLOC  CLCMAX  CLCNAM  CLCSEC
  CNDROL  CODROL  CPXROL  DMAROL  DSABL   DUMROL  EDTROL  ENDFLG
  EOT     ERRMNE  ERR.    ERR.A   ERR.B   ERR.D   ERR.E   ERR.I
  ERR.L   ERR.M   ERR.N   ERR.O   ERR.P   ERR.Q   ERR.R   ERR.T
  ERR.U   ERR.Z   EVEN    FLAGS   IMPPAS  IMPPAT  IMPURE  IMPURT
  IRPC    LCDROL  LIBROL  LSYROL  MAAROL  MABROL  MACP2   MACP2F
  MACROL  MEXIT   MODE    MOVBYT  NLIST   ODD     OPCERR  OPCLAS
  OVMACR  OVSTMT  PASS    PSTROL  REGBAS  REGROL  REGTOP  RELLVL
  ROLBAS  ROLSIZ  ROLTOP  RS.CND  RS.COD  RS.CPX  RS.DMA  RS.DUM
  RS.EDT  RS.LCD  RS.LIB  RS.LSY  RS.MAA  RS.MAB  RS.MAC  RS.PST
  RS.REG  RS.SAT  RS.SEC  RS.SRC  RS.SST  RS.STK  RS.SWT  RS.SYM
  R50ABS  R50DOT  SATROL  SAVREG  SECROL  SECTOR  SETXPR  SIZCND
  SIZCOD  SIZCPX  SIZDMA  SIZDUM  SIZEDT  SIZLCD  SIZLIB  SIZLSY
  SIZMAA  SIZMAB  SIZMAC  SIZPST  SIZREG  SIZSAT  SIZSEC  SIZSRC
  SIZSST  SIZSTK  SIZSWT  SIZSYM  SRCROL  SSTROL  STKROL  SWTROL
  SYMREG  SYMBOL  SYMROL  TOPCND  TOPCOD  TOPCPX  TOPDMA  TOPDUM
  TOPEDT  TOPLCD  TOPLIB  TOPLSY  TOPMAA  TOPMAB  TOPMAC  TOPPST
  TOPREG  TOPSAT  TOPSEC  TOPSRC  TOPSST  TOPSTK  TOPSWT  TOPSYM
  VALUE   WORD    XCTLIN  XMIT0   XMIT1   XMIT2   XMIT3   XMIT4
  XMIT5   XMIT6   XMIT7
```

```
** MODULE:MACRS

  PRDMCF  PROMT   SETMAC  WCIMT

** MODULE:MCALL

  MCALL

** MODULE:MLIBS

  CPYMAC  FINSML  GETFID  INISML  SMLFDR

** MODULE:MSCDR

  END     ERROR   GLOBL   PRINT   SBTTL   SETHDR  TITLE

** MODULE:NDRCT

  NARG    NCHR    NTYPE

** MODULE:PROCSI

  DSADDR  DSMSK   ENADDR  ENMSK   LIADDR  LIMSK   MLMSK   NLADDR
  NLMSK   PAMSK   PROCSI  SPMSK

** MODULE:PROPC

  AEXP    OPCL00  OPCL01  OPCL02  OPCL03  OPCL04  OPCL05  OPCL06
  OPCL07  OPCL08  OPCL09  OPCL10  PROPC

** MODULE:PROSW

  PROSW   SWTBAS  SWTTOP

** MODULE:PST

  BSYTOP  DFLCND  DFLGRM  DFLGEV  DFLMAC  DFLSMC  PSTBAS  PSTTOP
  SSTBAS  SSTTOP  WRDSYM

** MODULE:READ

  GETVBN  $READ

** MODULE:REPT

  ENDMAC  IRP     MPUSH   REPT

** MODULE:ROLHD

  APPEND  INSERT  LSAFGS  LSFLAG  LSGBAS  LSRCH   LSYBKN  MSRCH
  NEXT    OSRCH   ROLNDX  ROLUPD  SCAN    SCANW   SEARCH  SSRCH
  ZAP
```

```
** MODULE:RSDAT

  ARGMAX  CNDLVL  CNDMFX  CNDMSK  CNDWRD  CONCNT  CRADIX  EDINIT
  EDMASK  EDMBAK  EDMCSI  ED.AMA  ED.GBL  ED.LSB  ED.REG  ENDVEC
  EXMFLG  GMABLK  GMAPNT  LBLEND  LCBEGL  LCENDL  LCFLAG  LCINIT
  LCLVL   LCMASK  LCMCSI  LCSAVE  LCSAVL  LCSBAK  LC.     LC.BEX
  LC.BIN  LC.CND  LC.COM  LC.LD   LC.LOC  LC.MC   LC.MD   LC.ME
  LC.MER  LC.SEQ  LC.SRC  LC.SYM  LC.TOC  LC.TTM  LIBNUM  MACGSB
  MACLVL  MACNAM  MACNXT  MACTXT  MACWRT  MSBAPG  MSBBLK  MSBCNT
  MSBEND  MSBLGH  MSBMRP  MSBPRP  MSBTXT  MSBTYP  PRGIDN  PRGTTL
  SMLLVL  SRCNUM  STARS   STLBUF  TTLBRK  TTLBUF

** MODULE:RSEXEC

  BUFTBL  CLOSRC  CMIBUF  CMLBLK  CNTTBL  CONT    CSIBLK  DATTIM
  DEFMC   FDBTBL  FDB1    FDB2    GETFLG  GETPLI  HDRTTL  IOFTBL
  IO$EOF  IO.ERR  IO.NNU  IO.OPN  IO.OUT  IO.TTY  LOAMAC  LSTFIL
  MACLDG  OBJBUF  OBJFIL  PASSSW  PURGMC  RESTRT  RLDBUF  SPSAV
  SRCCLO  SRCMRK  SRCPNT  SRCSAV  TSTSTK  VARSAV  $LIMIT  $LSTVZ
  $SWTCH

** MODULE:R5UNP

  R50UNP

** MODULE:SECTR

  ASECT   CSECT   LIMIT   PSECT   SATBAS  SATTOP  SECINI

** MODULE:SETDIR

  SETDIR

** MODULE:SETDN

  SETDN   SETTIM

** MODULE:SETIMM

  SETDSP  SETIMM

** MODULE:SETMX

  SETMAX

** MODULE:SPACE

  MRKOUT  REMMAC  SHFMSB  SQZSTK

** MODULE:STMNT

  STMNT
```

```
** MODULE:SYMBL

  ARGCNT   ARGPNT   CHSCAN   CTTBL    CT.ALP   CT.COM   CT.EOL   CT.LC
  CT.NUM   CT.PC    CT.PCX   CT.SMC   CT.SP    CT.SPT   CT.TAB   CVTNUM
  DIV      DNC      DNCF     EXPFLG   GETCHR   GETNR    GETR50   GETSYM
  GSARG    GSARGF   MUL      MULR50   SETCHR   SETNR    SETR50   SETSYM
  TSTARG   TSTR50

** MODULE:WORDB

  BYTE

** MODULE:WRITE

  $QCMO    $WRITE
```

B.2 SAMPLE LISTING FOR LBR LIST SWITCHES (MACRO LIBRARY)

B.2.1 List Module Names

```
LBR>MAC,LP:
```

or

```
LBR>MAC,LP:LI
```

```
DIRECTORY OF FILE EXEMC.MLB;1
MACRO LIBRARY CREATED BY:  LBR VX03.4
LAST INSERT OCCURRED 2-JUN-75 AT 17:16:28
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 52
EPT ENTRIES ALLOCATED: 0; AVAILABLE: 0
FILE SPACE AVAILABLE: 00789 WORDS

ABODFS
CLKDFS
CUCDFS
CVCDFS
DEVDFS
EMBDFS
F11DFS
HDRDFS
HWDDFS
PCBDFS
PKTDFS
TCBDFS
```

B.2.2 List Module Names and Full Module Information

```
LBR>MAC,LP:/LE/FU

      or'

LBR>MAC,LP:/LI/LE/FU

     DIRECTORY OF FILE EXEMC.MLB;1
     MACRO LIBRARY CREATED BY:  LBR VX03.4
     LAST INSERT OCCURRED 2-JUN-75 AT 17:16:25
     MNT ENTRIES ALLOCATED: 64; AVAILABLE: 52
     EPT ENTRIES ALLOCATED: 0; AVAILABLE: 0
     FILE SPACE AVAILABLE: 00789 WORDS

     ** MODULE:ABODFS  SIZE:00201  INSERTED:2-JUN-75

     ** MODULE:CLKDFS  SIZE:00240  INSERTED:2-JUN-75

     ** MODULE:CUCDFS  SIZE:00376  INSERTED:2-JUN-75

     ** MODULE:CVCDFS  SIZE:00601  INSERTED:2-JUN-75

     ** MODULE:DEVDFS  SIZE:01014  INSERTED:2-JUN-75

     ** MODULE:EMBDFS  SIZE:00299  INSERTED:2-JUN-75

     ** MODULE:F11DFS  SIZE:00530  INSERTED:2-JUN-75

     ** MODULE:HDRDFS  SIZE:00321  INSERTED:2-JUN-75

     ** MODULE:HWDDFS  SIZE:00368  INSERTED:2-JUN-75

     ** MODULE:PCBDFS  SIZE:00221  INSERTED:2-JUN-75

     ** MODULE:PKTDFS  SIZE:00233  INSERTED:2-JUN-75

     ** MODULE:TCBDFS  SIZE:00439  INSERTED:2-JUN-75
```

B.3 SAMPLE EDITING OPERATIONS

Four sample editing operations are included in this section to illustrate how the various EDI commands can be used. In the first example, a file is edited using a few basic EDI commands. The second example, illustrates the use of the SAVE, UNSAVE and PASTE commands. In the second example, two save files are generated, modified, and appended to the original file. Any closed file may be appended to or inserted within an open file in the same manner shown in the second example. The third example illustrates how an immediate macro command can be defined and executed in a single step. The last example illustrates how a file containing errors can be edited using the macro commands.

B.3.1 File Editing Sample

```
>EDI PRTBLD.CMD
[PAGE     1]
*P  *
;
; COMMAND FILE TO BUILD
; PRNT SYMBIONT
; FOR RSX-11M MAXXED SYSTEM
;
;
[1,54]PRT/MM/-CP,LP:=PRTBLD/MP
;
; OPTIONS
;
STACK=40
PAR=PARK:0:10000
UNITS=4
TASK=PRT...
ASG=CO:2,LP:3
PRI=60
UC=[10,1]
;
; SPECIFY
; SPECIFY FLAG WHICH CONTROLS
; FILE DELETION AFTER PRINTING
;
; TO ENABLE DELETION USE
;
;       GBLPAT=PRT;$DELET:1
;
; TO INHIBIT DELETION USE
;
; DEFAULT FROM ASSEMBLY IS
; FILE DELETION ENARBLED
;
GBLPAT=PRT:$DELET
/
[*EOB*]
```

File PRTBLD.CMD is opened for editing. A PRINT * command is issued to print the contents of the file. The following errors are detected:

1 - PRNT should be PRINT.
2 - MAXXED should be MAPPED.

3 - /CP should have been used instead of /-CP.

4 - INPUT should be appended to the line containing the word OPTIONS.

5 - PARK should be PAR4K.

6 - UC should be UIC.

7 - The line containing ; SPECIFY should be deleted.

8 - The comment line containing the format used to inhibit deletion is missing.
9 - ENARBLED should be ENABLED.

10 - A :1 should be appended to the line following the word $DELET.

The end of buffer is reached and EDI causes the EOB message to be printed.

```
*TOP
[PAGE     1]
*PL PRNT
; PRNT SYMBIONT
*C/RN/RIN/
; PRINT SYMBIONT
*
; FOR RSX-11M MAXXED SYSTEM
*C/XX/PP/
; FOR RSX-11M MAPPED SYSTEM
*NP 3
[1,54]PRT/MM/-CP,LP:=PRTBLD/MP
*C,/-CP,/CP,
[1,54]PRT/MM/CP,LP:=PRTBLD/MP
*PL PAR=
PAR=PARK:0:10000
*C/RK/R4K/
PAR=PAR4K:0:10000
*NP -3
; OPTIONS
*AP  INPUT
; OPTIONS INPUT
*PL UC
UC=[10,1]
*C/UC/UIC/
UIC=[10,1]
```

A TOP command is issued to move the line pointer to top of file and editing is started.

1 - A PAGE LOCATE command is issued to locate the first line in error and the line is printed automatically. A CHANGE command is issued to correct the line and the corrected line is displayed automatically.

2 - A carriage return is entered following the prompt to move the line pointer and print the next line in error. A CHANGE command is issued to correct the line and the corrected line is displayed automatically.

3 - A NEXT PRINT 3 command is issued to locate the next line in error and the line is printed. A CHANGE command is issued to correct the line and the corrected line is displayed automatically.

4 - A PAGE LOCATE command is issued to locate the next line in error and the line is printed automatically. A CHANGE command is issued to correct the line and the corrected line is displayed automatically.

5 - A line in error was bypassed by mistake; therefore, a NEXT PRINT -3 command is issued to back the line pointer up. An ADD AND PRINT command is used to correct the line

6 - A PAGE LOCATE command is used to located the next line in error and the line is printed automatically. A CHANGE command is issued to correct the line and the corrected line is displayed automatically.

```
*
;
*
; SPECIFY
*DP
; SPECIFY FLAG WHICH CONTROLS
*PL INH
; TO INHIBIT DELETION USE
*I
;
;       GBLPAT=PRT:$DELET:0

*PL RB
; FILE DELETION ENARBLED
*C/R//
; FILE DELETION ENABLED
*
;
*
GBLPAT=PRT:$DELET
*AP :1
GBLPAT=PRT:$DELET:1
*TOF
[PAGE    1]
```

7 - The line pointer is moved down two lines via the carriage return option to locate the next line in error. A DELETE AND PRINT command is issued to delete the line containing ; SPECIFY and print the next line.

8 - A PAGE LOCATE command is issued to locate the point in the file where the new comment lines are to be inserted. EDI is switched to the Input mode, two lines are entered, and EDI is switched back to Edit mode by entering a carriage return as the first character in the line.

9 - A PAGE LOCATE command is issued to locate the next line in error. A CHANGE command is issued to correct the spelling error. The line is displayed automatically.

10 - The line pointer is moved down two lines using two carriage returns to locate the last line in error. An ADD AND PRINT command is issued to append :1 following the word $DELET.

The necessary corrections are complete so the line pointer is moved to the top of the file via a TOF command.

```
*P *
;
; COMMAND FILE TO BUILD
; PRINT SYMBIONT
; FOR RSX-11M MAPPED SYSTEM
;
;
[1,54]PRT/MM/CP,LP:=PRTBLD/MP
;
; OPTIONS INPUT
;
STACK=40
PAR=PAR4K:0:10000
UNITS=4
TASK=PRT...
ASG=CO:2,LP:3
PRI=60
UIC=[10,1]
;
; SPECIFY FLAG WHICH CONTROLS
; FILE DELETION AFTER PRINTING
;
; TO ENABLE DELETION USE
;
;       GBLPAT=PRT:$DELET:1
;
; TO INHIBIT DELETION USE
;
;       GBLPAT=PRT:$DELET:0
;
; DEFAULT FROM ASSEMBLY IS
; FILE DELETION ENABLED
;
GBLPAT=PRT:$DELET:1
/
[*EOB*]
*EX
[EXIT]
```

A PRINT * command is issued to print the complete file with all corrections

An EXIT command is issued to close the file and terminate the editing session.

B.3.2 SAVE and UNSAVE Example

```
*LI
THIS IS LINE 1 PAGE 1
THIS IS LINE 2 PAGE 1
THIS IS LINE 3 PAGE 1
THIS IS LINE 4 PAGE 1
THIS IS LINE 5 PAGE 1
[*EOB*]
*T
*SA 5 SAV1.DAT
```

The file to be used in this example is printed via a LIST command.

The line pointer is returned to the top. A SAVE command is used to save the five lines in a separate file.

```
*T
*SA 5 SAV2.DAT
*CL
EDI>SAV1.DAT
[PAGE    1]
*LI
THIS IS LINE 1 PAGE 1
THIS IS LINE 2 PAGE 1
THIS IS LINE 3 PAGE 1
THIS IS LINE 4 PAGE 1
THIS IS LINE 5 PAGE 1
[*EOB*]
```

The line pointer is returned to the top. A second SAVE command is used to generate a second saved file. The primary input file is closed. The first save file is opened and a LIST command is used to display the file.

```
*PA/PAGE 1/PAGE 2/
THIS IS LINE 1 PAGE 2
THIS IS LINE 2 PAGE 2
THIS IS LINE 3 PAGE 2
THIS IS LINE 4 PAGE 2
THIS IS LINE 5 PAGE 2
*CL
```

A PASTE command is used to change PAGE 1 to PAGE 2 in all lines.

The first save file is closed.

```
EDI>SAV2.DAT
[PAGE    1]
*LI
THIS IS LINE 1 PAGE 1
THIS IS LINE 2 PAGE 1
THIS IS LINE 3 PAGE 1
THIS IS LINE 4 PAGE 1
THIS IS LINE 5 PAGE 1
[*EOB*]
```

The second save file is opened.

The LIST command is used to display the contents of the file.

```
*PA/PAGE 1/PAGE 3/
THIS IS LINE 1 PAGE 3
THIS IS LINE 2 PAGE 3
THIS IS LINE 3 PAGE 3
THIS IS LINE 4 PAGE 3
THIS IS LINE 5 PAGE 3
*CL
EDI>START.DAT
[PAGE    1]
```

A PASTE command is used to change PAGE 1 to PAGE 3 in all lines.

The second save file is closed.
The original input file is opened again.

```
*BO
THIS IS LINE 5 PAGE 1
*UNS SAV1.DAT
*UNS SAV2.DAT
*T
*LI
THIS IS LINE 1 PAGE 1
THIS IS LINE 2 PAGE 1
THIS IS LINE 3 PAGE 1
THIS IS LINE 4 PAGE 1
THIS IS LINE 5 PAGE 1
THIS IS LINE 1 PAGE 2
THIS IS LINE 2 PAGE 2
THIS IS LINE 3 PAGE 2
THIS IS LINE 4 PAGE 2
THIS IS LINE 5 PAGE 2
THIS IS LINE 1 PAGE 3
THIS IS LINE 2 PAGE 3
THIS IS LINE 3 PAGE 3
THIS IS LINE 4 PAGE 3
THIS IS LINE 5 PAGE 3
[*EOB*]
*EX
[EXIT]
```

The last line in the file is located.
Two UNSAVE commands are used to append the two save files to the original input file.
A LIST command is used to display the contents of the combined file.

B.3.3 Use of Immediate Macro Command

```
*LI
ABC IN LINE 1 - ABC
ABC IN LINE 2 - ABC
ABC IN LINE 3 - ABC
ABC IN LINE 4 - ABC
ABC IN LINE 5 - ABC
 .
 .
 .
ABC IN LINE N - ABC
[*EOB*]
*4<F ABC&C/ABC/DEF/>
ABC IN LINE 1 - ABC
DEF IN LINE 1 - ABC
ABC IN LINE 2 - ABC
DEF IN LINE 2 - ABC
ABC IN LINE 3 - ABC
DEF IN LINE 3 - ABC
ABC IN LINE 4 - ABC
DEF IN LINE 4 - ABC
*
```

A LIST command is issued to print the file used in this example.

The immediate macro is defined and executed to find the first four lines which start with ABC and change the first occurrence of the string ABC to DEF. The FIND command causes the line to be printed before the change. The CHANGE command causes the line to be printed after the change.

B.3.4 Use of Macro Commands

```
*LI
THIS LITTLE FILE HAS
MANY CONNON ETTORS SO
WE CAN SHOW YOU HOW
YHE MACRO CONNANDS CAN
BE USED.
FIRST, YHE DESIRED MACRO
MUST BE DEFINED; YHE LINE
POINTER IS MOVED TO A LINE
WITH AN ETTOR; AND YHEN, YHE
MACRO EXECUTE CONNAND
IS ISSUED TO COTTECT YHE
ETTOR
[*EOB*]
*MACRO 1 C/NN/MM/
*MACRO 2 SC/TT/RR/
*MACRO 3 PA/YHE/THE/
*M3
THE MACRO CONNANDS CAN
FIRST, THE DESIRED MACRO
MUST BE DEFINED; THE LINE
WITH AN ETTOR; AND THEN, THE
IS ISSUED TO COTTECT THE
*NP2
MANY CONNON ETTORS SO
*M1
MANY COMMON ETTORS SO
*M2
MANY COMMON ERRORS SO
```

The LIST command is used to print the file and the file is checked for errors. The following errors are located.

1. The string NN is used in place of MM (see macro 1).
2. The string TT is used in place of RR (see macro 2).
3. The string YHE is used in place of THE (see macro 3).

The three macro definitions which will correct the errors are typed.

Macro 3 is used to change all YHE strings to THE.

NP2 is used to locate a line with errors.

M1 is used to change NN to MM.

M2 is used to change TT to RR.

```
*NP2
THE MACRO CONNANDS CAN
*M1
THE MACRO COMMANDS CAN

*M2

WITH AN ERROR; AND THEN, THE
*
MACRO EXECUTE CONNAND
*M1
MACRO EXECUTE COMMAND
*M2
IS ISSUED TO CORRECT THE
*M2

ERROR
*T
*LI
THIS LITTLE FILE HAS
MANY COMMON ERRORS SO
WE CAN SHOW YOU HOW
THE MACRO COMMANDS CAN
BE USED.
FIRST, THE DESIRED MACRO
MUST BE DEFINED; THE LINE
POINTER IS MOVED TO A LINE
WITH AN ERROR; AND THEN, THE
MACRO EXECUTE COMMAND
IS ISSUED TO CORRECT THE
ERROR.
[*EOB*]
```

NP2 is used to locate the next line in error.

M1 is used to change NN to MM.

M2 is used to locate the next TT string
and change it to RR.

A carriage return is used here to locate the next line in error.

M1 is used to change NN to MM.

M2 is used to locate the next TT string and change it to RR.

M2 is used to locate the last error in the file and correct it.

After all lines have been corrected, the file is printed using the LIST command.

B.4 SAMPLE DMP LISTINGS

B.4.1 Use of /LB Switch

```
DMP>TI:=SY:BIGMAC.TSK/LB
 STARTING BLOCK NUMBER = Ø,135163 C
DMP>TI:=SY:SYSGEN.CMD/LB
 STARTING BLOCK NUMBER = Ø,ØØ16Ø6
```

B.4.2 "Standard" Command Line

This command will dump virtual blocks 1 and 2 in SYSGEN.CMD in ASCII mode.

```
DMP>LP:=SY:SYSGEN.CMD/AS/BL:1:2
DMP>

DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0
                VIRTUAL BLOCK 0,000001 - SIZE 512. BYTES

000000  ^A ^● ; ^● ^P ^● ; S Y S G E N P
000020   A R T  1  ^A ^● ; ^● 6 ^● ; ^I D E
000040   T E R M I N E  S Y S T E M F
000060   E A T U R E  S A N D  E C U S T E
000100   M B L E  T H E  E X S K  A  I A
000120   V E ^A ^● ; ^● ' ^● .  A S K  A
000140   R E  Y O U  B U I L D I N G
000160   A  M A P P E D  S Y S T E M ^●
000200   < ^● . A S K  B  A R E  Y O M U
000220    R U N N I N G  O N  A M E
000240   C H I N E  W I T H  M O R D S ^A ^●
000260   T H A N  1 6 K  W O R L D  T H
000300   ; ^● ^^ ^● ;  I N S T A L L  H
000320   E  N E C  C E S S A R Y  T A S
000340   K S ^A ^● ; ^● ^O ^● S S E T U I C
000360   = [ 1 , 5 Ø ] ^● ^G ^● I N S  P I
000400   P ^● ^G ^● I N S  F L X ^● ^N ^● . I
000420   F F  B  I N S  M A C ^Z ^● . I
000440   F F  B  P I P  B I G M A C .
000460   T S K ; * / D E ^Q ^● .  I F T  B
000500    I N S  B I G M  A C ^● ^W ^● . I
000520   F T  B  P I P  M A C  . T S K
000540   ; * / D E ^● ^A ^● ; ^●  ^● ; D E
000560   L E T E  F I L E S  F R O M
000600   P A S T  S Y S G  E N  S R 4 ^● ; ^I
000620   N O T E ; ^I O  N  V I R G O N
000640   D I S K S ,  N O  N E  O F  T
000660   H E  F I ^L E S  W E  T E R Y
000700   T O 3 ^● ; ^I ^I D E L  E  T E  I
000720   L L  E X I S T .  P I P  W
000740   I L L  G I V E  E  R R O R  M
000760   E S S A G E ^● 5 ^● ; ^I ^I T H A
```

DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0
VIRTUAL BLOCK 0,000002 - SIZE 512. BYTES

000000	T		S	H	O	U	L	D		I	N	D	I	C	A	T
000020	E		T	H	A	T		T	H	E		F	I	L	E	
000040	D	O	E	S		N	O	T		E	X	I	S	T	.	F
000060	^A	^@	,	U	^P	^@	S	E	T		/	U	I	C	=	[
000100	1	1	,	2	0	]	^S	^@	P	I	P		R	S	X	A
000120	S	M	.	C	M	D	;	*	/	D	E	K	^S	^@	P	I
000140	P		R	S	X	B	L	D	.	C	M	D	;	*	/	D
000160	E		^N	^@	P	I	P		*	.	O	B	J	;	*	/
000200	D	E	^P	^@	S	E	T		/	U	I	C	=	[	1	1
000220	,	2	4	]	^S	^@	P	I	P		R	S	X	A	S	M
000240	.	C	M	D	;	*	/	D	E	H	^S	^@	P	I	P	
000260	R	S	X	B	L	D	.	C	M	D	;	*	/	D	E	^@
000300	^N	^@	P	I	P		*	.	O	B	J	;	*	/	D	E
000320	^P	^@	S	E	T		/	U	I	C	=	[	1	1	,	1
000340	0	]	^R	^@	P	I	P		S	Y	S	T	B	.	M	A
000360	C	;	*	/	D	E	^R	^@	P	I	P		R	S	X	M
000400	C	.	M	A	C	;	*	/	D	E	^A	^@	;	^@	7	^@
000420	;		N	O	W		W	E		R	U	N		T	H	E
000440		S	Y	S	G	E	N		P	R	O	G	R	A	M	
000460	T	O		S	E	L	E	C	T		T	H	E		E	X
000500	E	C	U	T	I	V	E	G	0	^@	;		F	E	A	T
000520	U	R	E	S		A	N	D		D	E	V	I	C	E	S
000540		W	E		W	A	N	T		I	N		T	H	E	
000560	N	E	W		S	Y	S	T	E	M	;	^@	;	^I	N	O
000600	T	E	:		I	F		Y	O	U		A	R	E		R
000620	U	N	N	I	N	G		O	N		A		1	6	K	
000640	W	O	R	D		M	A	C	H	I	N	E		Y	O	U
000660		S	H	O	U	L	D	S	1	^@	;	^I	S	P	E	C
000700	I	F	Y		T	H	A	T		N	O		A	S	S	E
000720	M	B	L	Y		L	I	S	T	I	N	G	S		B	E
000740		G	E	N	E	R	A	T	E	D	.	R	:	^@	;	^I
000760	I	F		Y	O	U		A	R	E		R	U	N	N	I

B.4.3 Dump Only the Header from SYSGEN.CMD

```
DMP>LP:=SY:SYSGEN.CMD/HD/BL:Ø:Ø
DMP>

DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0
                                        FILE HEADER

SYSGEN.CMD;15       (7157,35146)     7,/10.          11-OCT-74 11:48
  [200,200] [RWED,RWED,RWED,R]


000000     027027 007157 035146 000401 100200 160000 000000 001002
000020     000077 000000 000012 000000 000010 000000 000000 000000
000040     000000 000000 000000 000000 000000 000000 000000 075273
000060     026226 000000 012314 000015 000001 030461 041517 033524
000100     030464 032461 030462 030462 047461 052103 032067 030461
000120     034064 032461 000000 000000 000000 000000 000000 000000
000140     000000 001401 146006 002000 001606 000000 044156 001400
000160     044306 000000 000000 000000 000000 000000 000000 000000
000200     000000 000000 000000 000000 000000 000000 000000 000000
000220     000000 000000 000000 000000 000000 000000 000000 000000
000240     000000 000000 000000 000000 000000 000000 000000 000000
000260     000000 000000 000000 000000 000000 000000 000000 000000
000300     000000 000000 000000 000000 000000 000000 000000 000000
000320     000000 000000 000000 000000 000000 000000 000000 000000
000340     000000 000000 000000 000000 000000 000000 000000 000000
000360     000000 000000 000000 000000 000000 000000 000000 000000
000400     000000 000000 000000 000000 000000 000000 000000 000000
000420     000000 000000 000000 000000 000000 000000 000000 000000
000440     000000 000000 000000 000000 000000 000000 000000 000000
000460     000000 000000 000000 000000 000000 000000 000000 000000
000500     000000 000000 000000 000000 000000 000000 000000 000000
000520     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     000000 000000 000000 000000 000000 000000 000000 000000
000700     000000 000000 000000 000000 000000 000000 000000 000000
000720     000000 000000 000000 000000 000000 000000 000000 000000
000740     000000 000000 000000 000000 000000 000000 000000 000000
000760     000000 000000 000000 000000 000000 000000 000000 166212
```

B.4.4 Use of /BA Switch

The first command sets the base block address to 2, the next command causes virtual blocks 3 and 4 to be dumped.

```
DMP>/BA:Ø:2
DMP>LP:=SYSGEN.CMD/BY/BL:1:2
DMP>
```

```
DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0
                VIRTUAL BLOCK 0,000003 - SIZE 512. BYTES

000000  116 107 040 117 116 040 101 040 115 101 103 110 111 116 105 040
000020  127 111 124 110 040 115 117 122 105 040 124 110 101 116 040 061
000040  066 113 040 127 117 122 104 123 072 000 073 011 131 117 125 040
000060  115 101 131 040 107 105 124 040 101 123 123 105 115 102 114 131
000100  040 114 111 123 124 111 116 107 123 040 117 116 114 131 040 111
000120  106 040 131 117 125 040 104 111 122 105 103 124 040 124 110 105
000140  115 040 124 117 073 000 073 011 124 110 105 040 114 111 116 105
000160  040 120 122 111 116 124 105 122 040 050 114 120 060 072 051 040
000200  117 122 040 101 116 117 124 110 105 122 040 104 111 123 113 056
000220  040 040 124 110 105 122 105 040 111 123 040 116 117 124 040 040
000240  040 103 062 000 073 011 105 116 117 125 107 110 040 123 120 101
000260  103 105 040 117 116 040 124 110 105 040 123 117 125 122 103 105
000300  040 104 111 123 113 040 124 117 040 113 105 105 120 040 101 114
000320  114 040 124 110 105 040 031 000 073 011 101 123 123 105 115 102
000340  114 131 040 114 111 123 124 111 116 107 040 106 111 114 105 123
000360  056 073 001 000 073 105 010 000 122 125 116 040 044 123 107 116
000400  001 000 073 101 063 000 073 040 101 124 040 124 110 111 123 040
000420  120 117 111 116 124 040 127 105 040 122 105 116 101 115 105 040
000440  124 110 105 040 101 123 123 105 115 102 114 131 040 103 117 115
000460  115 101 116 104 040 106 111 114 105 040 046 000 073 040 101 116
000500  104 040 124 110 105 040 123 131 123 124 105 115 040 102 125 111
000520  114 104 040 103 117 115 115 101 116 104 040 106 111 114 105 040
000540  124 117 020 000 056 111 106 106 040 101 040 073 011 133 061 061
000560  054 062 060 135 020 000 056 111 106 124 040 101 040 073 011 133
000600  061 061 054 062 064 135 001 000 073 125 057 000 056 111 106 106
000620  040 101 040 120 111 120 040 133 061 061 054 062 060 135 057 122
000640  105 075 122 123 130 101 123 115 056 103 115 104 073 052 054 122
000660  123 130 102 114 104 056 103 115 104 073 052 011 057 000 056 111
000700  106 124 040 101 040 120 111 120 040 133 061 061 054 062 064 135
000720  057 122 105 075 122 123 130 101 123 115 056 103 115 104 073 052
000740  054 122 123 130 102 114 104 056 103 115 104 073 052 000 077 000
000760  056 101 123 113 040 132 040 104 111 104 040 131 117 125 040 101
```

```
DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0
                VIRTUAL BLOCK 0,000004 - SIZE 512. BYTES

000000  116 123 127 105 122 040 124 110 105 040 123 131 123 107 105 116
000020  040 121 125 105 123 124 111 117 116 123 040 124 117 040 131 117
000040  125 122 040 123 101 124 111 123 106 101 103 124 111 117 116 040
000060  020 000 056 111 106 124 040 132 040 056 107 117 124 117 040 061
000100  060 060 001 000 073 111 074 000 073 040 127 105 040 127 111 114
000120  114 040 105 130 111 124 040 116 117 127 040 123 117 040 124 110
000140  101 124 040 131 117 125 040 115 101 131 040 122 105 123 124 101
000160  122 124 040 124 110 111 123 040 103 117 115 115 101 116 104 040
000200  106 111 114 105 067 000 073 040 106 122 117 115 040 124 110 105
000220  040 102 105 107 111 116 116 111 116 107 056 040 124 110 111 123
000240  040 127 111 114 114 040 101 114 114 117 127 040 101 040 103 114
000260  105 101 116 040 125 120 040 117 106 040 124 110 105 122 070 000
000300  073 040 104 111 123 113 040 101 116 104 040 101 040 116 105 127
000320  040 122 125 116 040 117 106 040 123 107 116 040 124 117 040 123
000340  105 114 105 103 124 040 124 110 105 040 120 122 117 120 105 122
000360  040 117 120 124 111 117 116 123 022 000 073 040 106 117 122 040
000400  131 117 125 122 040 123 131 123 124 105 115 056 001 000 073 040
000420  012 000 056 107 117 124 117 040 061 060 060 060 006 000 056 061
000440  060 060 072 073 037 000 073 040 116 117 127 040 127 105 040 101
000460  123 123 105 115 102 114 105 040 124 110 105 040 105 130 105 103
000500  125 124 111 126 105 040 001 000 073 124 027 000 056 111 106 106
000520  040 101 040 123 105 124 040 057 125 111 103 075 133 061 061 054
000540  062 060 135 000 027 000 056 111 106 124 040 101 040 123 105 124
000560  040 057 125 111 103 075 133 061 061 054 062 064 135 073 013 000
000600  115 101 103 040 100 122 123 130 101 123 115 000 001 000 073 106
000620  062 000 073 040 116 117 127 040 127 105 040 102 125 111 114 104
000640  040 124 110 105 040 103 117 116 103 101 124 105 116 101 124 105
000660  104 040 117 102 112 105 103 124 040 115 117 104 125 114 105 040
000700  106 111 114 105 024 000 073 040 106 117 122 040 124 110 105 040
000720  105 130 105 103 125 124 111 126 105 056 001 000 073 104 024 000
000740  120 111 120 040 122 123 130 061 061 115 056 117 102 123 075 052
000760  056 117 102 112 026 000 120 111 120 040 056 117 102 112 057 122
```

APPENDIX C

RSX-11M SERIAL DESPOOLER TASK

The RSX-11M Serial Despooler task (PRT...) provides a means of eliminating contention for the system line printer. Rather than waiting for the line printer to become available, a task directs the output intended for the line printer to a disk file. The task issues a Send Data directive to the serial despooler, placing a data block that identifies the file to be spooled in the serial despooler queue. A Request directive is then issued by the task to activate the serial despooler, in case it is not already active. PRT handles FCS-created files, but RMS files can be read only if they are sequential. All files identified in the serial despooler queue are printed in first-in-first-out (FIFO) order.

You may wish to compare the capability of this task with the RSX-11M Queue Manager and its associated despooler. See Chapters 6 and 7 of this manual for more information.

C.1 RECEIVE QUEUE OPERATION

The standard method of placing a user file in the serial despooler receive queue (and requesting its execution) is via the PRINT$ macro call, which is described in the IAS/RSX-11 I/O Operations Reference Manual. Files are spooled in this same manner by the RSX-11 utilities that support the spool (/SP) option. Each entry in the serial despooler receive queue consists of a 13-word data block containing the file-related information illustrated in Figure C-1.

C.2 TEXT REQUIREMENTS

The serial despooler task prints ASCII text with a maximum line length of 132 bytes. It will properly handle files with all modes of FCS carriage control (that is, standard, embedded, and FORTRAN).

C.3 TASK-BUILD INFORMATION

The serial despooler task must be built during an RSX-11M system generation because the task image file (PRT.TSK) is not distributed on the standard release kits. Normally, the serial despooler is built to retain all files that have been spooled, but the serial despooler build file can be edited during system generation to enable the automatic delete feature. (Note also, that a nondeleting despooler can be made to delete, if the data block mentioned in Section C.1 and shown in Figure C-1 has a negative value in word 7.) When the serial despooler is built without automatic delete, spooled files are

retained after printing. If the system has a deleting serial despooler, all spooled files are deleted after printing.

See the RSX-11M System Generation and Management Guide for detailed system generation information.

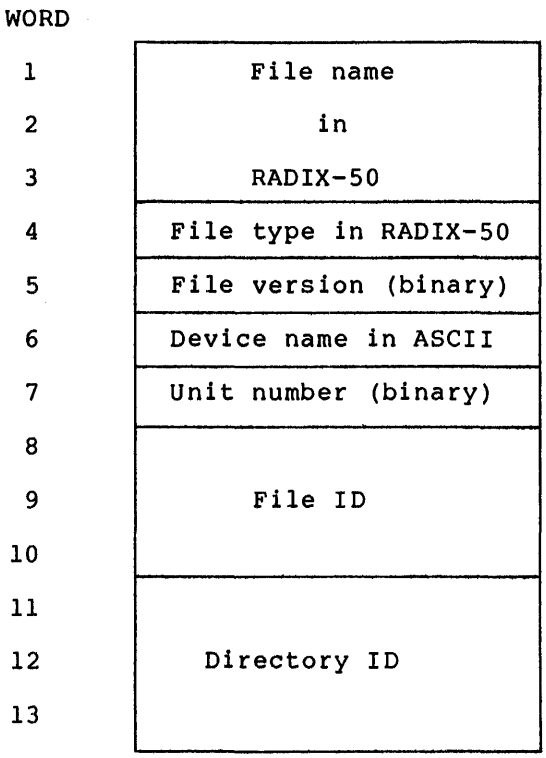


Figure C-1 PRT Send Data Buffer Format

The serial despooler is set to spool files to LP0:. It uses LUN3 as the spool device. The system manager can redirect print files to any record-oriented device (that is, a device similar to the printer) by reassigning LUN3. See the RSX-11M/M-PLUS MCR Operations Manual for information on the Reassign command.

C.4 PRT ERROR MESSAGES

All error messages issued by PRT are sent to the console terminal via pseudo-device CO:. The error messages have the following format:

```
PRT -- text
```

In all but the receive failure error, the messages supply information that identifies the sender task and the file in question. All PRT errors are fatal; upon error detection, printing of the input file is terminated, and a cleanup/restart procedure is entered.

In the case of the receive failure error, the sender and file information are unavailable. Furthermore, PRT does not attempt to dequeue additional spool requests because of the nature of this error condition. Instead, PRT exits causing its receive queue to be purged by the system.

RECEIVE FAILURE, d. -- TASK EXITING

Explanation: The Receive Data or Exit directive failed while attempting to obtain the next file specifier from the queue. The system error code (d.) is printed to identify the error.

NO DEVICE NAME - SENDER: task FILE: filename.typ;ver

Explanation: The dequeued print request did not contain a device name.

NO FILE ID - SENDER: task FILE: filename.typ;ver

Explanation: The dequeued print request did not contain a file ID.

OPEN FAILURE INPUT FILE - SENDER: task FILE: filename.typ;ver, d.

Explanation: The specified file could not be opened. One of the following conditions may exist:

1. The file is protected against access for read privileges.
2. A problem exists on the physical device (for example, device cycled down).
3. The volume is not mounted.
4. The specified file directory does not exist.
5. The named file does not exist in the specified directory.
6. The file is already deleted.

The system error code (d.) is printed to identify the failure.

ATTACH FAILURE - SENDER: task FILE: filename.typ;ver, d.

or

DETACH FAILURE - SENDER: task FILE: filename.typ;ver, d.

Explanation: The line printer could not be attached/detached (that is, the system does not contain a line printer). The system error code (d.) is printed to identify the error.

PRINT ERROR - SENDER: task FILE: filename.typ;ver, d.

Explanation: A Queue I/O request to the line printer has failed. The system error code (d.) is printed to identify the error.

I/O ERROR INPUT FILE - SENDER: task FILE: filename.typ;ver, d.

Explanation: An error was detected while reading the input file. One of the following conditions may exist:

1. A problem exists on the physical device (for example, device cycled down).

2. Length of the text line is greater than 132 bytes.

3. File is corrupted or the format is incorrect.

The system error code (d.) is printed to identify the error.

d. LINES WERE TRUNCATED

Explanation: While printing the file PRT truncated d. lines.

APPENDIX D

CRF--THE RSX-11 CROSS-REFERENCE PROCESSOR

CRF, the RSX-11 cross-reference processor, is an independent task that produces cross-reference listings for the MACRO-11 and Task Builder tasks. CRF is invoked by the /CR switch on the MACRO-11 or Task Builder command line. Once execution is complete, CRF builds cross-reference listings using the execution-time symbol tables built by those tasks.

Two cross-reference listings are built for the Task Builder, one listing the modules that reference global symbols during task execution and another that shows the modules contained in a given overlay segment.

Four cross-reference listings are produced for the MACRO-11 task, each showing a page and line number reference to a type of symbol referenced during execution of the MACRO-11 assembler.

This appendix describes how CRF processes data and shows the formats of files CRF uses during that processing. Also, CRF error messages are listed.

For information on how to invoke CRF via the MACRO-11 command line, refer to the IAS/RSX-11 MACRO-11 Reference Manual. For information on how to invoke CRF via the Task Builder command line, refer to the RSX-11M/M-PLUS Task Builder Manual.

D.1 HOW CRF PROCESSES DATA

When the /CR switch is specified on the MACRO-11 or Task Builder command line, those tasks perform additional processing before passing control to the CRF processor. This section describes how cross-reference listings are generated in two phases. The first phase consists of the processing steps performed by MACRO-11 or Task Builder; the second phase consists of processing steps performed by CRF. Figure D-1 is an overview showing the processing steps performed in each phase, along with input and output files used.

D.1.1 MACRO-11/Task Builder Processing

The first steps to produce cross-reference listings are performed by MACRO-11 or Task Builder. These processors generate three files during execution: an object file (file.OBJ) or task image file (file.TSK), a listing file (file.LST or file.MAP), and a CRF symbol table file (file.CRF).

The object file or task image file is directed to an appropriate device and does not affect CRF processing.

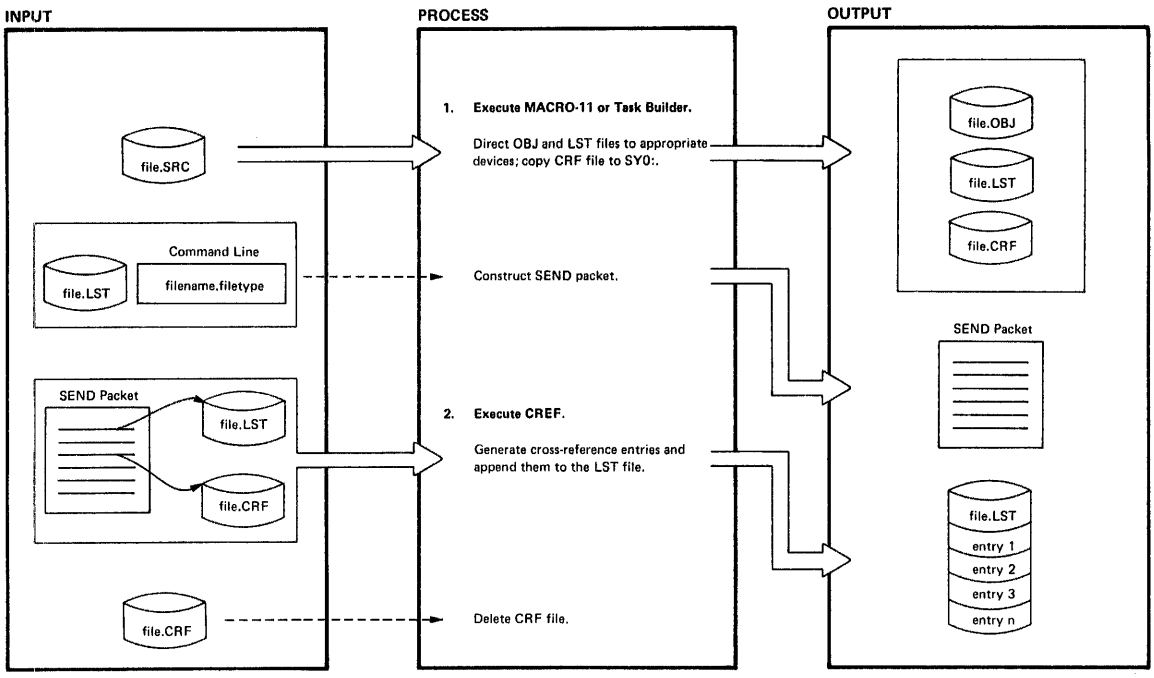


Figure D-1 How MACRO-11, Task Builder, and CRF Generate Cross-Reference Listings

The listing file consists of text information generated during the execution of the processors, for example, the listing of the source code in an assembler language program. If the output device for the listing file is sequential or record-oriented, as in the case of LP:, a temporary listing file is created on SY0:.

The CRF symbol table file consists of symbol tables built during MACRO-11 or Task Builder execution. These tables are used by CRF to generate cross-reference listings.

The task originating the request for cross-reference processing then constructs a SEND packet using information contained in the listing file and the filename and filetype specified on the command line. The SEND packet contains enough data to identify the symbol table file and the listing file. CRF receives the packet and uses it to locate the symbol table file and listing file produced by MACRO-11 or Task Builder.

The formats of the symbol table file and the SEND packet are shown in Figures D-2 and D-3.

If spooling is requested and the output device is a random-access device, the spooling flag in the SEND packet is set. CRF then passes the listing file to the print spooler for printing.

D.1.2 CRF Processing

The SEND packet contains pointers to the CRF symbol table file and the listing file. CRF uses the information in the listing file and the SEND packet to construct a file specification to use to open the symbol table file for processing:

```
symbol table filename = text filename
symbol table filetype = CRF
symbol table version  = contents of SEND packet word 11
symbol table
device and unit       = listing file device and unit
```

CRF then generates the requested cross-references and appends them to the listing file. When all cross-references have been generated, CRF deletes file.CRF.

D.2 THE CRF SYMBOL TABLE FILE

The CRF symbol table file is a series of contiguous 5-word data records preceded by a 9-word header record, as shown in Figure D-2. The symbol table file is assumed to reside on the same device and have the same name as the input listing file, except that the file has a filetype of CRF, that is:

```
filename.CRF
```

The header record contains the name of the originating task in two-word RADIX-50 format, followed by a number value assigned by the system to identify the task. This value allows CRF to locate the internal tables CRF uses to format output. The next five words define the creation date of the symbol table file. The last word contains the flags that define the output format. Bit zero of the flag word controls the width of the listing. When this bit is set to zero, the listing is generated in 132-column format. When this bit is set to one, the listing is generated in 80-column format.

The remaining records in the symbol table file are data records. The first two words of a data record comprise the symbol name in RADIX-50 format. The symbol value is an octal quantity associated with the symbol name. The second two words comprise the reference identifier, which is used as the cross-reference value. In the case of global symbols, this value is the module name in RADIX-50 format. In the case of MACRO-11 cross-references, the value is the page and line number of the reference. The last word of the data record comprises the attribute flag byte and the format number byte.

The attribute flag byte contains the bits describing the attributes of the reference. These flags cause the special characters and abbreviations to be displayed with the reference. A pound sign (#) means that the reference is the place where the symbol is defined. An asterisk (*) means that there is a destructive reference to the symbol (that is, its contents are changed). If there is nothing in front of the reference, it means that there is a nondestructive reference to the symbol (that is, its contents are read).

The format number byte defines the format of the output. The format number is used by CRF to locate a set of internal tables which define the listing format associated with the entry. The type of cross-reference and the set of symbols associated with the flags byte is determined by the format number. Use of the format byte permits several types of cross-reference output to be generated by a single originating task.

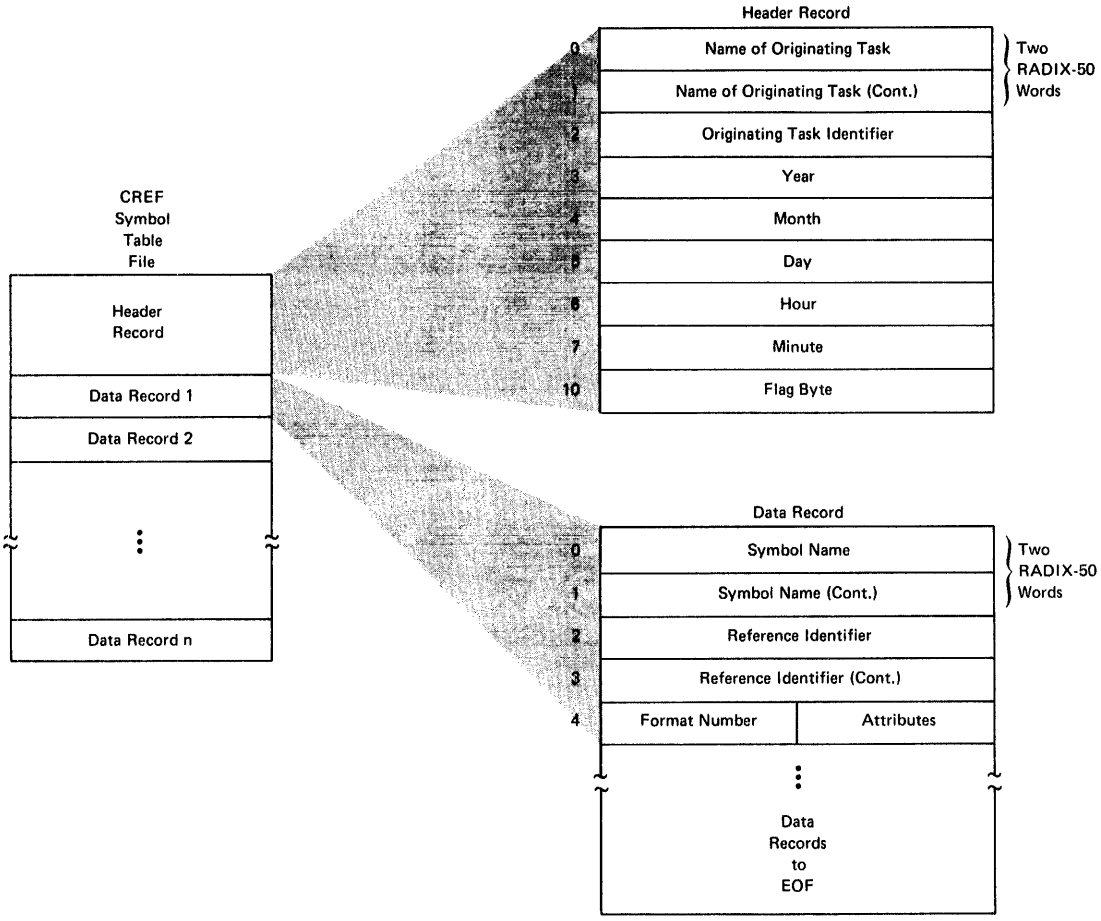


Figure D-2 Format of the CRF Symbol Table File

D.3 THE CRF SEND PACKET

The CRF SEND packet is a block of data that contains control information created by the Task Builder or MACRO-11 for use by CRF. This control information enables CRF to locate the symbol table file and the listing or memory allocation output file.

The SEND packet consists of 16 words, as described in Figure D-3.

Words 1 and 2 of the SEND packet contain the name of the sending task. Words 2 through 4 contain the filename of the output listing file in RADIX-50 format. Word 5 contains the filetype of the output listing file. Word 6 contains the version number of the output listing file. Words 7 through 11 contain the directory identifier. Word 12 contains the device name of the device on which the output listing file resides. The first byte of word 13 is a flag byte used by CRF for output processing. When this bit is set to 1, the output listing file is spooled offline when CRF completes processing. The second byte of word 13 specifies the unit on which the listing output file resides. Word 14 contains the symbol table file version number. Word 15 contains the device name of the output device. The first byte of word 16 is reserved. The second byte of word 16 is the unit number of the output device.

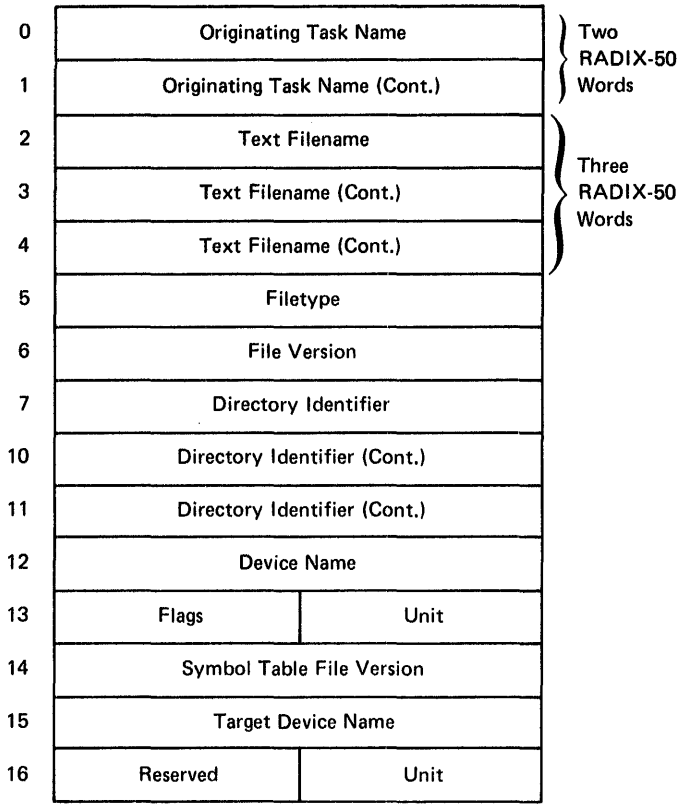


Figure D-3 Format of the CRF SEND Packet

D.4 CRF ERROR MESSAGES

The following error messages are output by the Cross Reference Task. Each message is preceded by one of the following prefixes:

```
CRF -- *DIAG*  - name of originating task - MESSAGE
CRF -- *FATAL* - name of originating task - MESSAGE
```

CRF INPUT FILE filename HAS ILLEGAL FORMAT

Explanation: This error is caused by a software error in the originating task. The symbol table file submitted for CRF processing contains no data.

User Action: Submit a Software Performance Report along with the related console dialogue and any other pertinent information.

FAILED TO DELETE FILE filename

Explanation: CRF was unable to delete the specified file for one of the following reasons:

- CRF did not have deletion privileges for the UFD under which the file resides.
- The device was not write-enabled or ready to perform I/O.

User Action: Initialize the device appropriately and ensure that the UFD has owner-delete privileges.

FILE filename NOT FOUND

Explanation: File filename could not be located; the file was probably deleted before CRF could process it.

User Action: Rerun the appropriate task to produce cross-reference output. Do not delete any files until CRF completes processing.

ILLEGAL ERROR/SEVERITY CODE data

Explanation: This error indicates a CRF software malfunction; CRF has called its error message processor with an illegal parameter.

User Action: Submit a Software Performance Report containing a copy of the message as printed. Correct any user errors and rerun the initiating task.

INPUT FROM UNKNOWN TASK

Explanation: Cross-reference processing requested by the originating task is not supported by CRF.

User Action: Delete the request for CRF processing and rerun the originating task.

I/O ERROR ON FILE filename

Explanation: An error has been encountered while reading or writing the specified file. A possible hardware problem is indicated; or the device may have insufficient space to accommodate the CRF output file.

User Action: Isolate the problem and take corrective action. Rerun MACRO or TKB to produce the cross-reference output.

INVALID OUTPUT FORMAT SPECIFIED

Explanation: This message indicates an inconsistency in the data file submitted for CRF processing.

User Action: Submit a Software Performance Report along with the related console dialogue and any other pertinent information.

NO DYNAMIC STORAGE AVAILABLE

Explanation: The Cross-Reference task requires more working storage than is available within the area of memory owned by the task.

User Action: If possible, install CRF in a larger partition or with a larger increment.

NO VIRTUAL MEMORY STORAGE AVAILABLE

Explanation: The Cross-Reference processor work file storage requirements exceed 65,536 words.

User Action: No recovery is possible from this error. If possible, install the task in a larger partition.

OPEN FAILURE ON FILE filename

Explanation: CRF was unable to open the named file for one of the following reasons:

- The device was not ready to perform I/O.
- The device was not write-enabled.
- A Files-11 device was not mounted.
- CRF did not have extend or delete privileges for the UFD under which the file resides.
- The file was deleted before it could be processed by CRF.

User Action: Isolate the problem, take appropriate corrective action, and rerun the task to obtain cross-reference output.

SYMBOL TABLE SEARCH STACK OVERFLOW

Explanation: This error is a CRF software error.

User Action: Submit a Software Performance Report along with the related console dialogue and any other pertinent information.

UNABLE TO OPEN WORKFILE

Explanation: Possible causes are:

- Workfile device not mounted.
- Workfile device write protected.

The workfile device is assigned to LUN 7 and is normally the device from which CRF was installed.

User Action: Retry the command after ensuring that the device is mounted and write-enabled.

WORK FILE I/O ERROR

Explanation: The Cross Reference Processor encountered an I/O error while reading or writing data on its workfile. Possible causes are:

- Device full.
- Hardware error.

User Action: If the device capacity has been exceeded, delete unnecessary files to make space available. Also, REASSIGN CRF LUN7 to another Files-11 device.

Switches and Subswitches

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

Please cut along this line.

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

RT/C SOFTWARE PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

Do Not Tear - Fold Here

Cut Along Dotted Line