RSX-11M/M-PLUS

Utilities Manual

Order No. AA-FD13A-TC

Includes Update Pages

digital

software

RSX-11M/M-PLUS

Utilities Manual

Order No. AA-FD13A-TC
Includes Update Pages

RSX-11M Version 4.2
RSX-11M-PLUS Version 3.0

digital equipment corporation · maynard, massachusetts

First Printing, December 1979
Revised, November 1981
Updated, April 1983
Revised, July 1985

DEC	DIBOL	PDT
DEC/CMS	EduSystem	RSTS
DEC/MMS	IAS	RSX
DECnet	MASSBUS	UNIBUS
DECsystem-10	MicroPDP-11	VAX
DECSYSTEM-20	Micro/RSTS	VMS
DECUS	Micro/RSX	VT
DECwriter	PDP	digital

ZK2945

HOW TO ORDER ADDITIONAL DOCUMENTATION

In Continental USA and Puerto Rico call 800-258-1710

In New Hampshire, Alaska, and Hawaii call 603-884-6660

In Canada call 613-234-7726 (Ottawa-Hull)
800-267-6215 (all other Canadian)

DIRECT MAIL ORDERS (USA & PUERTO RICO)*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire 03061

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575)

DIRECT MAIL ORDERS (CANADA)

Digital Equipment of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

DIRECT MAIL ORDERS (INTERNATIONAL)

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

CONTENTS

Page

Page

CONTENTS

Page

Page

TABLES

PREFACE

MANUAL OBJECTIVES

The _RSX-11M/M-PLUS Utilities Manual_ is a reference manual describing the use of 15 utilities supported 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 Line Text Editor (EDI).

Chapter 3 describes the Peripheral Interchange Program (PIP).

Chapter 4 describes the File Transfer Utility Program (FLX).

Chapter 5 describes the Disk Volume Formatter Utility (FMT).

Chapter 6 describes the Bad Block Locator Utility (BAD).

Chapter 7 describes the Backup and Restore Utility (BRU).

Chapter 8 describes the Disk Save and Compress Utility Program (DSC).

Chapter 9 describes the File Structure Verification Utility (VFY).

Chapter 10 describes the Librarian Utility Program (LBR).

Chapter 11 describes the File Dump Utility (DMP).

Chapter 12 describes the File Compare Utility (CMP).

Chapter 13 describes the Source Language Input Program (SLP).

Chapter 14 describes the Object Module Patch Utility (PAT).

Chapter 15 describes the Task/File Patch Program (ZAP).

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

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

ASSOCIATED DOCUMENTS

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

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

CONVENTIONS USED IN THIS DOCUMENT

Use of Red Ink

User input appears in red. Lines and prompting characters output by the system appear in black.

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 RET symbol, which denotes the RETURN key.

Use of Lowercase Characters

Lowercase characters in a command line indicate variables for which the user must substitute a word or value. For example:

```
     filename.filetype;version
```

This command indicates 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 Square Brackets ([])

Square brackets indicate optional entries in a command line or a file specification. Note that when an option is entered, the brackets are not included in the command line.

Square brackets also are a part of the User File Directory (UFD) and User Identification Code (UIC) syntax ([group,member]). When you use a UIC or UFD (in a file specification, for example), brackets are required syntax elements; that is, they do not indicate optional entries.

Use of Braces ({})

Braces indicate a choice of required entries for a command line. You can use any of the entries enclosed in the braces, but you must specify one of them.

Use of Commas (,)

Commas are used as separators for command line parameters and to indicate positional entries on a command line. Positional entries are those elements that must be in a certain place in the command line. Although you might omit elements that come before the desired element, the commas that separate them must still be included.

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 the file type is not specified, the period may be omitted from the file specification.

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 and subswitches in the command line. When shown in the command line format, they should be specified as shown.

. . .

A horizontal ellipsis indicates that the preceding item(s) can be repeated one or more times.

CTRL/X

The symbol CTRL/X indicates that you must press the key labeled CTRL while you simultaneously press another key, for example, CTRL/C , CTRL/O , CTRL/Z .

RET

Command lines are terminated by pressing the RETURN key unless otherwise indicated in the text. The form used to denote the RETURN key is RET .

Use of Shading

Shaded portions of text describe only one operating system. Pink shading indicates that the text describes only RSX-11M operating systems. Gray shading indicates that the text describes only RSX-11M-PLUS operating systems. The smallest shaded portion of text is a paragraph. The text that is not shaded describes both operating systems.

SUMMARY OF TECHNICAL CHANGES

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

Peripheral Interchange Program (PIP)

ANSI magnetic tapes are now supported.

& - Ampersand (new) - Separates each command when several commands are specified in the same command line.

% - Wildcard (new) - Denotes exactly one character in the file name and/or file type of an input file specification.

/CD - Creation Date switch (revised) - Can be set as the default (instead of /-CD).

/DD - Default Date switch (new) - Restricts file searches to files created during a specified period of time.

/DF - Default switch (revised) - Specified with no arguments, it returns the default device to SY0: and the UFD to the UIC from which PIP was invoked.

/EX - File Exclusion switch (new) - Excludes one file specification's worth of files during a file search.

/FR - Free Blocks switch (revised) - Displays the amount of available space on a specified volume, the largest contiguous space on that volume, and the number of available file headers.

/ID - Identity switch (revised) - Identifies the version of PIP being used and if PIP is linked to ANSI FCS.

/TD - Today switch (new) - Restricts file searches to files created on the current day.

File Transfer Utility Program (FLX)

The TU78 magnetic tape is now supported.

/DNS:6250 - Density switch (revised) - Specifies a density of 6250 bpi to support the TU78 magnetic tape.

Disk Volume Formatter Utility (FMT)

The RM05 disk is fully supported. FMT treats the RM05 as a larger RM03.

DL:-type devices are supported, and some corrupted DL:-type disk cartridges can be made usable again.

/VE - Verify switch (revised) - is now the default operation.

/-VE, /NOVE, /-VERIFY, /NOVERIFY - No Verify switch (new) - Inhibits the default verification operation.

/DENSITY, /MANUAL, /VERIFY (new) - Switch synonyms for /DENS, /MAN, and /VE respectively.

Bad Block Locator Utility (BAD)

A multiheader BADBLK.SYS file is now supported. The maximum allowable retrieval pointers in the bad block descriptor file have been expanded from 102 to 126 for non-last-track devices.

The updated stand-alone version of BAD supports the following devices: the RM05, RM80, and RP07 disk packs, and the RA80 fixed media disk.

/ALO:volume label - Allocate switch (new) - Prompts for blocks to be allocated to BADBLK.SYS and to be entered in the bad block descriptor file.

/PAT=m:n - Pattern switch (new) - Specifies the double-word data pattern used to locate bad blocks.

Backup and Restore Utility (BRU)

BRU now supports multivolume backup and restore operations using the /IMAGE switch.

BRU now supports the following devices: RA60/RA81/RC25/RD51/RX50 disks, RL11/RL02 cartridge disks, RH70/RM05/RM80/RP07 disk packs, UDA50/RP07 disk packs, TM78/TU78 magnetic tapes, ML11 electronic memory, and TS11/TSV05/TU80 magnetic tapes.

Disk volumes with multiheader index files (structure level 402) are supported.

There are now two stand-alone BRU systems. BRU64K is the stand-alone version for the RSX-11M and BRUSYS is the stand-alone version for RSX-11M-PLUS.

/DENSITY - Density switch (revised) - For TU78 magnetic tapes, you can specify a density of 1600 or 6250 bpi (6250 is the default bpi).

/IMAGE:option - Image switch (new) - Specifies that you want to do a multiple disk-to-disk backup or restore operation.

/UFD - User File Directory switch (new) - Directs BRU to create a UFD (if it does not already exist) on a mounted output volume, and then to copy into it the files from the same UFD on the input volume.

Disk Save and Compress Utility Program (DSC)

There are two new stand-alone versions of DSC: DSCSYS and DSC64K.

DSCSYS is a combination of DSCS8 and DSCS16. This version requires 28K words to run and a maximum blocking factor of 4. DSCSYS supports the following new devices: the RP07 disk, the RA08 fixed media disk, and the TU78 magnetic tape.

DSC64K is essentially an RSX-11M system with BAD, FMT, DSC, and CNF fixed in memory. The maximum blocking factor is 4 and DT, DX, DY, DD, DF, and DS devices are not supported.

DSC now supports up to 64K files on a volume.

/BAD=OVR - Override option (new) - Ignores the bad block descriptor area and accesses the last good block on the next to last track of the disk to obtain the data to create BADBLK.SYS.

/BAD=MAN:OVR - Manual Override option (new) - Allows manual entry of bad block data to the bad block file BADBLK.SYS.

/DENS=6250 - Density switch (revised) - Creates magnetic tapes at 6250 bpi. The /DENS:6250 option is valid with TU78 magnetic tapes only.

File Structure Verification Utility Program (VFY)

/DV - Directory Validation switch (new) - Validates directories against the files they list.

/HD - Header Delete switch (new) - Recognizes and deletes bad file headers on a volume. The /AL subswitch allows bad headers to be automatically deleted with no user intervention.

File Dump Utility Program (DMP)

Multiple-format dumps are supported. Any or all of the format switches can be specified in a command line.

ANSI magnetic tapes are supported.

/LC - Lower Case switch (new) - Specifies that the data should be dumped in lowercase characters.

/OCT - Octal switch (new) - Specifies that the data should be dumped in octal format in addition to other formats.

/SB:n or /SB:-n - Specifies the number of blocks DMP space Blocks switch (new) - Spaces forward or backward on a tape.

/SF:n or /SF:-n - Space Files switch (new) - Specifies the number of end-of-file (EOF) marks DMP spaces forward or backward on a tape.

File Compare Utility (CMP)

CMP's default output device/file is now TI:.

Any unspecified portions of the second input file specification default to the specifications for the first input file.

Source Language Input Program (SLP)

/SQ - Sequence switch (new) - Sequences the lines in the output file so that they reflect the line numbers of the original input file.

/RS - Resequence switch (new) - Resequences the lines in the output file.

/NS - No Sequence switch (new) - Does not sequence lines in the output file. New lines are indicated by the audit trail.

Task/File Patch Program (ZAP)

Four types of task image files are supported:

- Regular task image files (including those mapped to resident and supervisor mode libraries)
- Multiuser task image files (RSX-11M-PLUS only)
- I- and D-space (instruction and data space) tasks (RSX-11M-PLUS only)
- Resident libraries

Resident libraries can be changed in task image mode. ZAP finds the segments and allows you to make changes in both absolute mode and task image mode.

/LI - List switch (revised) - Includes read-only segments of a task in its segment table. ZAP finds the starting address and allows you to make changes in both absolute mode and task image mode. For I- and D-space tasks, the /LI switch lists the starting block number and the address boundaries of each segment.

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-11M/M-PLUS 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. Two appendixes are also included to provide you with a summary of commands and switches for 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-11M/M-PLUS utilities, this introduction:

- Describes how to enter RSX-11M/M-PLUS command lines and file specifications (Sections 1.2 and 1.3)
- Describes how to invoke utilities and enter command lines to them (Section 1.4)
- Describes how to use indirect command files (Section 1.5)

1.1 RSX-11M/M-PLUS UTILITY PROGRAMS

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

Line Text Editor (EDI)

Peripheral Interchange Program (PIP)

File Transfer Utility Program (FLX)

Disk Volume Formatter Utility (FMT)

Bad Block Locator Utility (BAD)

Backup and Restore Utility (BRU)

Disk Save and Compress Utility Program (DSC)

File Structure Verification Utility (VFY)

Librarian Utility Program (LBR)

File Dump Utility (DMP)

File Compare Utility (CMP)

Source Language Input Program (SLP)

Object Module Patch Utility (PAT)

Task/File Patch Program (ZAP)

The following sections briefly describe each utility.

Note that the utilities described in this manual are not the only programs on RSX-11M and RSX-11M-PLUS that are used as or considered to be utilities. TKB, CDA, and MAC are examples of other utility-like programs. Some programs, such as the editor EDT, are common across different operating systems. These other programs are documented elsewhere in the RSX-11M/M-PLUS documentation set. Refer to the RSX-11M/RSX-11S or RSX-11M-PLUS Information Directory and Index for information on what programs are available and where they are described.

1.1.1 Line Text Editor (EDI)

EDI is a line-oriented, interactive editor used to create and maintain text and source files. (The RSX-11M/M-PLUS Guide to Program Development gives specific information about using EDI to create and maintain program source files.)

1.1.2 Peripheral Interchange Program (PIP)

PIP copies files and performs several file control functions, such as concatenating, renaming, spooling, listing, deleting, and unlocking.

1.1.3 File Transfer Utility 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.

1.1.4 Disk Volume Formatter Utility (FMT)

FMT formats and verifies several types of Files-11 disks. FMT writes and verifies sector headers, sets the density for flexible disks, and allows spawning of the Bad Block Locator Utility (if your system allows spawned tasks).

1.1.5 Bad Block Locator Utility (BAD)

BAD determines the number and location of bad blocks on a volume. The information gathered from running BAD on a volume can be used in different ways when that volume is initialized.

1.1.6 Backup and Restore Utility (BRU)

BRU transfers files from a Files-11 volume to one or more backup volumes (including non-Files-11 volumes) and retrieves files from the backup volume (or volumes). BRU is faster than DSC (see Section 1.1.7) in most areas. Also, BRU compresses data, the volumes do not have to be initialized, and incremental backups are possible.

1.1.7 Disk Save and Compress Utility 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 on-line or stand-alone.

1.1.8 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.

1.1.9 Librarian Utility Program (LBR)

LBR is a library maintenance program that creates, displays, and modifies library files. LBR can process macro, object, and universal libraries.

1.1.10 File Dump Utility (DMP)

DMP is a file listing program that allows you to examine the contents of a file or volume of files. DMP also provides options that control the format of the contents.

1.1.11 File Compare Utility (CMP)

CMP compares two text files, record by record, and lists the differences between the two files.

1.1.12 Source Language Input Program (SLP)

SLP is a noninteractive editing program that is used to maintain and audit source files.

1.1.13 Object Module Patch Utility (PAT)

PAT updates, or patches, relocatable binary object modules.

1.1.14 Task/File Patch Program (ZAP)

ZAP is a patch utility that examines and directly modifies locations in a task image file or data file.

1.2 COMMAND LINES

The general format for command lines in most of the RSX-11M/M-PLUS utilities is:

```
outfile[...,outfile]=infile[...,infile] RET
```

The variables outfile and infile are file specifications for the output and input files to be operated on by the utility. (File specifications are described in Section 1.3.)

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. Most of the utilities also accept indirect command files containing command lines to the utility, as described in Section 1.5.

1.3 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 default length is 80 characters).

The format for entering file specifications is:

```
ddnn:[g,m]filename.type;version/switch.../subswitch...
```

ddnn:

The physical or logical device unit containing the desired volume. The name consists of two or three ASCII characters followed by an optional 1-, 2-, or 3-digit octal number and a colon, for example, DM0:, or TT116:.

The default is the user's system device, SY:.

[g,m]

The User File Directory (UFD) listing the desired file or files. The variables g and m are octal numbers from 0 to 377 that represent the group and member numbers, respectively, of the file's owner. The brackets are a mandatory part of the UFD syntax.

The default is the current UIC (User Identification Code) to which your terminal is set.

See the RSX-11M/M-PLUS MCR Operations Manual or the RSX-11M/M-PLUS Command Language Manual for more information on UICs and UFDs.

filename

The name of the file. File names can be from one to nine characters in length.

If you want to include special characters in the file name (for example, semicolons or exclamation marks), place double quotation marks around the name. If you use only one double quotation mark, MCR or DCL assumes an American National Standard X.327-1978 file name. (See the RSX-11M/M-PLUS I/O Operations Manual for more information on double quote support.) Note that not all utilities or other tasks allow special characters.

There is no default.

type

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. In this way, the file type identifies the nature of the contents of the file.

File type and file name are separated by a period. The file type may not be specified or can be up to three alphanumeric characters in length. The default for a file type depends on the utility or task you are working with and if the file is an input or output file.

See the RSX-11M/M-PLUS MCR Operations Manual or the RSX-11M/M-PLUS Command Language Manual for a list of standard file types.

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 range from 1 through 77777(8). However, you can also use 0 to specify the highest numbered version and -1 to specify the lowest numbered version (0 is the default). If a file has a version number of 77777, no more versions of it can be created.

Version number and file type are separated by a semicolon. The default is the latest version.

/switch (also /qualifier)

An ASCII name specifying a switch (or qualifier in BRU) 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 also take values in the form of ASCII strings and numeric strings. The values modify the function of the switch.

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 specification 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 examples of valid switch specifications:

```
/SW:27.:MAP:29.

/-SW

/NOSW:-#50:SWITCH
```

/subswitch

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 RET
```

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 THE UTILITIES

You invoke a utility from the command line interpreter (CLI) environment. The CLI can be the Monitor Console Routine (MCR), the DIGITAL Command Language (DCL), or an alternate user-written CLI. For more information on MCR, see the _RSX-11M/M-PLUS MCR Operations Manual_. For more information on DCL, see the _RSX-11M/M-PLUS Command Language Manual_.

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

You can work with a utility directly (interactively) 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 describes these methods. For systems in which not all utilities are installed, you can use the method described in Section 1.4.2.

Section 1.5 describes how to invoke a utility that can then accept commands from an indirect command file.

You can invoke a utility when MCR or DCL prompts you. The MCR prompts are:

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

The DCL prompts are:

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

In MCR, the utilities are always invoked by their 3-character names. DCL, however, has commands that access utilities transparently to the user. You do not have to explicitly specify the utility 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 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 command MCR (or MC).

1.4.1 Invoking Installed Utilities

RSX-11M/M-PLUS systems provided in distribution kits do not have any utilities installed. Once the system has been generated, the system manager usually installs any commonly used utilities. Use the MCR TAS or DCL SHOW TASKS /INSTALL commands to see which utilities are currently installed in the system. If the utility you want to use is not installed, any privileged user can install it with the MCR or DCL INSTALL command. Once the utility is installed, you can invoke it.

The following sections describe the three primary methods you can use to invoke installed utilities.

1.4.1.1 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 command-line
```

or

```
MCR>utilityname command-line
```

Using this method to invoke the utility allows you to enter a single command for execution. The utility is installed, the command is executed, and control returns to MCR. (The method described in the following section 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 1.4.1.3. (However, you can specify

```
>SLP @indirectcommandfile
```

or

```
>ZAP @indirectcommandfile
```

See Section 1.5 for more information.)

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

```
>commandname command-line
```

or

```
>MCR utilityname command-line
```

Using these methods to invoke the utility allow you to enter a single command for execution. The utility is installed, the command is executed, and control returns to DCL. With the first method, the DCL command transparently accesses the utility (see Section 1.4).

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 1.4.1.3. (However, you can specify

```
>SLP @indirectcommandfile
```

or

```
>ZAP @indirectcommandfile
```

See Section 1.5 for more information.)

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

For MCR:

```
>utilityname
```

For DCL:

```
>MCR utilityname
```

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 can then enter the command line 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.

1.4.2 Invoking Uninstalled Utilities

You can use the following method to invoke an uninstalled utility. This method is useful for smaller systems in which not all utilities are installed. This method uses either the MCR RUN command or the DCL RUN command to invoke the utility.

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

```
>RUN $utilityname
```

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 line is executed. To terminate the utility, enter CTRL/Z. Control is then returned to MCR or DCL.

A variation of this method allows the utility to run under a UIC other than the current UIC:

For MCR:

```
>RUN $utilityname/UIC=[g,m]
```

For DCL:

```
>RUN/UIC:[g,m] $utilityname
```

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

1.5 USING INDIRECT COMMAND FILES

An indirect command file normally contains a sequence of command lines that are interpreted by a 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 command lines contained in the indirect command file are executed when the indirect command file is invoked. If you invoke the file from MCR or DCL, each command line must begin with the name of the utility or command you want to use. If you invoke the file from the utility itself, the command lines do not begin with the name of the utility, but they must all be legal for that utility.

For example, an indirect command file might contain the following series of PIP command lines:

```
=DB2:[303,24]TESTPREP.CMD
TESTPREP.*;*/LI
TESTPREP.*/PU:2
*.CMD/SP
```

To invoke the indirect command file (PIPCMDS.CMD), enter one of the following sets of commands:

For MCR:

```
>PIP @PIPCMDS.CMD(RET)

or

>PIP(RET)
PIP>@PIPCMDS(RET)

or

>RUN $PIP(RET)
PIP>@PIPCMDS(RET)
```

For DCL:

```
>RUN $PIP(RET)
PIP>@PIPCMDS(RET)

or

>MCR PIP @PIPCMDS(RET)

or

>MCR PIP(RET)
PIP>@PIPCMDS(RET)
```

In this example, PIP is invoked and accesses the file PIPCMDS.CMD, which contains the sequence of PIP command lines. Because PIP is invoked first, the command lines in the file do not have to begin with PIP. PIP executes the command lines and returns control to MCR, DCL, or PIP, depending on which command set you use.

RSX-11M and RSX-11M-PLUS also allow you to use indirect command files that contain MCR or DCL commands. The command lines do not begin with MCR or DCL; they must only be legal for the CLI. You invoke the indirect command file by entering only the file specification preceded by the at sign (@) in response to the prompt (in this case, the MCR prompt):

```
>@indirectcommandfile
```

The default values for indirect command file specifications are:

- device - SY:
- [g,m] - 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 indirect command files, see the _RSX-11M/M-PLUS MCR Operations Manual_.

CHAPTER 2

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-11 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, and 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 and output commands** transfer text to and from input, output, and saved files.
- **Device output commands** send output to a terminal or a printer.
- **Close and exit commands** terminate editing operations.

Commands are categorized in this chapter as Basic EDI Commands (Section 2.2), EDI Commands: Function Summary (Section 2.3), and EDI Commands: Detailed Reference Summary (Section 2.4). Restrictions, system device considerations, and error messages for these commands are discussed in Sections 2.5 and 2.6.

2.1 USING EDI

This section gives background information about EDI that is important for you to know before you read the command descriptions.

2.1.1 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 prompt:

```
EDI>
```

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

2.1.1.1 **Entering File Specifications** - Enter a file specification in the following format:

```
ddnn:[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 2.1.2.)

If the message FILE DOES NOT EXIST is printed, 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 (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, edit mode if the file name already exists.

After EDI has identified the input file and created the output file, it is ready for commands. In edit mode, the first line available for editing is one line above the first line 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 issue a NEXT command to make that line available.

2.1.1.2 **Defaults in File Specifications** - EDI uses a default if any of the elements of the file specification, except the input file name, 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 editing 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 2-1.

Table 2-1
EDI Default File Specifications

Element	Default Value for Input File	Default Value for Output File
ddnn:	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 file name
filetype	Unspecified	Same as input file type
;version	Latest version	Latest version + 1

2.1.2 Control Modes: Edit and Input

EDI runs in two control modes:

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

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

In edit mode, EDI issues an asterisk (*) as a prompt. EDI acts upon commands and data to open and close files; to bring 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 by pressing the RETURN key, EDI sends a line feed to the terminal.

To switch from edit mode to input mode, enter the INSERT command and press the RETURN key. To return to edit mode, press the RETURN key as the only character on an input line. EDI will issue the asterisk prompt, signifying edit mode.

2.1.3 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 pressing the RETURN key.) 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 "pages." This feature is described in Section 2.1.3.3.

2.1.3.1 **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 discussed later in this chapter. 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 line, that line is no longer accessible unless you enter a TOF or TOP command (described in Section 2.4). TOF causes the input and output files to be closed, and the output file to become the new input file. TOF also ends line-by-line mode.

2.1.3.2 **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 with 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 2-2 briefly summarizes the differences between line-by-line and block mode. Regardless of the editing mode, the line pointer always points to the first character in the line.

Table 2-2
Line-by-Line and Block Mode Differences

Line-by-Line Mode	Block Mode
One line is available for modification at a time.	A block of lines is 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 search the entire file.	Search commands can search only the block in memory. To search more data, you must read in another block.

2.1.3.3 **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 command, 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 (EOF), EDI increments the page count to 1.

2.1.4 Text Files

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

2.1.4.1 **Input and Secondary Files** - EDI accepts input from the following:

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

2.1.4.2 **Output Files** - The output file defaults to the input file device, directory file name, and file type specifications. The version number is incremented by one.

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

2.1.5 Terminal Conventions

RSX-11M/M-PLUS and EDI provide terminal keyboard functions that allow you to:

- Delete characters on an input line
- Delete an entire input line
- Move the current line pointer forward in a file
- Move the current line pointer backward in a file
- Terminate an editing session and return control to your CLI (for example, MCR or DCL)

2.1.5.1 **Character Erase (DELETE or RUBOUT; CTRL/R)** - Pressing the DELETE key (marked RUBOUT on some terminals) deletes individual characters if used before the RETURN key is pressed. During editing operations, DELETE does not affect previously prepared text.

1. To delete the input file, use the CLOSE-AND-DELETE command or the EXIT-AND-DELETE command, or use PIP (see Chapter 3).

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

First DELETE typed	MISTKAE\E
Second DELETE	MISTKAE\EA
Third DELETE	MISTKAE\EAK
First non-DELETE	MISTKAE\EAK\AKE

For some CRT terminals, DELETE (or RUBOUT) works in a more obvious way. Each DELETE 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. It is typed by holding down the CTRL key and pressing R. CTRL/R echoes ^R and is followed by a return and line feed. For example, at a hardcopy terminal you enter:

```
     MISTKAE DEL  DEL  DEL  CTRL/R
```

The echoed result is:

```
     MISTKAE\EAK ^R
     MIST
```

2.1.5.2 **Line Erase (CTRL/U)** - CTRL/U deletes the line being input, if typed before the line is terminated with the RETURN key. It is typed by holding down the CTRL key and pressing U. CTRL/U echoes as ^U and is followed by a return and line feed.

2.1.5.3 **The RETURN Key** - The RETURN key has the following effects, depending on how it is used:

- When issued in place of an input file specification, the RETURN key causes EDI to terminate.
- When issued in edit mode, the RETURN key 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, the RETURN key causes a return to edit mode.
- When issued alone after an INSERT command, the RETURN key invokes input mode.

2.1.5.4 **Terminating the Previous Text Line (ESCape or ALTmode)** - When EDI is in edit mode, pressing the ESCape (or ALTmode) key 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.

When EDI is in input mode, ESC acts as a return and terminates the line. If ESC is the first character of an input line, EDI exits from input mode.

2.1.5.5 **Terminating EDI (CTRL/Z)** - CTRL/Z causes EDI to terminate. EDI writes the remainder of the input file into the output file and then closes both files before terminating. Use CTRL/Z to terminate EDI in edit mode and input mode. CTRL/Z erases your last input line if you enter the command as a line terminator.

2.1.6 EDI Command Conventions

EDI uses asterisks (*) and ellipses (...) in special ways. The following sections describe these and also the notation convention used to define EDI command abbreviations.

2.1.6.1 **Use of Asterisk (*)** - The asterisk (*) can be used in place of any numeric argument. It evaluates to 32767(10).

Example

The following command results in the printing of the remainder of the block buffer or file.

```
PRINT *
```

2.1.6.2 **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, you can use 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. 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/HERE IS AN EXAMPLE OF SPEC

results in

HERE IS AN EXAMPLE OF SPECIAL STRING CONSTRUCTS.

Case 3. C /STRING.../EDI STRING CONSTRUCTS.

results in

HERE IS A SAMPLE OF SPECIAL EDI STRING CONSTRUCTS.

Case 4. C /.../EXAMPLES OF SPECIAL EDI CONSTRUCTS.

results in

EXAMPLES OF SPECIAL EDI CONSTRUCTS.

2.1.6.3 **Command Abbreviations** - EDI permits the use of truncated commands. Where these shorter forms are allowed, the command format specifications represent the shortest acceptable form in uppercase letters. The lowercase letters may be entered optionally. The following example shows the abbreviations allowed for the VERIFY command. The command format specification is:

```
Verify
```

The following truncations are valid for the VERIFY command:

```
V
Ve
Ver
Veri
Verif
Verify
```

2.2 BASIC EDI COMMANDS

The basic EDI commands listed in Table 2-3 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. A more detailed description of each command follows the table. These commands are the most important EDI commands. As you become familiar with EDI operations, the additional commands listed in Section 2.3 and described in Section 2.4 will allow you to use all of EDI's capabilities.

Table 2-3
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 resulting line.
BOTTOM	BOttom	Move the current line pointer to the bottom of the current block (in block mode) or 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 positive; delete n lines preceding current line if n is negative. [-n] operates in block mode only.
DELETE & PRINT	DP [n] or DP [-n]	Same as DELETE, except new current line is printed.
ESC	Type the ESC (or ALT) key	Print previous line, make it new current line and exit from input mode (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.
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.

(continued on next page)

Table 2-3 (Cont.)
Basic EDI Commands

Command	Command Format	Description
RENEW	RENew[n]	Write current block to output file and read new block n from input file (block mode only).
RET	Press the RETURN key	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 not given.
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.

2.2.1 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 SP SP AGAIN.
```

2.2.2 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 used for the ADD command, the following command causes the new line to be printed as follows:

```
*AP AGAIN.
HAPPY DAYS ARE HERE AGAIN.
```

2.2.3 BOTTOM

This command 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 2.4.14).

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 printed.

2.2.4 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 asterisk prompt. If string1 is not given, string2 is inserted at the beginning of the line. If string2 is not given, 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 following example.

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.
```

2.2.5 **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.

2.2.6 **DELETE**

This command causes lines of text to be deleted in the following manner:

- 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.
- 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.
- If n is not given, 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
```

2.2.7 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 it:

```
*DP 2
THIS IS LINE 3
```

2.2.8 The ESCape Key

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 pressing a series of ESCapes. Pressing ESCape is equivalent to typing NP-1 (NEXT & PRINT command).

2.2.9 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 file name.

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. The version number is one greater than that of the input file.

2.2.10 INSERT

This command inserts a string immediately following the current line. The string becomes the new current line. If a 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 RETURN causes EDI to switch to input mode. A series of new lines may be input following the current line.

`*`	A RETURN or ESC as the only character in an input line causes EDI to return to edit mode and to prompt for a new command.

2.2.11 LOCATE

This command causes a search for a 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.

If a 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.

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.

2.2.12 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 for n can be used only in the block mode.

Example

In block mode, the following command moves the current line pointer back five lines:

```
*N -5
```

2.2.13 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 an NP command:

- Pressing the 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
* (RET)
LINE 4 OF THE FILE
* (ESC)
LINE 3 OF THE FILE
* (ESC)
LINE 2 OF THE FILE
```

2.2.14 PRINT

This command prints 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.

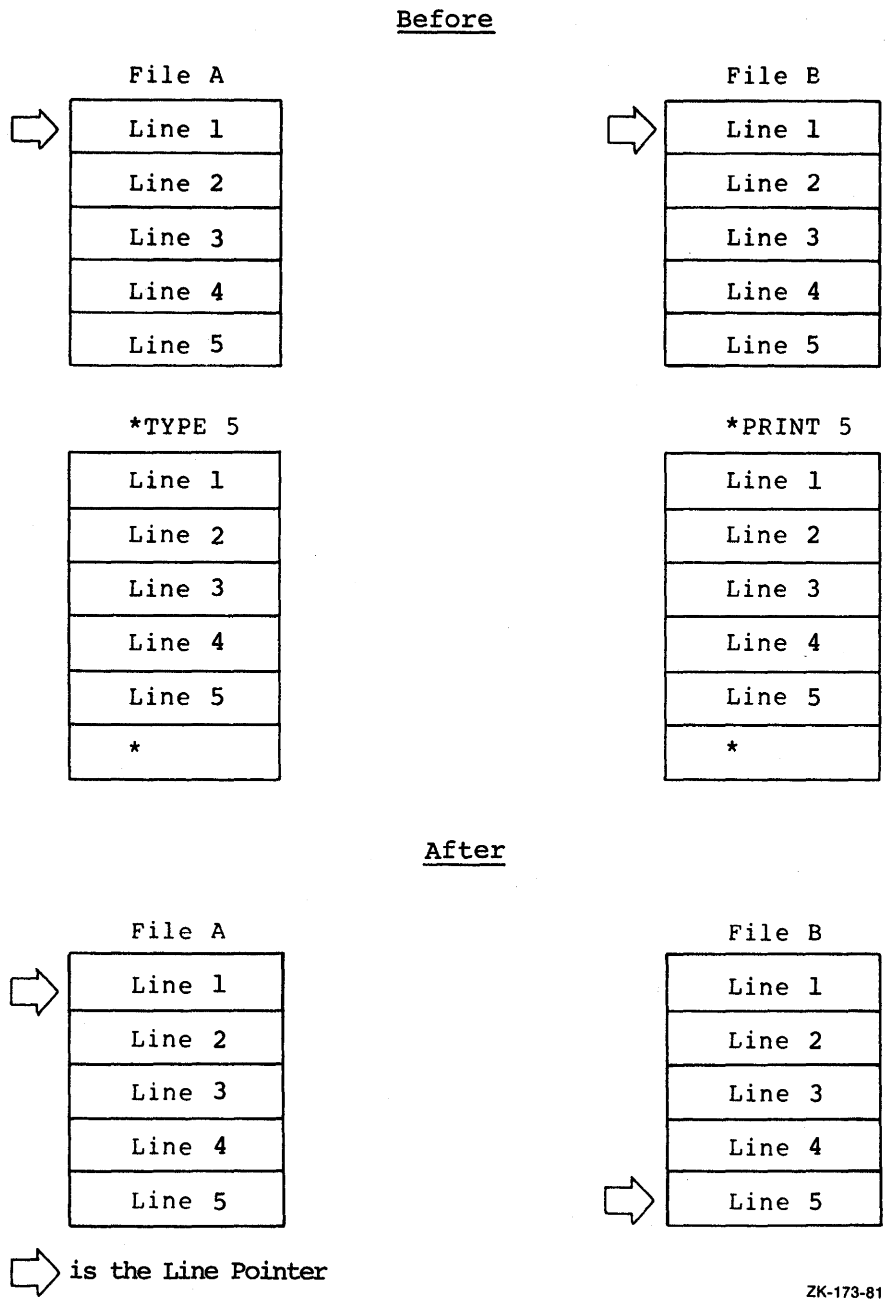
Format

```
Print [n]
```

Example

Example 2-1 illustrates both the PRINT and the TYPE commands:

Example 2-1 Line Pointer Position for the TYPE Command and the PRINT Command



2.2.15 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 process is performed. This command may be used only in block mode. Refer to Section 2.1.3 for information on how EDI block buffers are processed.

Format

```
RENew [n]
```

Example

```
*RENEW 10
```

Ten blocks are transferred consecutively from the input file to the block buffer. The initial contents of the block buffer and the next nine blocks are transferred consecutively to the output file. The current line pointer points to the first line in the tenth block, which is currently in the block buffer.

2.2.16 The RETURN Key

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 pressing a series of RETURNs. This command is equivalent to NP+1 (NEXT & PRINT command).

In input mode, a RETURN causes EDI to return from input mode to edit mode.

2.2.17 RETYPE

This command replaces the current line with a string. If a 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.

2.2.18 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.

2.2.19 TOP

This command 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 with 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.

2.3 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, and 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 a terminal or line printer.
- **Close and exit commands** terminate editing operations.

2.3.1 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, and the automatic verification of LOCATE commands. Setup commands are listed in Table 2-4.

Table 2-4
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 lowercase characters entered from terminal to uppercase characters.
VERIFY ON/OFF	Verify [ON] or Verify OFF	Select whether locator and change commands are verified.

2.3.2 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. These commands are listed in Table 2-5.

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 RETURN and ESCape (or ALTmode) keys act to relocate the line pointer. A RETURN moves the pointer to the next line. An 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 2-5
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.
ESC (or ALTmode)	Press ESC (or ALT) key	Print previous line, make it new current line, or exit from input mode (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.

(continued on next page)

Table 2-5 (Cont.)
EDI Locator Commands

Command	Format	Description
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.
RET	Press RETURN Key	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.

2.3.3 Text Modification and Manipulation Commands

The text modification and manipulation commands enable you to modify text. Table 2-6 lists these commands.

Table 2-6
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 resulting line.
CHANGE	[n]Change/string1/ string2[/]	Replace string1 with string2 in the current line n times.

(continued on next page)

Table 2-6 (Cont.)
EDI Text Modification and Manipulation Commands

Command	Format	Description
DELETE	Delete [n] or Delete [-n]	Delete current line and n-1 lines if n is positive; delete n lines preceding current line if n is negative. [-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).
PASTE	PAste/string 1/ string2[/]	Search all remaining lines in file or block buffer for string1 and replace with string2.
RETYPE	Retype[string]	Replace the current line with string or delete the current line if string is not given.
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 given, SAVE.TMP is used.

2.3.4 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. Table 2-7 lists the macro commands.

Table 2-7
EDI Macro Commands

Command	Format	Description
MACRO	MACRO x definition	Define macro number x. The value x may be 1, 2, or 3.
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].
MACRO IMMEDIATE	[n] <definition>	Define and execute a macro n times. Store it as macro number 1.

2.3.5 File Input/Output Commands

Input/output commands control the movement of text to and from input/output files, and save files. Table 2-8 lists these commands.

Table 2-8
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.

(continued on next page)

Table 2-8 (Cont.)
EDI Input/Output Commands

Command	Format	Description
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 on the device for 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.

2.3.6 Device Output Commands

These commands direct output to your terminal or to a pseudo device (CL:). They are listed in Table 2-9.

Table 2-9
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 the 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.

2.3.7 CLOSE and EXIT Commands

The CLOSE and EXIT commands terminate EDI operations and write the remainder of the input file into output file. Table 2-10 lists these commands.

Table 2-10
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 EDI.
EXIT	EXit [filespec]	Close files, rename output file, and terminate EDI.
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.

2.4 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.

2.4.1 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 2.2.1.

2.4.2 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 2.2.2.

2.4.3 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).

2.4.4 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.

2.4.5 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 2.2.3.

2.4.6 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 2.2.4.

2.4.7 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 file spec. 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.

2.4.8 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
```

2.4.9 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 file spec. 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

```
CD1 [filespec]
```

2.4.10 CONCATENATION CHARACTER (CC)

The concatenation character allows you to give commands on one input line. By default, the concatenation character is the ampersand (&). To reference text containing an ampersand (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 the ampersand.

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 the ampersand to the colon.

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 colon.)

The third command line changes the concatenation character back to the normal default value, &.

2.4.11 CTRL/Z

CTRL/Z is a Command Line Interpreter (CLI) function that terminates EDI. For usage information on CTRL/Z, refer to Section 2.2.5.

2.4.12 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 2.2.6.

2.4.13 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 2.2.7.

2.4.14 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 you issued 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).

2.4.15 **ERASE**

In line-by-line mode, this command erases the current line. In this mode, n can 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.

2.4.16 **The ESCape Key**

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 pressing a series of ESCapes. Pressing ESCape is equivalent to typing NP-1 (NEXT & PRINT command).

If EDI is in input mode, ESC acts like RETURN and terminates a line of input. ESC also exits from input mode if it is the first character of the line.

2.4.17 **EXIT**

EXIT writes all remaining records to the output file, closes the files, and terminates EDI.

Format

```
EXIT [filespec]
```

For examples and information on how to use EXIT, refer to Section 2.2.9.

2.4.18 **EXIT & DELETE (ED)**

This command functions in the same way as the CLOSE & DELETE command except that EDI also terminates.

Format

```
     EDx [filespec]
```

Example

```
        *EDX NEWFILE.DOC
        [EXIT]
        >
```

2.4.19 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 (that is, 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.

2.4.20 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.

2.4.21 FORM FEED (FF)

This command allows you to insert form feeds into the text to delimit pages. The form feed is inserted after the current line. The line containing the form feed then becomes the new current line.

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.

2.4.22 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 2.2.10.

2.4.23 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 your CLI by means of CTRL/Z or enter a file specification for a file to be edited.

2.4.24 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 not given, 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 the asterisk prompt.

Format

```
[n]LC /string1/string2[/]
```

Example

If the current line is:

```
THES ES THE LINE TO BE ESSUED.
```

The following commands would correct the errors:

```
*V ON
*LC /ES/IS
THIS IS THE LINE TO BE ISSUED
```

2.4.25 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.

2.4.26 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:.

2.4.27 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 2.2.11.

2.4.28 MACRO

This command is used to define macros. Space is available for three macro definitions. The definition 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 with 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 occurrences 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)
```

2.4.29 **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, and so on.

The file used to store the macro definitions must be the latest version of file MCALL -- that is, MCALL;n. The file type must be null or blank. If the macro definitions to be loaded are in a file of another name, you can use PIP with the /NV switch, to rename the file. (Refer to Chapter 3 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.

2.4.30 MACRO EXECUTE

This command executes a macro n times while passing it an optional numeric argument a. If a macro numeric argument is defined with 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 by means of a MACRO command or called with a MACRO CALL command.

Use the MACRO EXECUTE command to execute any one of the three macro definitions stored in the EDI macro storage area 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 the 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 2.6.4 illustrates how to use the EDI macro commands in editing a file.

2.4.31 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 2.6.3 illustrates the use of the MACRO IMMEDIATE command.

2.4.32 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 2.2.12.

2.4.33 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 2.2.13.

2.4.34 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 by means of 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, then the first block is read in.

2.4.35 **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 not 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.

2.4.36 **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.

2.4.37 PAGE

This command causes EDI to enter block mode, if not already in it, 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

```
PAGe n
```

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 (that is, 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.

2.4.38 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 can be used only in block mode.

Format

```
[n]PFind string
```

2.4.39 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 can 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 the string.

2.4.40 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, EDI returns the asterisk (*) 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.

2.4.41 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 2.2.14.

2.4.42 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 cannot 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.

2.4.43 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 2.2.15.

2.4.44 The RETURN Key

In edit mode, the RETURN key represents the return that displays the next line in the file or block buffer. In input mode, entering the RETURN returns EDI to edit mode. For information on EDI command modes, refer to Section 2.1.2. For information on the RETURN key, refer to Section 2.2.16.

2.4.45 RETYPE

RETYPE replaces the current line with string.

Format

```
Retype [string]
```

For information on how to use RETYPE, refer to Section 2.2.17.

2.4.46 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 on the device for 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. It remains on the specified volume until deleted.

2.4.47 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 the asterisk (*) 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 commands 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.

2.4.48 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.

2.4.49 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 Section 2.4.28.

2.4.50 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 depends 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 the discussion of block mode in Section 2.1.3.)

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.

2.4.51 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
```

2.4.52 **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 by means of 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 2.2.19.

2.4.53 **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 2.2.18.

2.4.54 **TYPE**

This command is similar to the PRINT command (Section 2.2.14). In line-by-line mode, the two are identical. In block mode, TYPE does not move the line pointer after displaying the requested text unless 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 2.2.14).

2.4.55 UNSAVE

This command retrieves all the lines in a specified file and copies them after the current line. If no file is specified, the default file is 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 2.6.2 contains an example using the SAVE and UNSAVE commands.

2.4.56 UPPER CASE ON/OFF

This command allows you to enter lowercase characters from a terminal and have them converted to uppercase characters. If UPPER CASE OFF is issued, all input characters are accepted as they are entered, including the EDI commands.

Format

```
UC [ON]
  or
UC OFF
```

If neither ON nor OFF is specified, then ON is assumed.

Example

```
*UC OFF
*I this line is entered in lowercase
*UC ON
*I this line is converted to uppercase
```

Assuming that the input terminal is capable of generating lowercase input, the commands in the example 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 uppercase before the file is closed.

To create a file containing lowercase characters, use the MCR SET /LOWER=TI: or the DCL SET TERM LOWER command and the EDI UC OFF command.

2.4.57 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. VERIFY ON is the default when EDI is started.

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.

2.4.58 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.

2.5 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:
  - Transfer the file to disk and perform the editing there.
  - Edit the file on DECtape and then use PIP or FLX to transfer the file to disk.
- To use a device other than SY:, mount it with the 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 or the DCL DELETE command.
- EDI accepts variable-length input lines up to 132(10) 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.

2.6 SAMPLE EDITING OPERATIONS

Sample editing operations are included in this section to illustrate how the various EDI commands can be used:

- A file is edited using a few basic EDI commands.
- 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 manner shown in the second example.
- An immediate macro command is defined and executed in a single step.
- A file containing errors is edited using the macro commands.

2.6.1 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 be /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]
*
;
*
; 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]
*P *
;
; COMMAND FILE TO BUILD
; PRINT SYMBIONT
; FOR RSX-11M MAPPED SYSTEM
;
;
```

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 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 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.

7 - The line pointer is moved down two lines by means of a 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 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 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 by means of a TOF command. A PRINT * command is issued to print the complete file with all corrections

```
[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                                  An EXIT command is issued to close the file and
[EXIT]                               terminate the editing session.
```

2.6.2 SAVE and UNSAVE Sample

```
*LI                                  The file to be used in this example is
THIS IS LINE 1 PAGE 1                printed using the LIST command.
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                                   The line pointer is returned to the top.
*SA 5 SAV1.DAT                       A SAVE command is used to save the
                                     five lines in a separate file.

*T                                   The line pointer is returned to the top.
*SA 5 SAV2.DAT                       A second SAVE command is used to generate
*CL                                  a second saved file. The primary input file is closed.
EDI>SAV1.DAT                         The first save file is opened and a
[PAGE    1]                          LIST command is used to display the file.
*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*]
```

```
*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
```

A PASTE command is used to change PAGE 1 to PAGE 2 in all lines.

```
*CL
EDI>SAVE2.DAT
[PAGE     1]
```

The first save file is closed.
The second save file is opened.

```
*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 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
```

A PASTE command is used to change PAGE 1 to PAGE 3 in all lines.

```
*CL
EDI>START.DAT
[PAGE     1]
```

The second save file is closed.
The original input file is opened again.

```
*BO
THIS IS LINE 5 PAGE 1
```

The last line in the file is located.

```
*UNS SAV1.DAT
*UNS SAV2.DAT
*T
```

Two UNSAVE commands are used to append the two save files to the original input file.

```
*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]
```

A LIST command is used to display the contents of the combined file.

2.6.3 Use of MACRO IMMEDIATE Command

A LIST command is issued to print the file used in this example.

```
*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*]
```

The immediate macro is defined and executed to find the first four lines that 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.

```
*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
*
```

2.6.4 Use of Macro Commands

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).

```
*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*]
```

The three macro definitions that will correct the errors are typed.

```
*MACRO 1 C/NN/MM/
*MACRO 2 SC/TT/RR/
*MACRO 3 PA/YHE/THE/
```

Macro 3 is used to change all YHE strings to 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 is used to locate a line with errors.

```
*NP2
MANY CONNON ETTORS SO
```

M1 is used to change NN to MM.

```
*M1
MANY COMMON ETTORS SO
```

M2 is used to change TT to RR.

```
*M2
```

```
MANY COMMON ERRORS SO
*NP2
THE MACRO CONNANDS CAN              NP2 is used to locate the next line in error.
*M1
THE MACRO COMMANDS CAN              M1 is used to change NN to MM.

*M2                                 M2 is used to locate the next TT string
                                    and change it to RR.
WITH AN ERROR; AND THEN, THE
*                                   A RETURN is used here to locate the next line
MACRO EXECUTE CONNAND               in error.
*M1
MACRO EXECUTE COMMAND               M1 is used to change NN to MM.
*M2
IS ISSUED TO CORRECT THE            M2 is used to locate the next TT string and
*M2                                 change it to RR.

ERROR                               M2 is used to locate the last error in the
*T                                  file and correct it.
*LI                                 After all lines have been corrected, the
THIS LITTLE FILE HAS                file is printed using the LIST command.
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*]
```

2.7 EDI ERROR MESSAGES

The four classes of EDI error messages are:

- Command level information messages
- File access error messages
- Error messages requiring EDI to restart
- Fatal error messages

The following sections describe all the messages that can be displayed in each class. If the recovery procedure is not evident, a suggested user action is given.

2.7.1 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 are 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 by using 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 by using the SIZE command procedure that follows:

1. Type KILL.
2. When EDI prompts for a new file specification, enter a nonexistent file name. EDI creates a new file and enters input mode.
3. Use the RETURN key 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 with 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 pressing RETURN and 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 (for example, a READ command in line-by-line mode).

[ILL NUM]

Explanation: Either you have supplied a nonnumeric 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 (for example, 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 the PIP /RE switch or the DCL RENAME command 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 contains a percent (%) character and thus demands a numeric argument.

User Action: Reenter the command line, specifying the appropriate numeric argument.

[MCALL FILE DOES NOT EXIST]

Explanation: You have issued an MCALL command to retrieve 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 or the appropriate DCL commands to copy or rename the MCALL file into the current directory.

[NO INPUT FILE OPEN]

Explanation: You have issued a PAGE, READ, or RENEW command 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: You have issued a CHANGE command with a string to be changed that is not in the current line.

[OVERLAYING PREVIOUSLY DEFINED MACRO]

Explanation: You have issued a MACRO command that redefines 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 provide 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: You have issued a CLOSE or KILL command 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: You entered a command 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(10) characters.

User Action: Reenter the command line to ensure that the line is valid.

[*BOB*]

Explanation: You have reached the beginning-of-buffer. 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: You have reached the end-of-buffer. 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: You have reached the end-of-file 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.

2.7.2 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: EDI did not accept the file name. The most common error is a file name containing embedded blanks.

User Action: Make sure that the file name is correct, then reenter it.

EDI -- DEVICE NOT IN SYSTEM

Explanation: You have given a FILE, OPENS, SAVE, or UNSAVE command and specified 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 and specified 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 (SPR).

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

2.7.3 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 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 pressing 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 the 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 9.

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 reads 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 9.

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 (for example, the task was aborted). Thus, the file attributes were not written and the file may contain inconsistent data.

User Action: Exit from EDI by pressing RETURN or CTRL/Z. Use the PIP /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 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(10) 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 (for example, the task was aborted). Thus, the file attributes were not written and the file may contain inconsistent data.

User Action: Use the PIP /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 (SPR).

EDI -- DUPLICATE ENTRY IN DIRECTORY

EDI -- END OF FILE

EDI -- ILLEGAL RECORD ACCESS BITS - FILE NOT USABLE

EDI -- ILLEGAL RECORD NUMBER - FILE NOT USABLE

2.7.4 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.

You 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 the PIP /RE switch or the DCL RENAME command 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 to. 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, or use the DCL PURGE and DELETE commands.

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 9.

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 your 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, or use the DCL PURGE or DELETE commands.

NOTE

The following error messages signify hardware problems. If possible, remove all important files from the volume, then contact your local DIGITAL Field Service representative.

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. If such a failure occurs, contact your local DIGITAL Field Service representative.

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
```

CHAPTER 3

PERIPHERAL INTERCHANGE PROGRAM (PIP)

The Peripheral Interchange Program (PIP) is a 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:

- Copying files from one device to another
- Deleting files
- Renaming files
- Listing file directories
- Setting the default device and UIC for PIP operations
- Unlocking files
- Spooling 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.

3.1 PIP COMMAND LINE

You request PIP functions by entering PIP command lines through the initiating terminal or by means of an indirect command file. The maximum nesting level for indirect command files is four. (Using indirect command files is described in Chapter 1.) The format of PIP command lines differs for each function. Therefore, the command line formats are described in separate sections.

3.1.1 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.

In the following example, T1.MAC;5 sets the defaults for the subsequent file specifications in the command line. Then, T2 is specified and overrides T1 as the default filename; however, .MAC remains the default file type. Finally, .TSK is specified, which overrides .MAC as the default, while T2 remains the default file name.

Note, in this example, that the version number does not default.

```
     PIP>T1.MAC;5,T2,.TSK/BR

     T1.MAC;5
     T2.MAC;1
     T2.TSK;3
```

Table 3-1 summarizes the rules PIP uses to set defaults.

Table 3-1
PIP Default File Specifications

Element	Default Value
dev:	For the first file specification, the unit on which the user's system disk is mounted (SY0:) or the default that you specify with the /DF switch (see Section 3.2.2.6). 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 User Identification Code or UIC (that is, the UIC under which you log on), the UIC you specify with the SET command, or the default you specify with the /DF switch (see Section 3.2.2.6). For subsequent file specifications, either you explicitly specify a new User File Directory or PIP assumes the UFD from the previous specification. Only the asterisk specification is valid as a wildcard (see Section 3.1.3).
filename	No default for the first file specification. For subsequent file specifications, the last file name that you explicitly specified. Asterisk and/or percent sign specifications are valid as wildcards (see Section 3.1.3).
.filetype	No default for the first file specification. For subsequent file specifications, the last file type that you explicitly specified. Asterisk and/or percent sign specifications are valid as wildcards (see Section 3.1.3).
;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 exist in the output directory. An exception is the PIP file delete function, which requires that a version number be specified. An explicit version number is defined to be of the form ;n where n is greater than 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, just the asterisk (wildcard) may be specified, as described in Section 3.1.3.

3.1.2 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 switch or subswitch can have arguments that are separated from the switch or subswitch name by a colon (:).

To allow several commands to be performed consecutively, more than one command can be specified in a line. To separate each command, the ampersand character (&) is used.

Most of the PIP switches operate on lists of file specifications. The exceptions are /DD, /DF, /ID, and /TD, which are used by themselves.

Table 3-2 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 in Section 3.2.2.

Table 3-2
PIP Switches and Subswitches

Switch	Subswitch	Function
/AP		Appends file(s) to the end of an existing file.
	/FO	Specifies the file owner for a file.
/BS:n[.]		Defines the blocksize for magnetic tape.
/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.
/DD		Restricts file searches to files created during a specified period of time.
/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.
/EX		Excludes one file specification's worth of files during file searches.

(continued on next page)

Table 3-2 (Cont.)
PIP Switches and Subswitches

Switch	Subswitch	Function
/FI:filenum:seqnum		Accesses a file by its file identification number (file-ID).
/FR		Displays the amount of available space on the specified volume, the largest contiguous free space on that volume, and the number of available file headers.
/ID		Identifies the version of PIP being used.
/LI		Lists directory files.
	/BR	Lists a directory file in brief format (an alternate mode for the /LI switch).
	/FU[:n[.]]	lists a directory file in full format (an alternate mode for the /LI switch).
	/TB	Lists the total number of blocks used for a directory, along with the total number blocks allocated and the number of files in that directory (an alternate mode for the /LI switch).
/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.
/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.

(continued on next page)

Table 3-2 (Cont.)
PIP Switches and Subswitches

Switch	Subswitch	Function
/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		Specifies that records may span disk block boundaries when copied from magnetic tape.
/SD		Selectively deletes files by prompting for your response before deleting.
/SP[:n[.]]		Spools files 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.
/TD		Restricts file searches to files created on the current day.
/TR		Truncates files 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 owner for a file.

Switches and subswitches are described in the following sections.

3.1.2.1 **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 device name, the UFD, the file name, file type, or version of the file on which it is to operate.

You may specify a switch once for a list of file specifications. For example:

```
filespec1,filespec2,filespec3/DE
```

The /DE switch applies to all of the file specifications. PIP deletes every specified file from its UFD.

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.

3.1.2.2 Subswitches

You can apply subswitches to one or more file specifications, depending on the placement of the subswitch. Subswitches can appear in either the output file specification or the input file specification.

If you place the subswitch in the output file specification, the subswitch applies to the entire list of input file specifications. For example, the Contiguous Output switch (/CO) is applied to both TEST.TSK and SAMP.DAT. (The /CO switch is used with the Copy function. See Section 3.2.1.)

```
PIP>/CO=TEST.TSK;1,SAMP.DAT;1
```

PIP copies TEST.TSK;1 and SAMP.DAT;1 such that the copies, TEST.TSK;2 and SAMP.DAT;2, are contiguous.

If you place the subswitch in the input file specification, it usually applies only to the file specification that immediately precedes it. In the following example, the New Version subswitch (/NV) is applied to the file ASDG.MAC. (The /NV subswitch is being used with the Rename switch, /RE.)

```
PIP>*.SMP=PRT2.QRT,ASDG.MAC/NV,KG.MAC/RE
```

PIP renames the files PRT2.QRT and KG.MAC, 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 (assuming a version of ASDG.SMP already exists).

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.CMD/GR:R/WO,FILE2.MAC/GR:RW
```

This command is equivalent to:

```
PIP>FILE1.CMD/GR:R/WO,FILE2.MAC/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 3.2.2.16.)

3.1.3 Wildcards

PIP allows you to specify wildcards in file specifications. The wildcard characters are the asterisk (*) and the percent sign (%) characters. You can use both wildcards in place of explicit specifications for file names, and types and just the asterisk wildcard in place of file directories and version numbers.

The asterisk can denote zero or more characters in the field you specify it in, while the percent sign character can denote exactly one character in the fields. (Correct syntax must be followed, however. See Section 3.1.3.2.)

Wildcards are restricted in some cases. The following sections describe and give examples of wildcards in input and output file specifications.

3.1.3.1 Wildcards in Output File Specifications

- Wildcards in the output file specifications are restricted. For the following PIP functions, the output file specification cannot have any wildcards:

- Concatenating files to a specified file
- Appending files to an existing file
- Updating (rewriting) an existing file
- Listing a directory

If you use wildcards in the output file specification for any of these functions, the meaning of the command line would be ambiguous. For example:

```
     PIP>LIST.*=[200,200]/LI
```

You have incompletely specified the output file specification. PIP returns an error message.

When you make copies of several files, the output specification must be *.*;* or defaulted from the input file specification(s).

For the Rename (/RE) and Enter (/EN) switches, the output specification may have wildcards (asterisk only) 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.

NOTE

The percent sign (%) cannot be used in output file specifications.

3.1.3.2 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.
- *.D*;* means all versions of all files with file types beginning with D.
- TEST.*;* means all versions of all types of files named TEST.
- T*.*;* means all versions of all types of files with names beginning with T.
- TEST.DAT;* means all versions of file TEST.DAT.
- TEST.D%T;* means all versions of files named TEST with three-character file types beginning with D and ending with T.
- T%N.*;* means all versions of all file types of all three-character file names beginning with T and ending with N.
- *.* means the most recent version of all files.
- *.DAT means the most recent version of all files of file type .DAT.
- *%.DAT means the most recent version of all files that have at least one character in their names and have the file type of .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 (1 to 377 octal).
- [n1,*] means all member numbers under group n1.
- [*,n2] means all group numbers for member n2.

NOTE

The percent sign (%) character cannot be used in the UFD.

3.2 PIP COMMAND FUNCTIONS

PIP copies Files-11 files and performs file control functions. The copying function and file control functions are described in the following sections.

3.2.1 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 defaulted or *.*;*, the input file name, file type, and version are used for the output file (see /NV and /SU subswitches). If you explicitly specify 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 not specified, 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 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

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 3-3 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:, naming it SAMP.DAT.

2.
```
PIP>DK1:[*,*]=DK0:[11,*]
```

   Copies all files from all members in group number 11 of DK0: to DK1:. The files are copied to the same UFD on DK1: that they were in on DK0:. Note that the user must have write access to all group number 11 UFDs on DK1:.

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 that 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. Note the result would be the same if the /ME switch was not specified.

5.
```
PIP>DK1:=DB2:TESTPROG.MAC,.OBJ
```

   Copies the latest versions of TESTPROG.MAC and TESTPROG.OBJ from DB2: to DK1:, using the current UFD for both DB2: 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 (SY:) to DT0:, using the current UFD.

8.
```
PIP>DP0:[200,10]=DT0:*.*
```

   Copies the latest versions of all files from DT0: in the current UFD to DP0:[200,10]. Note that the user must have write access to [200,10].

Table 3-3
PIP Copy Command and Merge Subswitches

Subswitch	Description
/BL:n[.]	Blocks Allocated -- This subswitch specifies the number of 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 can 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 copying from tape. (PIP allocates file space before the copy operation is executed. The length of magnetic tape input files is on the trailing label for the file.)
/-CO	Noncontiguous Output -- This subswitch specifies that the output file does not have to be contiguous. If you do not specify the /BL:n subswitch, the /CO subswitch, or the /-CO subswitch, PIP defaults to the size and attributes of the input file.

/FO	Set File Ownership -- This subswitch specifies that the owner of the output file will be the same as the output UFD. If you do not specify /FO, the 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],
	
	 `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,
	
	 `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.

(continued on next page)

Table 3-3 (Cont.)
PIP Copy Command and Merge Subswitches

Subswitch	Description
/FO (Cont.)	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 may already exist in a UFD. The existing file is deleted and a new one is created with the data from the input file(s). If the file does not already exist, it is created. The output file's name, type, and version number remain the same, but its file identification number (file-ID) may be 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 3-1 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.)

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 sub-switch must be explicit or default; no wild cards allowed.

ZK-174-81

Figure 3-1 Results of Copy Command With and Without /NV Specified

3.2.2 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 several PIP switches in a command line with no file specifications (that is, they may be entered by themselves). These switches include /DD, /DF, /ID, and /TD.

You can specify one or more commands in a line. When using multiple commands, the ampersand character (&) separates each command. For example:

```
PIP>TEST.DAT;2=SAMP.DAT;2/SU,SAMP.DAT;1&TEST.DAT;*/FU&SAMP.DAT/PU/LD
```

1. PIP merges SAMP.DAT;2 and SAMP.DAT;1 into TEST.DAT;2. The /SU subswitch causes TEST.DAT's file name, file type, and version number to remain the same, but TEST.DAT;2's contents are replaced with the input file's contents.

2. The /FU switch causes PIP to do a full format listing of all versions of the file TEST.DAT.

3. The /PU switch causes PIP to purge SAMP.DAT and the /LD subswitch causes PIP to list the deleted files.

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.

3.2.2.1 **/AP -- Append Switch** - The Append switch (/AP) opens an existing file and appends the input file(s) to the end of it. Specify the /AP 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(s) 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 not specified, 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 3.2.1 for examples of using /FO.

Example

```
PIP>DK1:FILE1.DAT;1=FILE2.DAT;1,FILE3.DAT;1,FILE4.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 that if the output file is contiguous before the appending, it may not be contiguous afterwards.

3.2.2.2 **/BS:n -- Block Size Switch** - The Block Size switch (/BS:n) 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. The default block size is 512(10) bytes per block. Specify the /BS switch using the following format:

```
outfile/BS:n=infile
```

outfile

The output file specification.

/BS:n[.]

The Block Size switch, where n is an octal or decimal number specifying the number of bytes in a block.

infile

The input file specification.

The /BS switch specifies the block size of the output file. If the block size specified is smaller than the actual block size, an I/O error occurs.

Example

```
PIP>MT:BA.DOC/BS:2048.=AMBER.DOC
```

This command increases the block size of the output file, BA.DOC, to 2048(10) bytes per block.

3.2.2.3 **/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 explicit or implicit Merge switch (/ME) and/or with an output magnetic tape device. Specify the /CD switch in the following format:

```
     outfile/CD=infile1[,infile2...,infilen]

            or

     outfile=infile1/CD[,infile2...,infilen]
```

outfile

The output file specification.

infile

The input file specification.

/CD

The Creation Date switch.

Example

```
     PIP>/LI

     DIRECTORY DB1:[200,200]
     21-NOV-80 14:02

     FILE.DAT;7          12.      6-OCT-80 16:13


     PIP>TEST.DAT/CD=FILE.DAT/LI


     DIRECTORY DB1:[200,200]
     21-NOV-80 14:05

     FILE.DAT;7          12.      6-OCT-80 16:13
     TEST.DAT;1          12.      6-OCT-80 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.

3.2.2.4 **/DD -- Default Date Switch** - The Default Date switch (/DD) restricts file searches to files created during a specific period of time. Specify the /DD switch in the following format:

```
     /DD:startdate:enddate
```

/DD

The Default Date switch

startdate

The beginning date of the specified time period in the form dd-mm-yy. May be unlimited by using the wildcard character (*).

enddate

The ending date of the specified time period in the form dd-mm-yy. May be unlimited by using the wildcard character (*).

Specifying the wildcard for both startdate and enddate negates the /DD switch:

```
/DD:*:*
```

The date restrictions for the file searches are now disabled.

Examples

1. `PIP>/DD:01-JUN-80:01-JUL-80/LI`

   Lists all files created from 1 June 1980 through 1 July 1980.

2. `PIP>/DD:*:1-JUN-80/LI`

   Lists all files created on or before 1 June 1980.

3. `PIP>/DD:1-JUN-80:*/LI`

   Lists all files created on or after 1 June 1980.

3.2.2.5 **/DE -- Delete Switch** - The Delete switch (/DE) deletes files from a UFD. Optionally, you can specify that the deleted files be listed on your terminal. Specify the /DE switch 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 an unspecified file name, file type, and version does not default to *.*;*.

The input file specification can take all the usual forms, including wildcards (even in [ufd]). The only special requirement is that the version field must always be specified.

Examples

1. `PIP>TEST.DAT;5/DE`

   Deletes version 5 of file TEST.DAT from the current default directory on the default device.

2. `PIP>TEST.DAT;1,;2/DE`

   Deletes versions 1 and 2 of file TEST.DAT from 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 both file types.

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.

3.2.2.6 **/DF -- Default Switch** - The Default switch (/DF) changes the default device and/or UFD for the current PIP task.

The usual default device of PIP is the user's system device (SY0:).

The usual default UFD is the UIC under which PIP is currently running. The /DF switch alters only the default UFD. It does not affect the UIC under which PIP is running, nor does it circumvent file protection.

Specify the /DF switch in one of the following formats:

```
dev:[ufd]/DF

     or

dev:/DF

     or

[ufd]/DF

     or

/DF
```

dev:

If specified, the new default device to be applied to subsequent PIP command lines.

[ufd]

If specified, the new default UFD to be applied to subsequent PIP command lines.

/DF

The Default switch.

The /DF switch specified with no arguments returns the default device to the user's system device (SY0:) and the default UFD to the UIC from which PIP was invoked.

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. `PIP>/DF`

   Returns the user's default device to SY0: and the default UFD to the UIC from which PIP was invoked.

3.2.2.7 **/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. A file name, file type, 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/EN[/NV]
```

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 returns an error message that requests that the line be reentered.

The default input file specification is *.*;*.

/EN

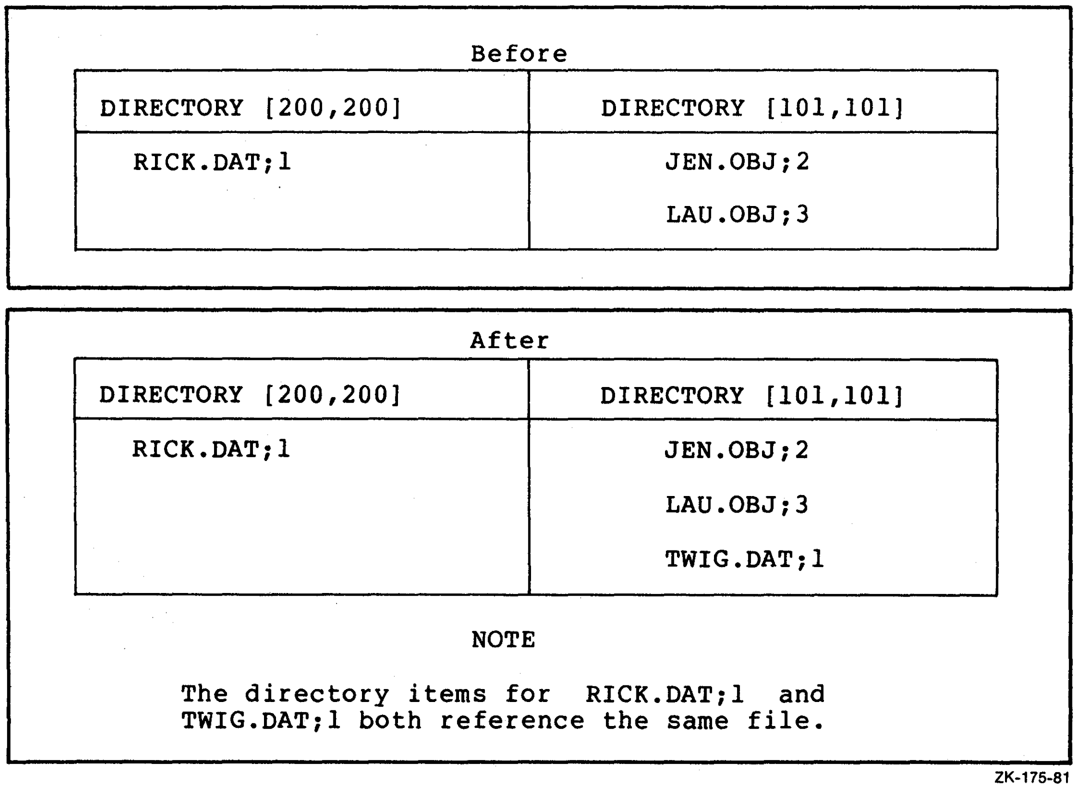
The Enter switch.

/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. (Specifying the /NV subswitch is not necessary when both the input and output files are under the same file directory.)

Example (see Figure 3-2)

```
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

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 3-2 Sample Directories Before and After Execution

3.2.2.8 **/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 classified group or world to the file owner's UIC, you need read access to read the attribute and write access to change it.

Specify the /EOF switch in the following format:

```
infile1/EOF[:block:byte][,...infilen/EOF[:block:byte]]
```

infile

The input file specification.

The file specification must be issued because an unspecified file name, file type, and version do not default to *.*;*.

block

The block number where the EOF pointer is to be placed. Usually, the EOF pointer cannot be placed beyond the highest number of blocks allocated to the file. However, if all the bytes of the allocated blocks are used, the EOF pointer can be placed in the first byte of the next block (/EOF: blocks allocated plus one:0). The block number can be octal or decimal.

byte

The byte location of EOF is the first unused byte of the specified 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 past 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 returns an error message.

Note that the /EOF switch 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:23:0
```

where the file AA.TMP has 22 blocks allocated.

3.2.2.9 **/EX -- File Exclusion Switch** - The File Exclusion switch (/EX) excludes one file specification's worth of files during file searches. You can exclude any field in the file specification. The fields can have in them characters and/or wildcards. Specify the /EX switch in the following format:

```
filespec/EX
```

filespec

The file specification. The file name and/or the file type and/or the version number can be a wildcard, but not all three fields. Also, you cannot specify devices or UFDs.

/EX

The File Exclusion switch.

Specifying the /EX switch by itself negates it.

```
PIP>/EX
```

Example

```
PIP>*.CMD;*/EX/LI

DIRECTORY DB1:[301,7]
8-JUL-81 14:50
*.CMD;*EXCLUDED

EXECM.MAC;23        45.          23-FEB-80 14:23
RUN.TSK;46          5.           29-OCT-80 11:59
FRANK.OBJ;16        33.          02-MAY-80 13:58
DEBBIE.COR;2        5.0.         14-JAN-81 12:01
```

Excludes all files of the type .CMD from the search done in *.*;*/LI.

3.2.2.10 **/FI -- File Identification Switch** - The File Identification switch (/FI) allows you to access an existing file by its file identification number (File-ID). Specify the /FI switch in the following format:

```
outfile=/FI:filenum:seqnum
```

/FI

The File Identification switch.

filenum

The file number.

seqnum

The sequence number of the file.

The file identification number (file-ID) 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). The /FU switch displays the file identification and sequence numbers and other information describing the file.

Examples

1. You can use the /FI switch to create a directory entry for a file.

```
PIP>XYZ.TSK=/FI:301:27/EN
```

2. You can copy a file using the /FI: switch.

```
PIP>A.B=/FI:301:27
```

3. To list entries in the directory file whose file identification is 1275,47, use the /FI switch in the format:

```
PIP>/FI:1275:47/LI


DIRECTORY DR2: FILE ID 001275,000047,0
8-AUG-81 15:58

MCR.CMD              1.          29-NOV-80 13:22
DCL.CMD              1.          29-NOV-80 13:24
```

3.2.2.11 **/FR -- Free Switch** - The Free switch (/FR) displays the amount of available space on a specified volume, the largest contiguous space on that volume, the number of available file headers, and the number of file headers used. Specify the /FR switch in the following format:

```
[dev:]/FR
```

If you do not specify dev:, PIP defaults to SY0:.

The format of the information from the /FR switch is shown below.

```
dev: HAS xxxx. BLOCKS FREE, yyyy. BLOCKS USED OUT OF zzzz.
LARGEST CONTIGUOUS SPACE = nnnn. BLOCKS
aaaa. FILE HEADERS ARE FREE, bbbb. HEADERS USED OUT OF cccc.
```

Usually, the number of free file headers corresponds to the number of files that can be created. However, fragmented files and files that are too large for one file header must be allocated more than one file header.

The number of file headers will not exceed the number of files that can be created.

Example

```
PIP>db7:/FR

DB7: HAS 10662. BLOCKS FREE, 330008. BLOCKS USED OUT OF 340670.
LARGEST CONTIGUOUS SPACE = 4189. BLOCKS
9025. FILE HEADERS ARE FREE, 11931. HEADERS USED OUT OF 20956.
```

3.2.2.12 **/ID -- Identify Switch** - The Identify switch (/ID) identifies the version of PIP being used. Specify the /ID switch in the following format:

```
/ID
```

When you specify this switch, the version number is listed on the input terminal as follows:

```
PIP VERSION Mvvee (ANSI)
```

vv

The version number.

ee

The edit number.

(ANSI)

If PIP is linked to an ANSI FCS, this field will appear. Otherwise, it is blank.

NOTE

For RSX-11M operating systems, if you want ANSI magnetic tape support, PIP must be built with ANSLIB instead of SYSLIB.

Example

```
PIP>/ID

PIP -- PIP VERSION M1340 (ANSI)
```

3.2.2.13 **/LI -- List Switch** - The List switch (/LI) lists one or more files contained in a UFD, along with their status information. Three alternate mode subswitches (/BR, /FU, and /TB) allow you a choice of directory listing formats. Table 3-4 describes these switches. Specify the /LI switch in the following format:

```
[listfile=]infile1[,infile2...,infilen]/LI[/subswitch]
```

listfile

The file specification to be listed 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.

/subswitch

The alternate mode subswitch of the List switch (described in Table 3-4).

Table 3-4
List Subswitches

Subswitch	Description
/BR	Specifies the brief form of directory listing. This switch lists only the file name, file type, and version.
/FU[:n[.]]	Specifies the full directory format. Because the /FU format uses 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 classified 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 3.2.2.16.) If specified, n is the number of characters per line. If not specified, the number defaults to the buffer size of the output device. The /FU 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)

(continued on next page)

Table 3-4 (Cont.)
List Subswitches

Subswitch	Description
/FU[:n[.]] (Cont.)	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] 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 used.

/TB	This switch only outputs 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 3-3 contains sample directory listings in the various formats.

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
```

ZK-176-81

Figure 3-3 Directory Listing Examples

Examples

1. `PIP>/LI`

   Lists the directory of the current default device and directory. (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 that only RSX-11M-PLUS has transparent spooling (see Examples, Section 3.2.1).

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.

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 that only RSX-11M-PLUS has transparent spooling (see Examples, Section 3.2.1).

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 that only RSX-11M-PLUS has transparent spooling (see Examples, Section 3.2.1).

3.2.2.14 **/ME -- Merge Switch** - The Merge switch (/ME) creates a single file from two or more existing files. The /ME switch is used in copying Files-11 files and is described in Section 3.2.1.

3.2.2.15 **/NM -- No Message Switch** - The No Message switch (/NM) suppresses the PIP error messages, NO SUCH FILES(S) and FILE NOT LOCKED, 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, /PU, or /UN switches and any of their respective subswitches.

/NM

The No Message switch.

The /NM switch applies not only to the file specification preceding it, but also to all file specifications to the right of it.

Example

```
PIP>*.MAC;*/NM,TEST.DAT;1,FILES.OBJ;*/DE
```

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.

3.2.2.16 **/PR -- Protect Switch** - The Protect switch (/PR) allows you 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.

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 the /PR subswitches (/SY, /OW, /GR, /WO) or octal representation (/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
```

File specification must be issued because an unspecified file name, file type, and version do not default to *.*;*.

/PR

The Protect switch.

/SY,/OW,/GR, and /WO

The subswitches that specify protection level for a file. These subswitches specify which protection level is to be altered (others are left intact). The values that 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, in the format:

```
/PR:n
```

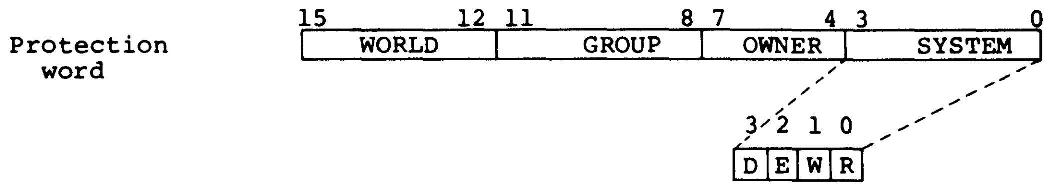
The variable n is the octal representation of the protection to be assigned to the file. This octal number is taken as the new protection word. (See the RSX-11M or RSX-11M-PLUS Mini-Reference Manual for the list of octal codes.) The format of the protection word is shown in Figure 3-4.

/FO

The Set File Ownership subswitch, which allows you 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 that are owned by another UIC, the command

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

causes all files to be owned by [200,200] without changing their protection.



(bit set means NO access permitted.)

Example

TEST.DAT;5/PR:3

(bits 0 & 1 set)
deny write and read access to the system
for file TEST.DAT;5.

ZK-177-81

Figure 3-4 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.

3.2.2.17 **/PU -- Purge Switch** - The Purge switch (/PU) deletes a specified range of obsolete versions of a file. Optionally, you can specify that the names of deleted files be listed on your terminal.

Specify the /PU switch in the following format:

```
infile1[,infile2...,infilen]/PU[:n][/LD]
```

infile

The file specification for the file(s) to be deleted. The file specification takes the form:

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

Note that a version number is not needed. If specified, it is ignored.

/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 3-5). 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.

/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 deleted files.

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.

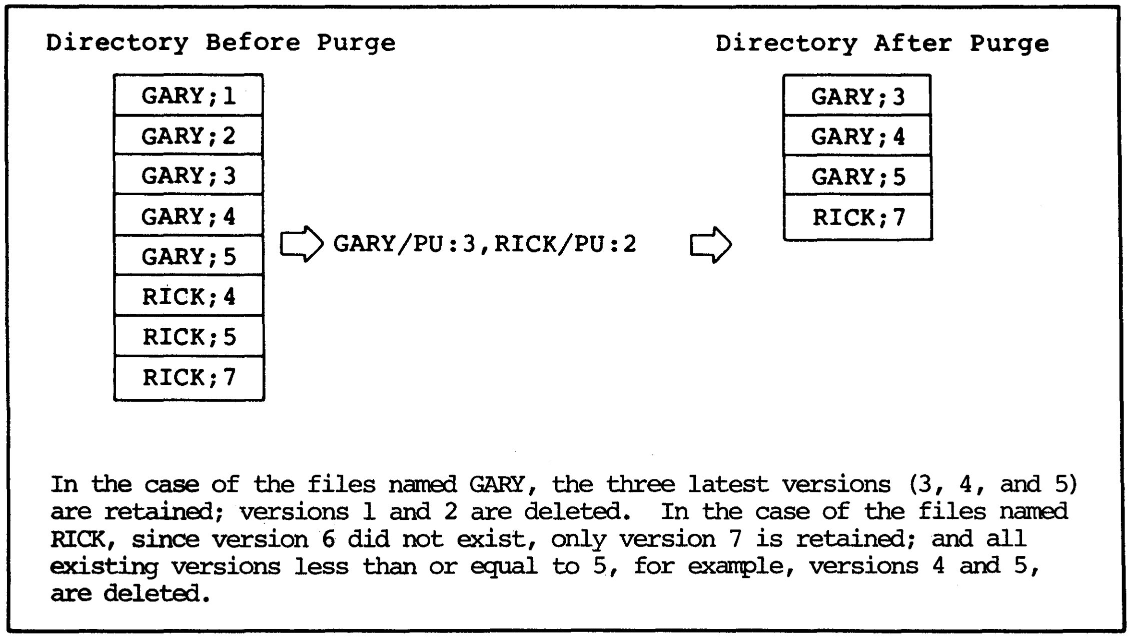

Directory Before Purge
GARY;1
GARY;2
GARY;3
GARY;4
GARY;5
RICK;4
RICK;5
RICK;7
GARY/PU:3,RICK/PU:2
Directory After Purge
GARY;3
GARY;4
GARY;5
RICK;7
In the case of the files named GARY, the three latest versions (3, 4, and 5) are retained; versions 1 and 2 are deleted. In the case of the files named RICK, since version 6 did not exist, only version 7 is retained; and all existing versions less than or equal to 5, for example, versions 4 and 5, are deleted.
ZK-178-81

Figure 3-5 Use of the Purge Switch

3.2.2.18 **/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 (see Figure 3-6). Note that you cannot rename a file that is copied from one device to another device. 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. The output file specification takes the following form:

```
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. The input file specification takes the following form:

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

An unspecified file name, file type, and version defaults to *.*;*.

The /RE switch 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 returns an error message 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. (Specifying /NV is not necessary when both the input and output files are under the same directory file.)

Examples

1. `PIP>TESTFILE.DAT;1=TEST.DAT;5/RE`

   Renames TEST.DAT;5 to TESTFILE.DAT;1.

2. `PIP>BACKUP.*;*=TEST.*;*/RE`

   Renames all versions of all files with file names TEST to BACKUP, preserving the file type and version of each file.

3. `PIP>*.*;1=*.*;*/RE`

   Renames all copies of all files to version 1.

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 3-6 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
```

ZK-179-81

Figure 3-6 Results of Rename Switch With and Without /NV Specified

3.2.2.19 **/RM -- Remove Switch** - The Remove switch (/RM) removes an entry from a UFD, but does not delete the file associated with that entry. The Remove switch 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 9). Specify the /RM switch in the following 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:.

3.2.2.20 **/RW -- Rewind Switch** - The Rewind switch (/RW) directs PIP to rewind magnetic tape. (The /RW switch cannot be used for DECtapes.) You can apply this switch to both input and output specifications. When you specify the /RW switch with the output specification, PIP begins writing the file at the beginning of the tape. You can use this technique to erase a tape before writing files on it. Specify the /RW 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 the /RW switch to the input specification, it rewinds the tape before searching for 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.

Example

```
PIP>MT:/RW=[200,200]
```

Starts the beginning of the tape and outputs all files in the directory [200,200].

```
PIP>AMBER.DOC=MT:AB.DOC/RW
```

Rewinds the tape, then searches the tape for the file AB.DOC and outputs the file renaming it AMBER.DOC.

3.2.2.21 **/SB -- Span Blocks Switch** - The Span Blocks switch (/SB) allows you to control whether records copied from magnetic tape to disk 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:

```
outdsk:outfile/SB=inmag:infile
```

outfile

The disk output file.

infile

The magnetic tape input file.

/SB

The Span Blocks switch.

Example

```
PIP>DK1:FILES.DAT/-SB=MM0:FILES.DAT
```

Copies FILES.DAT records to the disk from magnetic tape. Records on the disk will not cross block boundaries.

3.2.2.22 **/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 for deletion. The response choices are carriage return (<RET>) or control-Z (^Z), or Y, N, G, or Q, each followed by either a carriage return (<RET>) or control-Z (^Z). Table 3-5 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 3-5
Response Choices for the Selective Delete Switch

Letter	Terminator	Operation
Y	RET	Delete this file and continue.
Y	CTRL/Z	Delete this file and exit from PIP.
[N]	RET	Save this file and continue.
[N]	CTRL/Z	Save this file and exit from PIP.
Q	RET	Save this file and return to command mode.
Q	CTRL/Z	Save this file and exit from PIP.
G	RET	Delete this and all remaining candidates, list deleted files, and return to PIP command mode.
G	CTRL/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 RET
DELETE FILE DB1:[200,200]MYFILE.DAT;2 [Y/N/G/Q]?  G RET

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 then PIP prompts for the next command.

2.
```
PIP>TEST.*;*/SD

DELETE FILE DB1:[200,200]TEST.DAT;1 [Y/N/G/Q]?  N RET
DELETE FILE DB1:[200,200]TEST.TXT;3 [Y/N/G/Q]?  Q CTRL/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 and then exits from PIP.

3.2.2.23 **/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. (For more information on the Queue Manager and the Print Spooler, see the RSX-11M/11M-PLUS Batch and Queue Operations Manual.) Specify the /SP 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 *.*;*.

If the file is specified by its file identification number (file-ID), it will be printed. File identification numbers are discussed in Section 3.2.2.10.

/SP

The Spool switch.

n

The number of copies you want spooled. (If a deleting spooler was specified during system generation, 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.

3.2.2.24 **/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.

3.2.2.25 **/TD -- Today Default Switch** - The Today Default switch (/TD) restricts file searches to files created on the current day. Specify the /TD switch in the following format:

```
/TD
```

/TD

The Today Default switch

Note that specifying the wildcard for both startdate and enddate of the /DD switch (/DD:*:*) also negates /the TD switch (/DD, see Section 3.2.2.4).

Examples

```
PIP>/TD/LI

DIRECTORY DB2:[301,357]
20-JAN-81 11:02
DAY OF 20-JAN-81

TEST.DAT;1          1.          20-JAN-81 10:40
FILES.TXT;1         1.          20-JAN-81 10:41
INFO.DAT;1          1.          20-JAN-81 10:50

TOTAL OF 3./15. BLOCKS IN 3. FILES
```

3.2.2.26 **/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 an unspecified file name, file type, and version do not default to *.*;*.

/TR

The Truncate switch.

Example

```
PIP>*.MAC/LI

DIRECTORY DR2:[301,7]
2-AUG-81 15:32

A.MAC;1     3.      20-SEP-80 14:02
B.MAC;1     2.      20-SEP-80 15:38
C.MAC;2     5.      28-SEP-80 09:54

TOTAL OF 10./15. BLOCKS IN 3. FILES
```

```
PIP>*.MAC/TR
PIP>*.MAC/LI

DIRECTORY DR2:[301,7]
2-AUG-81 15:33

A.MAC;1     3.      20-SEP-80 14:02
B.MAC;1     2.      20-SEP-80 15:38
C.MAC;2     5.      28-SEP-80 09:54

TOTAL OF 10./10. BLOCKS IN 3. FILES
```

FCS allocates a certain number of blocks to a file. The file may or may not use the number of blocks allocated to it. The /TR switch moves the EOF pointer to the end of the file and frees the unused blocks for use by other files.

3.2.2.27 **/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 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 create the UFD(s) from which you are obtaining the files being transferred. Specify the /UF switch in the following format:

```
outfile/UF[/FO]=infile1...,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. The /FO subswitch is described in Section 3.2.1.

Example

```
PIP>DK6:[*,*]/UF/FO=SY:[104,20]*.MAC,*.OBJ
```

To use the /UF switch, 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 the /UF switch. If the volume to which you are writing files is your 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 the /UF switch 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:.

3.2.2.28 **/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 that contain the end-of-file information

After you have used the /UN 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 the file RICK1.OBJ;3 in directory [100,100] on device DK1:.

3.2.2.29 **/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 3.2.1) /UP does not delete the existing file before rewriting the data. Therefore, 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 the /SU switch, 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 for 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. The input file specification(s) take the form:

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

An unspecified 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 the /FO switch, 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 3.2.1 for examples for 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.

3.3 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 a message file on the system device. If PIP cannot access the message file, 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 3-6.

-mm

One of the standard system, file primitive, or FCS 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, then reenter the command line. Also, use the BRU or DSC utilities to compress the files on your disk. BRU is described in Chapter 7 and DSC is described in Chapter 8.

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, then reenter the command line. Also, use the BRU or DSC utilities to compress the files on your disk. BRU is described in Chapter 7 and DSC is described in Chapter 8.

PIP -- BAD USE OF WILD CARDS/CHARACTERS IN DESTINATION FILE NAME

Explanation: A wildcard/character was specified for an output file name when use of a wildcard/character was explicitly disallowed.

User Action: Reenter the command line with the output file explicitly specified.

PIP -- CANNOT EXCLUDE *.*;*

Explanation: The /EX switch does not accept all wildcards as the input file specification.

User Action: Determine the files to be excluded and reenter the command line.

PIP -- CANNOT FIND DIRECTORY FILE

Explanation: The specified UFD does not exist on the 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 line was(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. If the failure is on the output file, the output 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 9.

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 drive had not been allocated, the device was not mounted, or another user had mounted the device.

User Action: Allocate the drive and/or mount the device, then 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 device 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 off-line or nonresident.

User Action: Ensure that the device is on-line and reenter the command line.

PIP -- FAILED TO ATTACH TERMINAL

Explanation: PIP could not attach a terminal, probably because of a privilege violation.

User Action: Determine the cause of the failure and correct it. 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 was(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 (SPR).

PIP -- FAILED TO OPEN INDEX FILE

Explanation: PIP was unable to read the index file, probably because of a privilege violation.

User Action: Retry the operation by running PIP under a system UIC, or have the system manager change the protection on the index file.

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 the operation 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 9.

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 (using /RE) into another directory, and failed to replace the original directory entry.

User Action: Run the lost check of the File Structure Verification Utility (VFY) to recover the file name. VFY is described in Chapter 9.

PIP -- FAILED TO SPOOL FILE FOR PRINTING

Explanation: The Queue Manager is not installed.

User Action: Install the Queue Manager 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 9.

PIP -- FAILED TO WRITE ATTRIBUTES

Explanation: The volume you specified 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 9.

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 file name. VFY is described in Chapter 9.

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 reenter the command line. If the error persists, submit a Software Performance Report (SPR).

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: The command was not recognized by PIP.

User Action: Reenter the command line with the PIP command correctly specified.

PIP -- ILLEGAL EOF VALUE

Explanation: You specified an illegal block and/or byte value in the command line.

User Action: Reenter the command line with the correct values.

PIP -- ILLEGAL RESPONSE - TRY AGAIN

Explanation: Self-explanatory.

User Action: Check which response you want and enter it when PIP prompts you.

PIP -- ILLEGAL SWITCH

Explanation: The 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 OR LATEST VERSION

Explanation: The use of either a wildcard version number or a latest 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. With a Merge, Update, or Supersede command, the attributes of the output file will be those of the first input file. With 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 on-line
- The device is not mounted
- The hardware has failed
- The volume is full (output only)
- The input file is corrupted

Note that these are the most common conditions. Conditions other than those listed may have caused the message.

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 with 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 was(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.

Note that these are the most common conditions. Conditions other than those listed may have caused the message.

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 or use /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.

3.4 PIP ERROR CODES

Table 3-6 identifies the error codes PIP issues when it does not have access to the message file. The descriptions and suggested user actions for these error codes are identical to those described in Section 3.3.

Table 3-6
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/CHARACTERS 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 OR LATEST 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	FAILED TO ATTACH TERMINAL
30	ILLEGAL RESPONSE - TRY AGAIN
31	CANNOT EXCLUDE *.*;*
32	CANNOT FIND DIRECTORY FILE
33	FAILED TO ATTACH OUTPUT DEVICE
34	FAILED TO GET TIME PARAMETERS
35	NOT A DIRECTORY DEVICE
36	FAILED TO WRITE ATTRIBUTES
37	FAILED TO READ ATTRIBUTES
38	FILE NOT LOCKED
39	FAILED TO ENTER NEW FILE NAME
40	FAILED TO RESTORE ORIGINAL DIRECTORY ENTRY - FILE IS LOST
41	CANNOT RENAME FROM ONE DEVICE TO ANOTHER
42	FAILED TO SPOOL FILE FOR PRINTING
43	CANNOT SPOOL BY FILE ID (RSX-11D only)
44	FAILED TO OPEN STORAGE BITMAP FILE
45	FAILED TO OPEN INDEX FILE
46	FAILED TO FIND FILE(S)
47	CANNOT FIND FILE(S)
48	NO SUCH FILE(S)
49	FAILED TO REMOVE DIRECTORY ENTRY
50	DIRECTORY WRITE PROTECTED
51	NOT ENOUGH BUFFER SPACE AVAILABLE
52	FAILED TO TRUNCATE FILE
53	CANNOT TRUNCATE THIS FILETYPE
54	ILLEGAL EOF VALUE

CHAPTER 4

FILE TRANSFER PROGRAM (FLX)

The File Transfer Utility Program (FLX) allows you to use foreign volumes (not in Files-11 format) in DIGITAL's DOS-11 or RT-11 format. FLX converts the format of a file to the format of the volume the file is being transferred to.

FLX can be used to initialize and list directories of cassettes and RT-11 or DOS-11 file-structured volumes. FLX can also be used to delete files from RT-11 or DOS-11 formatted 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	**Device Abbreviation**
PC11 paper tape punch	PP
PC11 or PR11 paper tape reader	PR
RK05 cartridge disk	DK
TU78 magnetic tape	MF
TE16, TU16, TU45, or TU77 magnetic tape	MM
TS04 magnetic tape	MS
TU10, or TS03 magnetic tape	MT
TU56 DECtape	DT
TU60 tape cassette	CT

Valid RT-11 devices are:

Device	Device Abbreviation
RL01/RL02 cartridge	DL
RK05 cartridge disk	DK
RK06 or RK07 cartridge disk	DM
RX01 floppy disk	DX
RX02 floppy disk	DY
TU56 DECtape	DT
TU58 DECtape II data cartridge	DD

FLX supports all Files-11 devices, including RSX-format cassettes. They are volumes that you have initialized using the MCR INITVOL or DCL INITIALIZE command. DOS-11 and RT-11 volumes are initialized using FLX. On RSX-11M-PLUS, DOS-11 and RT-11 volumes must be mounted with foreign characteristics before you can use FLX. On RSX-11M, such volumes must be unmounted.

You can use FLX interactively or by means of an indirect commmand 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 line. If you only specify FLX, the utility responds with the prompt:

```
FLX>
```

FLX can also access an indirect command file in the following manner:

```
>FLX @FOO.CMD

     or

FLX>@FOO.CMD
```

4.1 FLX COMMAND FORMAT

Although formats for specifying FLX functions vary, the general format for entering FLX command lines is:

```
devicespec/sw=infile/sw,...,infilen/sw
```

devicespec

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. Do not specify a UFD if the output device is in RT-11 format.

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 take the form:

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

The UFD is not specified for RT-11 volumes.

/sw

One of three types of FLX switches described below in Section 4.2.

FLX supports 9-character file names for DOS-11-format magnetic tapes. When you transfer the file back to Files-11 format, (which uses a 12-character filename) FLX will recover the last three characters.

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).

4.2 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.

4.2.1 Volume Format Switches

FLX has three volume format switches that define the format of the specified volumes.

/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 line.

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 line, 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.

4.2.2 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 4-1 describes the transfer mode switches.

Table 4-1
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 a carriage return/line feed (RET-LF), form feed (FF), or vertical tab (VT). In transfers from DOS-11 or RT-11 files to Files-11 files, RET-LF pairs are removed from the end of records. In transfers from Files-11 files to DOS-11 or RT-11 files, RET-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. 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 that are 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 (10) 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 (10) 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 described in Section 4.6).

Mode	**Switch**	**File Type**
Image	/IM:n	.TSK, .OLB, .MLB, .SYS, .SML, .ULB, .EXE
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.

The RSX-11M/M-PLUS MCR Operations Manual defines the above file types.

4.2.3 Control Switches

FLX provides a number of control switches to control file processing. Table 4-2 describes these switches.

Table 4-2
FLX Control Switches

Switch	Description
/BL:n[.]	Indicates the number of contiguous blocks (n) in octal or decimal to be allocated to the output file. This switch is normally used with the /CO switch (described later in the table). Because all RT-11 files are contiguous, the /CO switch is not required with the /BL:n switch for RT-11 output. 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.
/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 on paper tape, cassette, or DOS-11 magnetic tape, /BL is also required. FLX transfers the

(continued on next page)

Table 4-2 (Cont.)
FLX Control Switches

Switch	Description
/CO (Cont.)	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 line has no output specification.
/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, a wildcard is assumed. See Section 4.3 for information on DOS-11-volume directory manipulation. See Section 4.4 for information on RT-11-volume directory manipulation.
/DNS:n	Specifies the density of the magnetic tape; where n is 800, 1600 or 6250 bpi. If n is any other value or not specified, FLX prints an error message. If you do not specify /DNS:n, the magnetic tape density defaults to 6250 bpi for the TU78, 1600 bpi for the TS04, and 800 bpi for all other Magtape devices. If you specify /DNS with a nonmagnetic device, FLX ignores the switch.
/FC	When using FORTRAN files, indicates that FORTRAN carriage control conventions are to be used. 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) in octal or decimal 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 that can be allocated is 37(8) 31 (10).

(continued on next page)

Table 4-2 (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 /RW switch with a non-magnetic-tape device, or with /LI, /DI, or /ZE, FLX ignores /RW.
/SP	Indicates that the converted file is to be spooled by the print spooler task or the queue management system. The /SP switch applies only to Files-11 output files.
/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 a cassette to be read and verified. The /VE switch is only valid with a CT output file specification.
/ZE[:n.]	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. The /ZE switch does not require a file specification. For DOS-11 DECtape, /ZE creates an entry for the current UIC.

4.3 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, DOS-11 volumes must be mounted with foreign characteristics before you can use FLX. On RSX-11M, the volumes must be unmounted.

4.3.1 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 line lists on your terminal the directory of all .MAC files under UFD [100,100] on the DOS-11 DECtape on DT0:.

Figure 4-1 shows sample directory listings for a DOS-11 DECtape and a TU60 cassette.

```
DECtape Directory Listing

  ❶ DIRECTORY    ❷ DT:[200,200] ❸
  ❹ 19-SEP-78

  ❺ FLX.TSK         ❻104.  ❼19-SEP-78  <233>❿
    UFD.TSK           8.     19-SEP-78  <233>
    TKN.TSK           6.     19-SEP-78  <233>
    MOU.TSK           14.    19-SEP-78  <233>

   ❾ TOTAL OF 132. BLOCKS IN 4. FILES

Cassette Directory Listing

  ❶ DIRECTORY    ❷ CT:[200,200] ❸
  ❹ 19-SEP-78

  ❺ UFD.TSK-0                ❻ 28.  ❼19-SEP-78 128.❽
    TKN.TSK-0                  20.    19-SEP-78 128.
    MOU.TSK-0                  52.    19-SEP-78 128.

   ❾ TOTAL OF 100. BLOCKS IN 3. FILES
```

ZK-180-81

Figure 4-1 DOS-11 Directory Listings

Notes on Figure 4-1:

❶ This line identifies the listing as a directory listing.

❷ The device name and unit number.

❸ The User File Directory.

❹ The date the directory was listed.

❺ The file name, file type, version number, and sequence number (cassettes only).

❻ The file size in decimal blocks.

❼ The file creation date.

❽ The record size in decimal bytes for the file (cassettes only.)

❾ A total of the actual file sizes and the total number of files in the directory.

❿ The default protection code provided by the system.

4.3.2 Deleting DOS-11 Files

You can delete files from DOS-11 disks or DECtapes using the Delete switch (/DE). The /DE switch requires only the file specification for the file you are deleting. For example:

```
     FLX>DK1:[100,100]SYS1.MAC/DE
```

This command line deletes SYS1.MAC under UFD [100,100] from the DOS-11 disk on DK1:.

4.3.3 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 line initializes the DECtape on DT1: in DOS-11 format.

4.4 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, RT-11 volumes must be mounted with foreign characteristics before you can use FLX. On RSX-11M, the volumes must be unmounted.

4.4.1 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 specification 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 4-2 shows a sample directory listing for an RT-11 disk.

```
❶DIRECTORY ❷DK:
❸ 4-JUN-78

 SIPBOO.MAC❺ 49. ❻4-JUN-78
❹< UNUSED >    6.
 SIP   .MAC   10.   4-JUN-78
 SIPCD .MAC    7.   4-JUN-78
 < UNUSED >   21.
 SIPQIO.MAC    7.   4-JUN-78
 < UNUSED > 4686.

❼ 4713. FREE BLOCKS

❽ TOTAL OF 73. BLOCKS IN 4. FILES
```

ZK-181-81

Figure 4-2 RT-11 RK05 Cartridge Disk Directory Listing

Notes on Figure 4-2:

❶ This line identifies the listing as a directory listing.

❷ The device name and unit number.

❸ The date the directory was listed.

❹ The file name and file type; <UNUSED> indicates free space.

❺ The number of blocks in the file or free space.

❻ The file creation date, or blank for free space.

❼ The total number of free blocks on the volume.

❽ The total number of blocks allocated to files on the volume.

4.4.2 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 line on which you specify /DE/RT requires only the file specification for the file you are deleting. For example:

```
FLX>DK1:SYS/.MAC/DE/RT
```

This command line deletes SYS/.MAC from the RT-11 volume on DK1:.

4.4.3 Initializing RT-11 Volumes

You can initialize RT-11 volumes using the /ZE switch with the /RT switch. The /ZE switch requires only the device specification for the volume you are initializing. For example:

```
FLX>DT1:/ZE/RT
```

This command line initializes the DECtape on DT1: in RT-11 format. When you initialize RT-11 volumes, the /ZE switch takes an optional argument in the form:

```
/ZE:n
```

The value n specifies the number of extra words per directory 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 uses seven words; 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
```

Some RT-11 applications require extra words in the directory entries. When you specify extra words for directory entries (/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 the /NU switch with the /ZE and /RT switches 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 line 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 uses 9 words.

4.5 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 shown in Table 4-3.

Table 4-3
Differences Between Files-11 and DOS-11 Cassette Files Format

Files-11 Format	DOS-11 Format
Standard level 2	Standard level 0
12-character file name (9-character file name and 3-character file type)	9-character file name (6-character file name and 3-character file type)
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 block sizes can also share a given cassette. FLX uses the block size contained in the file label data when reading a file.

4.5.1 Multivolume Cassette Support

FLX supports multivolume cassette files in both Files-11 and DOS-11 formats. No special switches are required to notify FLX that a multivolume file is being accessed.

4.5.2 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]
   ```

   The variables 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 output 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.

   b. Mount the next output cassette volume and type Z, followed by a carriage return. The new output cassette is rewound, and FLX continues by transferring data onto it. Thus, the tape is effectively zeroed (initialized) before data is transferred to it.

   c. Enter a carriage return to terminate the transfer.

      If you select this alternative, FLX assumes that end-of-file (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.

4.5.3 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
   ```

   The variables 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. If you select this alternative, 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.

   b. Type a carriage return to terminate the transfer. If you select this alternative, FLX assumes that end-of-file (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.

4.6 FLX PAPER TAPE SUPPORT

FLX supports the DIGITAL standard paper tape devices, such as the PC-11 Paper Tape Reader/Punch and the PR-11 Paper Tape Reader, as DOS-11 devices.

FLX lets you 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, .EXE, 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 by default, with fixed-length, 512(10)-byte records.

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 also use PIP to create a contiguous file from the file that is read back from tape (see Chapter 3).

The use of explicit transfer mode switches to transfer .TSK, .OLB, .MLB, .SYS, .SML, .EXE, and .ULB files between paper tape and Files-11 volumes can cause files read back 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 line writes an output 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. Therefore, you must specify the /DO switch with paper tape file specifications. The following examples 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 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
```

4.7 FORTRAN DIRECT ACCESS FILES

FORTRAN direct access files must be transferred in Image mode. For example:

```
FLX>DK0:/DO/IM=FOO.TJP/RS
```

To recover the file, you must specify the record length in bytes (not to exceed 512(10) bytes). For example:

```
FLX>/RS/IM:n=DK0:FOO.TJP/DO
```

The variable n specifies the record length in bytes.

4.8 FLX ERROR MESSAGES

Errors encountered by FLX during processing are reported on the initiating terminal. The FLX error messages, their explanations, and suggested user actions are described in this section.

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 specification 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 question 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, the disk is not spinning).
- The volume is not mounted or is allocated to another user.
- The volume is not on-line.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

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, magnetic tape) was entered with the /CO switch.

User Action: Reenter the command line without specifying the /CO switch.

FLX -- CASSETTE ERROR I/O TERMINATED

Explanation: A 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 qualifiers were entered. For example:

```
SY0:=DT0: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 the /DO switch has an incorrect DOS-11 file structure, or the device specified with the /RT switch has an incorrect RT-11 file structure.

User Action: Correctly identify the file structure on each volume, and then reenter the command line.

FLX -- DT: UFD FULL

Explanation: The DECtape directory is full.

User Action: Delete all unnecessary files, and reenter the command line.

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 4.5.2 if an output transfer is being performed or Section 4.5.3 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 in the proper format.
4. A hardware error occurred during a directory I/O operation (for example, 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. Reenter the command line.

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 multivolume cassette 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: Reenter the command line with only one level of indirect command file specified.

FLX -- @ FILE SYNTAX ERROR

Explanation: A syntax error occurred in the indirect command file.

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 is not of the specified type.

User Action: If the file is corrupted, no recovery is possible. If the file type is incorrect, reenter the command line 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 for the purpose specified. For example, a line printer was specified as an input device.

User Action: Reenter the command line with a valid 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 command line with the proper syntax.

FLX -- INVALID SWITCH

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 off-line.
- A hardware error occurred (for example, bad tape).

User Action: Ensure that the device is on-line. Reenter the command line. If a hardware error recurs, 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 device is not on-line.
- The specified volume is not mounted.
- A hardware error occurred (for example, bad tape).

User Action: Ensure that the device is on-line 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 from the initiating terminal; 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 SY0: for operations involving DOS-11 CT, DT, or MT volumes. This error occurs when one of the following conditions exists:

- SY0: is not on-line and mounted.
- SY0: 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 operation. There is a hardware problem with the output device (for example, a 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 an operation that does not allow one.

User Action: Reenter the command line 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: A carriage return 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. Therefore, 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 n specified with the /FA, /FB, or /IM switch 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 directory, or the specified volume may not be a valid Files-11 volume.

User Action: Reenter the command line specifying the correct file name and directory.

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 volumes, the /CO switch can only be specified when the /BL switch is also specified.

User Action: Reenter the command line, specifying the /BL switch.

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.

FLX -- ILLEGAL DENSITY VALUE

Explanation: Either the specified density value is not supported by the target tape drive or some value other than 800, 1600 or 6250 was input.

User Action: Reenter the command with the proper density value.

CHAPTER 5

DISK VOLUME FORMATTER (FMT)

The Disk Volume Formatter (FMT) utility formats and verifies disk cartridge, disk pack, fixed media disk, and flexible disk volumes under any RSX-11M or RSX-11M-PLUS operating system that includes on line formatting support in the Executive. (Check with your system manager to determine whether your system includes this feature.)

In general, FMT performs the following functions:

- Writes a complete header for each sector of the volume it is formatting.
- Verifies the address contents of each sector header.
- Sets the density for RX02 floppy disks (DY).
- Lets you specify an error limit for the volume being formatted. FMT terminates processing when the error limit is reached.
- Lets the Bad Block Locator task (BAD) run (spawn) if your system permits spawned tasks.

5.1 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]
```

The variable dd is the abbreviation for the volume you are formatting and n is the unit number of the volume. The possible switches are:

/BAD

Runs the Bad Block Locator task (BAD) if it is installed on the system.

Note that the /BAD switch can only be used with operating systems that allow spawning of tasks. RSX-11M provides spawned tasks as a system generation option. RSX-11M-PLUS always includes support for spawned tasks.

/DENSITY or /DENS

Selects high (double) or low (single) density for RX02 floppy disks.

/ERL

Determines the number of errors FMT will allow on the volume.

/MANUAL or /MAN

Enters manual operating mode and formats the sector or track you specify.

/NOVERIFY or /NOVE or /-VERIFY or /-VE

Inhibits the default verification of a successful FMT operation.

/OVR

Overrides or ignores the Manufacturer's Defined Bad Sector File (MDBSF).

/WLT

Rewrites the MDBSF (on the last track of the device) to add bad sectors found during FMT operation.

/VERIFY or /VE

Verifies that an FMT operation was successfully completed. This switch is the default.

/@Y

Informs FMT that it is receiving input from an indirect command file that you have created. A FMT command in this form does not allow operator intervention in the process.

These switches are described in detail in Section 5.4.

To terminate FMT, type CTRL/Z (^Z) following the FMT prompt, as explained in Chapter 1 of this manual.

5.2 MODES OF FMT OPERATION

FMT lets you format volumes in two operating modes: normal and manual. In normal operating mode, FMT formats the entire volume. In manual operating mode, FMT permits you to format individual sectors (or tracks for DM:-type disks) that you specify in response to FMT prompts. FMT uses normal operating mode unless you specify manual mode with the /MAN switch in the command line.

FMT normally retries an operation 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 FMT encounters an error during the verification operation, it prints one of the following messages on your terminal:

```
     Error reading header

             or

     Header compare error
```

FMT then continues the verification operation.

5.2.1 Normal Operating Mode

When you invoke FMT in normal operating mode (without the /MANUAL switch), FMT prints the following message:

```
     ** WARNING - Data will be lost on ddn:  **
     Continue?  [Y or N]
```

CAUTION

If you answer yes, FMT erases all data previously stored on the disk.

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, FMT returns control to the system monitor.

Normal FMT operation varies slightly according to the volume you are formatting (see Section 5.3).

5.2.2 Manual Operating Mode

If you specify manual operating mode (/MAN), FMT prints:

```
     ** WARNING - Data will be lost on ddn:  **
     Continue?  [Y or N]
```

CAUTION

If you answer yes, FMT erases all data previously stored on the disk.

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 pound sign (#) to indicate an explicit octal response. If you enter a parameter that is out of the range for the volume, FMT returns an error message and exits. Table 5-1 lists the valid ranges for FMT manual mode operations.

Note that FMT manual operating mode cannot be used with RX02 floppy disks.

FMT manual operating mode works the same on all disk volumes, with one exception: On DM:-type volumes (RK06 and RK07), FMT formats a specific track.

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= 237
    Track   = 1
```

FMT then formats the entire track you specified.

Table 5-1
Ranges for Manual FMT Operations

Devicel	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
RM05	0-31	0-18	0-822
RM80	0-31	0-13	0-558

5.3 FMT SUPPORTED DISK VOLUMES

The following sections describe using normal FMT operating mode with the different types of FMT supported devices. Table 5-2 lists the disk volumes that allow formatting and their device mnemonics.

Table 5-2
FMT-Supported Disk Volumes

Disk Volumes	Device Mnemonic
RP04 disk pack	DB:
RP05 disk pack	DB:
RP06 disk pack	DB:
RK05 disk cartridge	DK:
RK05F fixed media disk	DK:
RL01 disk cartridge	DL:
RL02 disk cartridge	DL:
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:
RM05 disk pack	DR:
RM80 fixed media disk	DR:
RX02 floppy disks	DY:

The status FMT requires for the disk volumes varies with the operating system. On an RSX-11M operating system, the disk volume must be unmounted.

On an RSX-11M-PLUS operating system, the disk volume must be mounted with foreign characteristics (MOUNT/FOREIGN).

5.3.1 DB:-type Devices (RP04/RP05/RP06 Disk Packs)

When FMT formats a DB:-type volume, it tries to write 22 sector 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.

Unless you specify the /-VE switch, FMT verifies 11 headers at a time until it has verified the 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.

5.3.2 DK:-type Devices (RK05 Disk Cartridge or RK05F Fixed Media Disks)

When FMT formats a DK:-type volume, it tries to write each sector header individually until it has formatted the entire volume. If FMT encounters an error, it retries each header before reporting the header as bad.

Unless you specify the /-VE switch, FMT verifies 12 headers at a time until it has verified the 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.

5.3.3 DL:-type Devices (RL01/RL02 Disk Cartridges)

FMT does not format a DL:-type volume, but reads each block, one track at a time, and determines which blocks were marked as bad by the manufacturer's formatting process. If the MDBSF is corrupt or has been written over, FMT then rewrites the Manufacturer Detected Bad Sector File (MDBSF) on the last track with this information. When using FMT on a DL:-type device, the Write Last Track switch (/WLT) must be specified.

5.3.4 DM:-type Devices (RK06/RK07 Disk Cartridges)

FMT writes DM:-type 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 before it designates which headers are bad.

Unless you specify the /-VE switch, 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 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:-type devices, it sets bad sector flags in the headers already marked as bad in the MDBSF. Unless you specify the /-VE switch, FMT indicates whether the bad sector was flagged in the MDBSF.

5.3.5 DP:-type Devices (RPR02/RP02/RP03 Disk Packs)

When FMT formats a DP:-type volume, it tries to write 10 headers at a time until it has formatted the volume. If FMT encounters an error, it attempts to write each header individually and designates which headers are bad.

Unless you specify the /-VE switch, FMT verifies 10 headers at a time until it has verified the 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.

5.3.6 DR:-type Devices (RM02/RM03/RM05/RM80 Disk Packs)

When FMT formats a DR:-type volume, it tries to write 32 headers at a time until it has formatted the volume. If FMT encounters an error, it attempts to write each header individually and designates which headers are bad.

Unless you specify the /-VE switch, FMT verifies 16 headers at a time until it has verified the 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 verification operation.

When FMT writes headers on DR:-type volumes, it sets bad sector flags in headers already designated as bad by the MDBSF. Unless you specify the /-VE switch, FMT indicates whether the sector was marked bad in the MDBSF.

On the RM80 disk, in addition to performing the functions indicated for other DR: devices, FMT reads a Skipped Sector File to identify the sectors the manufacturer's formatter designated as "skipped sectors," and then sets the skipped sector flag bit in the appropriate sector headers. All data originally intended for a sector designated as a skipped sector is moved to the following sector (for more information, see the _RSX-11M/M-PLUS I/O Drivers Reference Manual_).

5.3.7 DY:-type Devices (RX02 Floppy Disks)

You can use FMT to set an RX02 floppy disk to either high (double) or low (single) density by using the /DENS switch. Unless you specify the /-VE switch, FMT writes and reads block 0 and the last block on the disk to determine that the density is consistent.

Note that manual operating mode cannot be used with DY:-type devices.

5.4 FMT SWITCH DESCRIPTIONS

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.

/BAD

The Bad switch spawns the Bad Block Locator task (BAD) after FMT completes its processing. The BAD operation tests for the number and location of any unusable blocks. BAD records this bad-block information which is used by the initializing function. If BAD is not installed on the system, FMT prints a warning message on your terminal and exits.

Note that the /BAD 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.

The format for an FMT command using the /BAD switch is:

```
FMT>ddn:/BAD
```

/DENSITY

The Density switch sets DY:-type floppy disks to either high (double) or low (single) density. The default is low density. (This switch can also use SINGLE and DOUBLE as options.)

The formats for an FMT command using the /DENS switch are:

```
FMT>DYn:/DENS=HIGH (or DOUBLE)
FMT>DYn:/DENS=LOW (or SINGLE)
```

/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 the operation. The default error limit is 256(10) errors. Any value greater than 0 or less than or equal to 256(10) is valid.

The format for an FMT command using the /ERL switch is:

```
FMT>ddn:/ERL=n.
```

/MANUAL

The Manual switch puts FMT in manual operating mode and permits you to format an individual sector (or track for DM:-type disk cartridges) of a device. FMT assumes cylinder, track, and sector numbers are decimal values unless they are preceded by a pound sign (#), which indicates octal values.

Note that manual operating mode cannot be used with DY:-type devices.

In manual operating mode, FMT displays the following prompts:

```
** WARNING - Data will be lost on ddn:  **

Continue [Y OR N]?
Entering manual mode
Cylinder=
Track   =
Sector  =

Operation complete
```

The format for an FMT command using the /MAN switch is:

```
FMT>ddn:/MAN
```

/NOVERIFY

The Noverify switch inhibits the operation performed by the default /VERIFY switch.

The format for an FMT command using the /NOVERIFY switch is:

```
FMT> ddn:/NOVERIFY
```

/OVR

The Override switch causes FMT to ignore the Manufacturer's Detected Bad Sector File (MDBSF) on DM:- and DR:-type 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 the reporting operation.

The format for an FMT command using the /OVR switch is:

```
FMT>ddn:/OVR
```

/VERIFY

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. This switch is the default.

The format for an FMT command using the /VE switch is:

```
FMT>ddn:/VE
```

/WLT

The Write Last Track switch, when used with the Verify switch on DM:- and DR:-type volumes, rewrites the MDBSF 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 must be specified when using FMT on a DL:-type device.

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>ddn:/WLT=n
```

/@Y

If you specify the @Y switch, FMT receives input from an indirect command file that you have created. In this method of operation, FMT will not generate any operational messages or warnings to your terminal. No user intervention is possible until the FMT operation is complete.

The format for an FMT command using the /@Y switch in an indirect command file is:

```
FMT ddn:/@Y
```

Note that to run FMT from an indirect command file, FMT must be installed before hand. Otherwise, you will receive an error message and the FMT operation will discontinue.

Example

To format a DK:-type volume using an indirect command file, create the indirect command file FMTIND.CMD (you can specify any file name) which contains the following:

```
FMT DK3:/@Y
```

Then issue the following indirect command from the prompt:

```
>@FMTIND
```

FMT will start to format the disk, DK3: and the /@Y switch will inhibit any FMT message from printing on your terminal until after FMT has finished.

5.5 FMT MESSAGES

This section describes the messages FMT generates, and possible user responses.

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 hardware configuration.

User Action: Determine the correct command line with the correct device mnemonic 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, then reenter the FMT command.

Device offline

Explanation: The device is not in the system 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 exceeded either the number of errors specified with the /ERL switch or the default 256(10) 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 when it tried to read data from a disk.

User Action: None required. FMT retries the operation 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 and continues the verification.

Error setting diskette density

Explanation: FMT tried to format a RX02 floppy disk but 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 tried to write sector headers.

User Action: None required. FMT retries the operation 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.

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 tried to read the MDBSF on the last track of a device.

User Action: Reenter the command, including the Override switch (/OVR).

Fatal hardware error

Explanation: A fatal error occurred in the system hardware configuration.

User Action: Contact your DIGITAL Field Service representative.

Header compare error

Explanation: FMT found an error when it tried to compare headers with the expected value during a verification error.

User Action: None required. FMT retries the operation.

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 file (MDBSF) on the last track of the disk is in an unusable format.

User Action: Reenter the 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 your 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 mounted or 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 for the volume.

User Action: Check Table 5-1 (Section 5.2.2) for valid 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: An FMT command specified the Bad switch (/BAD), but BAD could not be spawned. Either the operating system does not spawn tasks or BAD is not installed.

User Action: Run the BAD utility separately (see Chapter 6 for more information).

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, or refer to the error codes in the IAS/RSX-11 I/O Operations Reference Manual.

CHAPTER 6

BAD BLOCK LOCATOR UTILITY (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 volume. Then you use the MCR Initialize Volume command (INI), which allocates the bad blocks to the file [0,0]BADBLK.SYS. The bad blocks are marked as in-use and therefore 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 (see Sections 6.4.1.1 and 6.4.1.2). You can use BAD in its task version, which runs at the same time as other tasks, or in its stand-alone version ([1,51]BADSYS.SYS), which runs by itself on the computer. The stand-alone version is required if you have a system with a single disk drive.

6.1 BAD COMMAND FORMAT

The command line for BAD is in the following format:

```
BAD>dev:[/sw]...
```

dev

Specifies a physical device. The specification consists of two alphanumeric characters followed by a 1- to 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.

6.2 BAD SWITCHES

Table 6-1 contains a reference list of BAD switches along with a brief description of each. For a detailed description of BAD switches see Section 6.6.

Table 6-1
BAD Switches

Switch	Function
For Task and Stand-Alone Versions	
/ALLOCATE:volume label or /ALO:volume label	Prompts you for blocks to be allocated to BADBLK.SYS and to be entered in the bad block descriptor file.
/LIST or /LI	Lists bad blocks as they are located.
/MANUAL or /MAN	Prompts you for additional bad blocks which are entered in the bad block descriptor file.
/OVERRIDE or /OVR	Creates the bad block descriptor file on a last-track device.
/PATTERN=m:n or /PAT=m:n	Specifies the double word data pattern used to locate bad blocks.
/RETRY	Recovers soft errors.
/UPDATE or /UPD	Reads the bad block descriptor file and prompts for input.
For Stand-Alone Version Only	
/CSR=nnnnnn	Specifies the CSR address of a device that is not in a standard location.
/VEC=nnn	Specifies the interrupt vector address of a device that is not in a standard location.
/WCHK	Causes a write check.
/NOWCHK	Negates /WCHK.

6.3 BAD AND INDIRECT COMMAND FILES

BAD can access an indirect command file that contains a series of BAD command lines 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 lines. BAD executes the comands and returns with the BAD prompt. BAD allows nested command files -- one command file can invoke another to a maximum depth of three.

BAD can also be invoked within an indirect command file. Such a command file can contain command lines for more than one utility and is accessed by entering only the file specification preceded by the at sign. For example:

```
>@INDIRECT.CMD
```

The default values for indirect command file specifications are:

dev	SY0:
ufd	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 command files, see the RSX-11M/M-PLUS MCR Operations Manual.

6.4 PROCESSING BAD BLOCK DATA

This section contains information on how BAD tests the reliability of disks and DECtapes and formats bad block descriptor entries and how the MCR INI command uses bad block information.

6.4.1 Verifying Devices

BAD verifies disks and DECtapes by writing a test pattern onto each of the blocks on the device, reading the pattern into a buffer in memory, and comparing the pattern written with the pattern read. 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 /PATTERN switch may be used to specify the double-word test pattern. Its default values are 165555(8) and 133333(8), which are replicated 128(10) times per block. If BAD finds no bad blocks during individual testing, the error logging subsystem may still log errors due to long data transfers.

6.4.1.1 **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. On non-last-track devices, BAD storage allows 126(10) entries of bad block addresses. If more than the maximum number of entries is recorded, BAD terminates with an error message. There must be at least one good block in the last 256(10) blocks of the volume for BAD to create this file, which is called the bad block descriptor file.

6.4.1.2 **BAD and Last-Track Devices** - BAD records bad block information differently on last-track devices than on non-last-track devices. Last-track devices include the RK06/07, RL01/02, RP07 and the RM02/03/05/80. 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 manufacturer 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. BAD storage allows 126(10) entries of bad block addresses. The information contained in the two last-track files is combined to form [0,0]BADBLK.SYS when you issue the MCR INI command.

6.4.2 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 recorded as a double-word in the SDBSF. The first word of the double word contains the address of the cylinder on which the bad block exists. The high-order and low-order bytes of the second word contain, respectively, the track and sector addresses of the bad block.

6.4.3 The INI Command and BAD

Use BAD followed by 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 [0,0] BADBLK.SYS file has allocated to it those blocks found to be bad, thus ensuring that the file system does not allocate a known bad block to a file.

For information on how to use the INI command, see the RSX-11M/M-PLUS MCR Operations Manual.

6.5 USING BAD

Before BAD can validate a device, that device must be formatted by the manufacturer or by FMT (see Chapter 5).

On an RSX-11M system, the volume must not be mounted. Issue the MCR Dismount command if the device contains a mounted volume.

On an RSX-11M-PLUS system, the volume must be mounted as foreign.

The following example illustrates a typical sequence of steps for introducing a disk (DK1:) to an RSX-11M or RSX-11M-PLUS system.

```
RSX-11M Commands                  RSX-11M-PLUS Commands

ALL DK1: RET                      ALL DK1: RET
FMT DK1:[/sw] RET                 MOU DK1:/FOR RET
BAD DK1:[/sw] RET                 FMT DK1:[/sw] RET
INI DK1:[label][/sw] RET          BAD DK1:[/sw] RET
MOU DK1:[label][/sw] RET          INI DK1:[label][/sw] RET
                                  DMOU DK1: RET
                                  MOU DK1:[label][/sw] RET
```

You may execute BAD while other RSX-11M/M-PLUS tasks are executing.

Note that if the /ALO switch (Section 6.6.1) is used with BAD, the volume must be mounted as a Files-11 device and the user must have a privileged account.

6.5.1 Programming Considerations

This section contains information you should know before you use BAD.

6.5.1.1 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.

6.5.1.2 Device Controller Errors

The error logging subsystem may log 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.

6.6 BAD SWITCH DESCRIPTIONS

The following sections describe the switches you can use with BAD commands. The command format is described in Section 6.1.

6.6.1 Switches for Both Task and Stand-Alone Versions of BAD

/ALLOCATE:volume label

Causes BAD to prompt you for additional bad blocks which are added to the bad block descriptor file and allocated to [0,0] BADBLK.SYS. The /ALLOCATE switch eliminates the need to reinitialize the disk after updating the bad block descriptor file. This switch does not cause BAD to write pattern checks.

NOTE

To use this switch, the volume must be mounted as a Files-11 device and the user must be privileged.

/LIST

Causes all bad blocks to be printed by number (in decimal) on your terminal. The bad blocks are listed as BAD performs a data pattern check on each block. BAD does not list manually entered blocks that are tested as reliable. This switch is valid for all devices.

/MANUAL

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 or the SDBSF.

/OVERRIDE

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 with the INI command 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.

/PATTERN=m:n

Causes BAD to locate bad blocks by means of a user-specified double word data pattern.

The variable m:n represents the two 16-bit octal numbers used as the double word data pattern. A decimal number may be specified by placing a period after the number.

/RETRY

Causes BAD to attempt a recovery of hardware errors by means of the device driver. This also means that soft errors, such as an ECC (Error Correction Code) correctable error, will be recovered and the block will be marked as good.

/UPDATE

Causes BAD to immediately read the bad block decriptor file and prompt you for additional bad blocks. 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.

Examples of the /MANUAL, /ALLOCATE and /UPDATE Switches, which require user input, are described in the following sections.

6.6.2 The /MANUAL, /ALLOCATE and /UPDATE Switches: Examples

If you enter bad blocks by using the /MANUAL, /ALLOCATE or /UPDATE switches, BAD will prompt you as follows:

```
     BAD>LBN(S)=
```

You may then enter bad blocks in the format:

```
     blocknum:number
```

The variable 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 format. Both blocknum and number default to decimal values unless preceded by a pound sign (#), which indicates an octal value. For example:

```
     BAD>LBN(S)=70:3(RET)
```

This command enters the block numbers 70, 71, and 72 in the bad block descriptor file. If you are using the /ALLOCATE switch, the blocks are also allocated to [0,0]BADBLK.SYS.

You can also specify a single bad block. For example:

```
     BAD>LBN(S)=3(RET)
```

This command enters block 3 in the bad block descriptor file. The /ALO switch also allocates block 3 to BADBLK.SYS.

You can use both of these forms on the same command line. For example:

```
     BAD>LBN(S)= 100:2,3,  200:100  45:1
```

This command enters blocks 100, 101, 3, 200 through 299, and 45 in the bad block descriptor file. The /ALO switch also allocates these blocks to [0,0] BADBLK.SYS. You can separate bad block sequences with a space, tab, or comma.

If you are using the Manual or Update switches, and 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)= (RET)
       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.

If you are using the /ALLOCATE switch and enter a carriage return in response to the prompt, BAD will list all the LBNs allocated to BADBLK.SYS before it will list the LBNs in the bad block descriptor file. For example:

```
BAD>LBN(s)= RET

LBNs allocated to BADBLK.SYS =

 004799:001
 000100:002
 000003:001
 000200:001

LBNs in BAD BLOCK File =

 000100:002
 000003:001
 000200:001
 000045:001

BAD> LBN(s) =
```

In this example, LBNs 100, 101, 003, and 200 are allocated to [0,0] BADBLK.SYS, as is LBN 4799 which is the LBN for the bad block descriptor file on this particular disk. Note that the LBN for the bad block descriptor file does not appear in the Bad Block file. However, this disk has one LBN (LBN 45) which is contained in the Bad Block file but is not yet allocated to [0,0] BADBLK.SYS. You can now allocate LBN 45 by entering it in response to the LBN(s)= prompt. (The "Duplicate block number" message will appear because the LBN already exists in the Bad Block file.)

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 for the /MANUAL, /ALLOCATE, or /UPDATE switch, enter ESCAPE, ALTMODE, or CTRL/Z in response to the prompt. The bad block descriptor file will then either be rewritten with the new bad block information (if you are using the /UPDATE or /ALLOCATE switch) or pattern checking will start (if you are using the /MANUAL switch). Blocks that you enter manually and that BAD decides are reliable are included in the bad block descriptor file.

6.6.3 Switches for Stand-Alone System Version Only

/CSR=nnnnnn

The variable nnnnnn is a new CSR address.

This switch allows you to specify the CSR address of the device so that it conforms to that of the device in the user's system. The /CSR switch remains in effect and need not be repeated if more command lines are issued.

/VEC=nnn

The variable nnn is a new interrupt vector address.

This switch allows you to specify the interrupt vector address so that it conforms to the vector address of the device in the user's system. The /VEC switch remains in effect if more command lines are issued.

/WCHK

This switch causes a write-check operation to occur after each write operation. The switch is not available for DT:-, DX:-, or DY:-type devices.

/NOWCHK

This switch negates the /WCHK switch.

BAD expects to see all switches on a single command line. For example:

```
BAD>DM3:/OVR/LI/VEC=300/CSR=174406
```

This command line locates all bad blocks on DM3:, ignores the last-track data, lists all bad blocks, specifies 300 as the interrupt vector, and specifies 174406 as the CSR address. All switches are validated for proper syntax before the actual bad block detection takes place.

6.7 DEVICES SUPPORTED BY BAD

The devices in Table 6-2 are supported by the stand-alone version of BAD. If you have a task version of BAD, the Executive will support any device suitable to your system's configuration.

Table 6-2
Devices Supported by Stand-alone BAD

Mnemonic	Type	CSR	Vector
DB	RP04/05/06	176700	254
DD	TU58 DECtape II	175600	300
DF	RF11 Fixed-Head Disk	177460	204
DK	RK03/05/05F Cartridge Disk	177404	220
DL	RL01/RL02 Cartridge Disk Pack	174400	160
DM	RK06/07 Cartridge Disk Pack	177440	210
DP	RPR02/RP02/03 Disk Pack	176714	320[1]

1. Nonstandard Vector Address

(continued on next page)

Table 6-2 (Cont.)
Devices Supported by Stand-alone BAD

Mnemonic	Type	CSR	Vector
DR	RM02, and RM03, RM05, RM80, RP07 Disk Pack	176700	340[1]
DS	RS03/04	172040	310[1]
DT	TU56 DECtape	177342	214
DU	RA80 Fixed Media Disk	177510	154
DX	RX01 Floppy Disk	177170	264
DY	RX02 Floppy Disk	177170	350[1]
EM	ML11 Electronic Memory	172000	

1. Nonstandard Vector Address

6.8 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 messages are arranged alphabetically beginning with the text following the device symbol (ddu:).

BAD --ddu: Allocation Failure

Explanation: BAD failed to allocate the block number sequence you entered. The I/O failed for a reason other than because the block number was already allocated to another file. This message applies to the /ALLOCATE switch only.

User Action: Either the volume is bad or the drive requires maintenance. Use another volume or contact your DIGITAL Field Service Representative to fix the drive.

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 126(10) entries of bad blocks. This message usually indicates a device unit failure.

User Action: Either the volume is bad or the drive requires maintenance. Use another volume or contact your DIGITAL Field Service Representative to fix the drive.

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 already allocated - LBN= numb

Explanation: The block number sequence you entered is already allocated to a file (the file may or may not be BADBLK.SYS). The value numb is the sequence you entered. The block sequence indicated by numb and list of block numbers which follow numb were neither allocated to [0,0]BADBLK.SYS nor entered into the bad block descriptor file. This message only applies to the /ALLOCATE switch.

User Action: Reenter the command line with another value.

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: BAD did not recognize the command line entered from the keyboard.

User Action: Reenter the command line.

BAD -- Command too long

Explanation: The command was longer than 80. characters.

User Action: Reenter the command line.

BAD -- ddu: CSR address not in system

Explanation: Self-explanatory. This message occurs only in the stand-alone system version of BAD.

User Action: Reenter the command line, specifying the proper value for the /CSR switch.

BAD -- ddu: Device offline

Explanation: In the stand-alone version of BAD, the specified device is not in the hardware configuration, or the /CSR switch is improperly set.

User Action: Reenter the command line, setting the CSR and Vector 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. BAD ignores any block number sequences you may have entered after the duplicate block numbers.

This message applies only to the /ALLOCATE, /MANUAL and /UPDATE switches. If this message appears when using the /ALLOCATE switch, it means that the block number which was allocated to [0,0]BADBLK.SYS already existed in the bad block descriptor file.

User Action: Reenter the command line with another value. This message applies to the /MANUAL, /ALLOCATE and /UPDATE switches only.

BAD -- ddu: Failed to attach

Explanation: BAD could not gain control of the device to be tested.

User Action: Determine if another task has attached the device. If so, wait until the task exits or abort the task to gain control of the device for BAD.

BAD -- ddu: Failed to read BADBLK.SYS header

Explanation: Self-explanatory. This message only applies to the /ALLOCATE switch.

User Action: The disk must be initialized using the MCR INI command.

BAD -- ddu: Failed to read Manufacturer's Bad Sector File

Explanation: A disk-read hardware error occurred while BAD was attempting to read the factory-written bad block data on the last-track device cartridge.

User Action: Reenter the command line with the /OVERRRIDE 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 line with the /OVERRIDE 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 line. 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 line.

BAD -- ddu: Home block not found

Explanation: BAD was unable to read the home block while attempting to validate the volume label. This message only applies to the /ALLOCATE switch.

User Action: The disk must be initialized using the MCR INI 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 reformat your device before running BAD.

BAD -- Invalid block number - numb

Explanation: You entered an invalid block number sequence. The value numb is the invalid sequence.

User Action: Type another value and reenter the command line. This message applies to the /MANUAL, /ALLOCATE or /UPDATE switches only.

BAD -- Invalid switch

Explanation: Self-explanatory

User Action: Reenter the command line 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 (for use only by Field Service).

User Action: Mount and process another cartridge.

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 line with the /OVERRIDE switch included.

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 and specified the right device.

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, then reenter the command line.

BAD -- ddu: Privilege violation

Explanation: An operation was attempted for a device that was mounted or allocated to another user.

User Action: Allocate another device, mount the device (if necessary), 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 volume.

User Action: Write the bad blocks count on the volume label.

BAD -- ddu: Unrecoverable error n

Explanation: An I/O error caused BAD to terminate. The value [n] is the error code number of the I/O error returned by the driver.

User Action: See the IAS/RSX-11 I/O Operations Reference Manual for an explanation of the error code number. If the same error persists, contact your DIGITAL Field Service representive.

BAD -- ddu: Vector not multiple of four

Explanation: Self-explanatory.

User Action: Reenter the command line including the /VEC switch with the proper value.

BAD -- ddu: Volume label incorrect

Explanation: The volume label entered with the /ALLOCATE switch did not match the label on the disk.

User Action: Reenter the command line using the correct volume label.

BAD -- ddu: Write locked

Explanation: The unit is write-locked.

User Action: Write-enable the unit and reenter the command line.

CHAPTER 7

BACKUP AND RESTORE UTILITY (BRU)

The Backup and Restore Utility (BRU) allows you to back up and restore Files-11 volumes. You can use BRU to transfer 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 backup files with BRU. In the process of copying, BRU also reorganizes and compresses files for efficient storage and access.

You can use BRU stand-alone as well as on-line. BRU64K is the stand-alone version on RSX-11M and BRUSYS is the stand-alone version on RSX-11M-PLUS.

Backup and restore operations take place on disk and magnetic 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, BRU provides command qualifiers that:

- Initialize disks
- Perform selective backup and restore operations
- Control tape processing such as density, length, rewinding, appending, and labeling according to the American National Standard (ANS) X.327-1978
- Perform volume and data checking
- Display information such as backup set names and file names

Section 7.9 contains examples of various BRU operations.

BRU can also be invoked through the DCL BACKUP command. For more information, see the <u>RSX-11M/M-PLUS Command Language Manual</u>.

7.1 ON-LINE BRU DISK AND TAPE DEVICE INFORMATION

BRU uses disk and tape volumes for its backup and restore operations. Input disks must be in Files-11 format. For tapes and multivolume backup disks, BRU has its own format.

BRU backs up from unmounted, mounted, and mounted foreign disk volumes.

BRU does not use the file system on input disks. However, when you are using a mounted volume for a backup operation, BRU checks the read access privileges of UFDs and files against the UIC under which BRU is running. To back up from a mounted disk volume that is in Files-11 format, you must specify the /MOUNTED qualifier. For unmounted disks or disks mounted foreign, no qualifier is necessary.

BRU also restores to unmounted and mounted disk volumes. Specify the /INITIALIZE qualifier to restore to an unmounted volume on an RSX-11M system or a volume mounted foreign on an RSX-11M or M-PLUS system. This qualifier initializes the volume to Files-11 format. To restore to a mounted Files-11 volume on either system, specify the /NOINITIALIZE qualifier to indicate to BRU that the disk is mounted and already in Files-11 format.

BRU backs up to and restores from tape volumes. The tapes must be unmounted or mounted foreign on RSX-11M or mounted foreign on RSX-11M-PLUS. No qualifier is necessary in either case.

Table 7-1 summarizes how to initialize and mount a volume on each system. BRU returns an error message for any wrong combination of conditions.

Table 7-1
Mounting and Initializing Volumes

System	Volume	Mount Status	Mandatory Qualifier
RSX-11M	Input disk	Not mounted	None
		Mounted Files-11	/MOUNTED
		Mounted foreign	None
	Output disk	Not mounted	/INITIALIZE
		Mounted Files-11	/NOINITIALIZE
		Mounted foreign	/INITIALIZE
	Input tape/ output tape	Not mounted	None
		Mounted foreign	None
RSX-11M-PLUS	Input disk	Mounted foreign	None
		Mounted Files-11	/MOUNTED
	Output disk	Mounted foreign	/INITIALIZE
		Mounted Files-11	/NOINITIALIZE
	Input tape/ output tape	Mounted foreign	None

For more detailed information on Files-11, refer to the IAS/RSX-11 I/O Operations Reference Manual.

With BRU, you can also specify that a disk volume contain up to 65,500 (64K-36) files. The default is the value assigned to the input disk. See the descriptions of the /HEADERS and /MAXIMUM qualifiers in Section 7.4 for more information.

7.1.1 Backup Sets

A backup set consists of all the data directed to or from a tape or disk volume (or volumes) during a single backup or restore operation. Physically, more than one backup set may be contained on a tape or disk, or a backup set can extend over several tapes or disks.

7.1.2 Tape Sets and Disk Sets

More than one backup set can be contained on a tape or disk, or a backup set can extend over several tapes or disks. In either case, the resulting output is called a tape or disk set.

A tape or disk set consists of the tape or disk volume (or volumes) to which data is transferred during a single backup operation.

7.1.2.1 **Tape and Disk Backup Operations** - There are several classifications of disk and tape backup operations to choose from. You can classify a backup operation by how much of the original tape or disk you are backing up and also by the format of the output medium.

The following backup operations are available to you:

- Full backup
- Selective backup
- Conventional backup
- Image backup (for disks only)

A full backup or a selective backup refers to the files you are backing up. A conventional backup or an image backup refers to the format of the output tape or disk. When you choose to do either a full backup or a selective backup, you may or may not also choose to do a conventional backup or an image backup.

The following sections explain the different kinds of backup operations.

7.1.2.2 **Full and Selective Backup Operation** - A full backup transfers all the original files to a backup volume (or volumes). Thus, a full backup ensures that you have a complete copy of the original disk.

A selective backup is a partial backup. If you do a selective backup, a subset of the original tape or disk is backed up. You select and identify the files to be backed up by UFD, date, or file specification.

One type of selective backup is the incremental backup. An incremental backup is a backup of files by date only. Incremental backups and selective backups are helpful over a short time span when a full backup would take up too much time or system resources.

7.1.2.3 Conventional Backup Operation

A conventional backup to tape copies the files from the original disk to a backup set on a BRU format tape. The backup set can span multiple tapes and you can append additional backup sets to a volume (or volumes). You cannot access the files in the backup set directly. Therefore, you must restore the backup set to disk before you can access the individual files contained in it.

A conventional backup to disk copies the files from the original disk to another Files-11 disk. You can access the files on the input disk or the output disk directly, eliminating the need to do a restore operation.

If the output disk already has a Files-11 structure, you can add the files from the original disk. If the output disk does not have a Files-11 structure, you must use BRU with the /INITIALIZE qualifier to create a Files-11 structure on the output disk.

7.1.2.4 Image Backup to Disk Operation

An image backup to disk copies the original disk to a container file on another Files-11 disk (or disks). Image backups are used for multivolume backup operations.

A disk container file has features similiar to features of tapes created by a conventional backup to tape. A disk container file can span multiple disks, and you can add several backup sets to a container file.

If you want to do a backup operation, you must specify the SAVE option with the /IMAGE qualifier. If you want to do a restore operation, you must specify the RESTORE option with the /IMAGE qualifier. In order to access the files in a backup set, you must restore the backup set to another disk. (See Section 7.4 for a description of command qualifiers.)

If the output disk already has a Files-11 structure, you can add the container file to it. If the output disk does not have a Files-11 structure, BRU creates its own structure on the disk with the container file.

7.1.3 Multivolume Tape and Disk Operations

When you specify a magnetic tape drive as the output device or when you specify a disk for an image backup in a BRU operation, BRU transfers the data contents of the input disk to the tape or disk on the drive. This data transfer often involves more than one reel of tape or more than one disk and may use more than one tape or disk drive.

You can specify more than one type of drive in a single BRU command. However, although you can specify up to eight drives per command, you can specify an individual tape or disk drive only once.

If the number of volumes required exceeds the number available, BRU lets you replace tapes or disks on the specified drive in round-robin fashion.

You can only use all 7-track or all 9-track tapes in a multivolume tape set. You cannot switch from one track type to the other within the set.

You can only use the same disk types when backing up to multiple disk in image mode. You cannot mix disks.

7.1.4 Supported Devices

Table 7-2 lists all the devices that on-line BRU supports. The disks from DB through EM are all block-structured devices.

Table 7-2
Devices Supported by On-Line BRU

Mnemonic	Type
DB	RH11/RP04/RP05/RP06 or RH70/RP04/RP05/RP06 disk pack
DD	TU58 cassette (DECtape II)
DF	RF11/RS11 fixed head disk
DK	RK11/RK05/RK05F cartridge pack
DL	RL11/RL01/RL02 cartridge disk
DM	RK611/RK06/RK07 cartridge disk
DP	RP11/RP02/RP03 disk pack
DR	RH70/RM03/RM05/RM80/RP07 or RH11/RM02 disk pack
DS	RH11/RS03/RS04 or RH70/RS03/RS04 fixed head disk
DT	TC11/TU56 DECtape
DU	RA80/RA60/RA81/RC25/RD51/RX50 disk
DX	RX11/RX01 floppy disk
DY	RX211/RX02 floppy disk
EM	ML11 electronic memory
MF	TM78/TU78 magnetic tape
MM	RH11/TM02-03/TE16/TU16/TU45/TU77 and RH70/TM02-03/TE16/TU16/TU45/TU77 9-track magnetic tape
MS	TS11/TSV05/TU80 magnetic tape
MT	TM11/TE10/TU10 7- or 9-track magnetic tape or TS03 9-track magnetic tape

7.2 COMMAND LINES

This section describes the rules for entering command lines for BRU. The section defines the command line syntax and describes prompts, command line parameters, and command qualifiers.

7.2.1 Command Line Syntax

BRU command lines have a maximum length of 256(10) characters except in one case of using continuation lines (see Section 7.2.2.2). The general syntax of the BRU command line is:

```
BRU>/qualifier(s) indevice1:,...[filespec,...] outdevice1:,...
```

However, if you type only

```
>BRU(RET)
```

the following three prompts appear on the terminal:

```
FROM:

TO:

INITIALIZE [Y/N]:
```

FROM

Requests that you enter the name (or names) of the devices on which the input volume (or volumes) reside. The names should be in the form specified in the description of the command line parameters (see Section 7.2.2). If you want, you may specify a UIC, but this is not required.

TO

Requests that you enter the name (or names) of the output devices. The names should be in the form specified in the description of the command line parameters (see Section 7.2.2). The UIC should not be specified in the command line because BRU tries to copy the entire UIC.

INITIALIZE [Y/N]

Enter Y (for YES) if you want to initialize the output volume.

Enter N (for NO) if you do not want to initialize the output volume.

There is no default answer. You must respond with either Y or N.

7.2.2 Command Line Parameters

The parameters for BRU command lines are qualifiers, device specifications, and file specifications.

/qualifier(s)

Specifies any of the command qualifiers listed in Section 7.3. If two or more qualifiers are specified, they must be contiguous, that is, separated with a slash only. The qualifiers can appear in any order.

You can use a shorter form of a qualifier as long as it is unique. All BRU qualifiers are unique to three characters. For example:

```
BRU>/REW/INI/OUT:BACKUP MM0: DB0:
```

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 and for disks (if you are using the /IMAGE qualifier) you can specify more than one input device and more than one type of drive. Devices are specified in the form:

```
dd[nn]:
```

The variable dd represents the device mnemonic and nn represents the octal unit number associated with that device. The unit number is 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.

Separate the device specifications with commas when you specify more than one device.

filespec

Indicates the file specification used to select particular files or categories of files to be backed up or restored. The file specification is in the following format:

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

You can specify up to 16(10) file specifications in each command line. When you enter a command line with no file specifications, all the files on the input volume are copied to the output volume.

Files can be backed up or restored selectively by UFD, file name, file type, 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 (*). The wildcard 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 and for disks (if you are using the /IMAGE qualifier) you can specify more than one output device. The form is the same as for indevice (described previously).

7.2.2.1 **Wildcards in Input Specifications** - The following wildcard (*) features are provided for input file UFD specifications:

```
[*,*] means all group,member combinations.
[n1,*] means all member numbers for group n1.
[*,n2] means all group numbers for member n2.
```

BRU also supports the wildcard in the remaining elements of a file specification: file name, file type, and version number. BRU generally follows the rules for use of wildcards (see the RSX-11M/M-PLUS MCR Operations Manual), except in the following two instances:

- When you omit a file specification element, BRU treats the omitted element as if it were a wildcard. For example, when you specify only file name and file type in a file specification, all version numbers are transferred in the backup or restore operation. However, when you omit the UFD, it defaults to your current UIC.
- When you specify particular UFDs on a command line but do not specify file names and/or file types, all the files in those UFDs are transferred in the backup or restore operation. That is, you do not have to specify [ufd]*.*.

7.2.2.2 **Continuation Lines** - BRU command lines have a maximum length of 256(10) characters. BRU allows you to continue a command line onto more than one line by using a hyphen (-) as the continuation character.

On RSX-11M systems, BRU supports continuation lines only when you invoke BRU and then respond to the BRU> prompt. BRU does not support continuation lines when run from an indirect command file or when you enter the command line on the same line as the one on which you invoked BRU.

On RSX-11M-PLUS systems, BRU supports continuation lines under all circumstances. However, when you use BRU from the MCR command level, the total number of characters for the initial command line and any subsequent continuation lines cannot exceed 80.

Section 7.9 gives examples of continuation lines on RSX-11M and RSX-11M-PLUS.

7.3 SUMMARY OF COMMAND QUALIFIERS, OPTIONS, AND DEFAULTS

Table 7-3 lists the command qualifiers available for backup and restore operations. For a detailed explanation of each qualifier, see Section 7.4. Examples of using various qualifiers are given in Section 7.9.

Table 7-3
Summary of BRU Command Qualifiers

Command Qualifiers	Options	Default
/APPEND		None
/BACKUP_SET:name		Volume name of the disk being backed up

(continued on next page)

Table 7-3 (Cont.)
Summary of BRU Command Qualifiers

Command Qualifiers	Options	Default
/BAD:	MANUAL AUTOMATIC OVERRIDE	BAD:AUTOMATIC
/BUFFERS:number		Number of FCBs from the input disk
/COMPARE		None
/CREATED:	BEFORE:(dd-mmm-yy hh:mm:ss) BEFORE:dd-mmm-yy BEFORE:hh:mm:ss AFTER:(dd-mmm-yy hh:mm:ss) AFTER:dd-mmm-yy AFTER:hh:mm:ss	Current date
/DENSITY:number		Default density of drive
/DIRECTORY		None
/DISPLAY		None
/ERRORS:number		/ERRORS:25.
/EXCLUDE		None
/EXTEND:number		Number of blocks from the input disk
/HEADERS:number		Number of headers allocated to the input volume
/IMAGE:	SAVE RESTORE	None
/INITIALIZE		None
/INVOLUME:name		None
/LENGTH:number		The length of the output tape

(continued on next page)

Table 7-3 (Cont.)
Summary of BRU Command Qualifiers

Command Qualifiers	Options	Default
/MAXIMUM:number		Maximum number of files allowed on input the volume
/MOUNTED		None
/NEW_VERSION		/NOSUPERSEDE
/NOINITIALIZE		None
/NOPRESERVE		None
/NOSUPERSEDE		/NOSUPERSEDE
/OUTVOLUME:name		Input disk volume name
/POSITION:	BEGINNING MIDDLE END BLOCK:number	Index file position on the input disk
/PROTECTION:	SYSTEM:value OWNER:value GROUP:value WORLD:value	Protection of the input disk
/REVISED:	BEFORE:(dd-mmm-yy hh:mm:ss) BEFORE:dd-mmm-yy BEFORE:hh:mm:ss AFTER:(dd-mmm-yy hh:mm:ss) AFTER:dd-mmm-yy AFTER:hh:mm:ss	Current date
/REWIND		None
/SUPERSEDE		/NOSUPERSEDE
/TAPE_LABEL:label		None
/UFD		None
/VERIFY		None
/WINDOWS:value		Number of mapping pointers on the input disk

7.3.1 Command Qualifier Functions

When you initialize a disk using the BRU /INITIALIZE qualifier, use the following qualifiers to specify various characteristics for the output disk.

/BAD	/NOPRESERVE
/BUFFERS	/OUTVOLUME
/EXTEND	/POSITION
/HEADERS	/PROTECTION
/MAXIMUM	/WINDOWS

The /MOUNTED command qualifier allows you to copy files from a mounted disk.

The following command qualifiers allow you to copy files to a mounted disk with various results. Note that these qualifiers are not available in stand-alone BRU (see Section 7.5).

/NEW_VERSION

/NOINITIALIZE

/NOSUPERSEDE

/SUPERSEDE

/UFD

The following command qualifiers allow you to backup or restore data according to:

- File specification
- Date and time of creation
- Date and time of revision

The qualifiers are:

/CREATED

/EXCLUDE

/REVISED

The following qualifiers allow you to control backup and restore tape processing:

/APPEND	/LENGTH
/BACKUP_SET	/REWIND
/DENSITY	/TAPE_LABEL
/ERRORS	

The following command qualifiers allow you to detect differences between data on the input volume and data on the output volume:

/COMPARE

/INVOLUME

/VERIFY

The /DIRECTORY and /DISPLAY command qualifiers display information about the files being transferred.

The /IMAGE command qualifier is for both disk image backups and all restore operations.

7.4 DESCRIPTIONS OF COMMAND QUALIFIERS

The following paragraphs describe the BRU command qualifiers in detail.

/APPEND

Directs BRU to append a backup set from the input disk volume to the last backup set on the output tape, or on the output disk if you are using the /IMAGE qualifier.

If the output tape was positioned at the beginning the /APPEND qualifier causes BRU to skip to the logical end-of-tape before it writes the new backup set. BRU searches the output volume for the last logical end-of-file.

If the output tape is already positioned at the logical end-of-tape, /APPEND causes BRU to start writing where the device is currently positioned.

If the output tape is not positioned at the beginning, or if it is not at the logical end-of-tape, you can use the /REWIND qualifier with /APPEND to rewind the tape and then space forward until the logical end-of-tape.

If the tape is a continuation tape (that is, not the first tape in a set) or if the last backup set does not end on the tape, BRU displays an error message.

If the output device is a disk and you are using the /IMAGE qualifier, /APPEND causes BRU to check the container file header for the logical end-of-file on the output disk. BRU then starts writing at the logical end-of-file.

If the output disk is a continuation disk (that is, not the first disk in a set) or if the last backup set does not end on the disk, BRU displays an error message.

You cannot use the /APPEND qualifier during a backup operation to a mounted disk.

/BACKUP_SET:name

Specifies the name of the backup set (refer to Section 7.1.1) to be placed on tape or disk. For tape and for an unmounted disk, the default name is the volume name of the disk being backed up. This name may be up to 12(10) characters long. For a mounted input or output disk during an image backup or restore operation, you can specify the full backup set file name with the /BACKUP_SET qualifier. If you do not specify the file name, the default is [0,0]BACKUP.SYS.

When applied to an output 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.

When this qualifier is applied to an input tape volume, BRU searches the first tape for the specified backup set name. If you do not specify a backup set name with the input volume, BRU restores the first backup set it finds on the tape. You can restore several sequential backup sets from the same tape without rewinding the tape between BRU operations. BRU does not rewind the first device in a backup set unless you specify the /REWIND qualifier.

When this qualifier is applied to an input disk volume, BRU searches the entire disk for each backup set you specify. Each backup set is then restored in the order of the backup set names you provided.

/BAD:AUTOMATIC
OVERRIDE
MANUAL

The /BAD qualifier creates the file BADBLK.SYS on the output disk. The qualifier is used with the /INITIALIZE qualifier during tape-to-disk or disk-to-disk operations.

For complete information on how to use the options for /BAD, see Section 7.6. The following are summary descriptions only.

For last-track devices, the AUTOMATIC option 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. AUTOMATIC is the default option.

The OVERRIDE option 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 and ignores the manufacturer-written information.

The MANUAL option accepts the addresses of bad blocks you enter interactively at your terminal. It also specifies that BRU use either the manufacturer-written bad block information and the software-detected bad sector file (for last-track devices) or the bad block descriptor block (for nonlast-track devices) to create BADBLK.SYS.

/BUFFERS:number

Specifies the default number of directory File Control Blocks (FCBs) on each volume. The FCBs are stored in memory by the Ancillary Control Processor (ACP) when the volume is mounted. The more FCBs there are stored in memory, the faster that files contained in heavily used directories are found. The default number of buffers is the same as 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 line 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, /NOINITIALIZE, and /APPEND qualifiers.

/COMPARE Output

When the compare operation detects differences, it displays a message at your terminal. The compare operation always displays the mnemonic of the device on which the difference was detected and the type of record in which the difference was encountered (a control record, a header record, or a data record).

If the record type is a header record, the compare operation also displays the file-ID for the file. If the record type is a data record, the compare operation also displays the file-ID, the Logical Block Number (LBN) of the block in error, and the name of the file if it is available.

/CREATED:BEFORE:(dd-mmm-yy hh:mm:ss)
BEFORE:dd-mmm-yy
BEFORE:hh:mm:ss
AFTER:(dd-mmm-yy hh:mm:ss)
AFTER:dd-mmm-yy
AFTER:hh:mm:ss

Backs up or restores files created before or after the specified date and/or time.

If you use the BEFORE option, BRU copies any files created before the specified date and/or time.

If you use the AFTER option, BRU copies any files created on or after the specified date and/or at or after the specified 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 the current date as the default. If you specify only a date, the time defaults to 00:00.

/DENSITY:number

Specifies the density (bpi) at which BRU writes to tape. The following chart shows the values you can specify.

Drive	Default Density	Optional Density
TU10/TE10	800	None
TU16/TE16	800	1600
TU45	800	1600
TU77	800	1600
TS11	1600	None
TSV05	1600	None
TU78	6250	1600
TU80	1600	None

If you specify /DENSITY with /APPEND, you must specify the density at which the existing tape data was written. For example, if the tape was first written at a density of 800 bpi, you must specify a density of 800. If you specify a density other than the original density, BRU displays a 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 at your terminal the backup set names or files on the specified tape or disk volume. In a multivolume tape set, the directory is on the first tape of the set. In a multivolume disk set, the directory is on the first disk of the set.

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  2-JAN-83
VOL1    BACKUP2 LABEL1  3-JAN-83

BRU>/DIRECTORY DU0:

VOL1    BACKUP1 LABEL1  13-JUN-83
VOL2    BACKUP1 LABEL2  14-JUN-83
```

Using /DIRECTORY to Display File Names:

To display the names of files in a backup set, enter the backup set name with /DIRECTORY in the form shown below.

If the backup set is not on the tape or disk, BRU halts execution and displays a message at your terminal.

An example with tape follows:

```
>
>RUN BRU
BRU>/BACKUP_SET:23MAY82A/DIRECTORY MM1:
VOL1.   23MAY82A        HWHDOC  13-JUN-82 23:37:11
[000,000]
[303,013]
27DECE.LST;1
2JANA.LST;1
18JANC.LST;1
4JANA.LST;2
```

```
25DECA.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]

RSX11M.STB;3
[005,054]
[306,006]
APNDXB.MAC;1
BRU - COMPLETED ON MM1:

BRU> CTRL/Z
>
```

/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

Terminates a restore operation after the specified number of nonfatal tape read errors. The range for number is 0 to 65535. The default number of errors before termination is 25(10).

/EXCLUDE

Backs up or restores all of the files on the tape or disk except the files specified on the command line.

/EXTEND:number

Specifies the default number of blocks by which a file is extended when that file has exhausted its allocated space. This value is used by an ACP when the volume is mounted. The default is the number of blocks from the input disk.

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 bitmap file. (The storage bitmap 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 default is the number of headers allocated to the input volume.

The /HEADERS qualifier is used with the /INITIALIZE qualifier during tape-to-disk or disk-to-disk operations.

If you want to copy files from a disk with a single-header index file (structure level 401) to a disk with a multiheader index file (structure level 402), specify a number of file headers with /HEADERS and a number of files with the /MAXIMUM qualifier that are both large enough to make the output disk contain a multiheader index file. See the description of the INI command in the RSX-11M/M-PLUS MCR Operations Manual for a table of maximum and default values.

/IMAGE:SAVE
:RESTORE

Specifies that you want to do a multiple disk-to-disk backup or restore operation. If you are doing a backup operation, you must specify the SAVE option on the command line. If you want to do a restore operation, you must specify the RESTORE option on the command line.

If you want to do a backup operation, you must use this qualifier when you create the backup file that represents the image copy of the input disk or disks. For example, this qualifier must be used when you copy a large disk to several small disks, or if you copy several small disks to a mounted large disk.

If you want to do a restore operation, you must use this qualifier when restoring from a backup file that represents the image copy of the original disk.

/INITIALIZE

Specifies that you want to initialize the output disk during a tape-to-disk or disk-to-disk operation. Initialization places a Files-11 structure on the disk, including the boot block, the home block, and such files as INDEXF.SYS, BADBLK.SYS, and 000000.DIR.

The conditions for initializing an output volume differ between RSX-11M and RSX-11M-PLUS. BRU returns a privilege violation if the conditions are not met satisfactorily (see Table 7-1).

On RSX-11M, the volume must be unmounted or mounted foreign.

On RSX-11M-PLUS, the volume must be mounted foreign.

Along with the /INITIALIZE qualifier, you can specify the following qualifiers when you are initializing a disk:

/BAD	/NOPRESERVE
/BUFFERS	/OUTVOLUME
/EXTEND	/POSITION
/HEADERS	/PROTECTION
/MAXIMUM	/WINDOWS

If you do not specify any of these qualifiers, BRU defaults to the characteristics of the input volume except for the /BAD qualifier and the /NOPRESERVE qualifier. For the /BAD qualifier, the default is AUTOMATIC. For the /NOPRESERVE qualifier, there is no default.

/INVOLUME:name

Specifies the volume label of the input disk. This name can be up to 12(10) characters long.

For disk-to-tape or disk-to-disk operations, the /INVOLUME qualifier directs BRU to look for the volume label of the input volume 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 check the volume label of the disk that is stored in the backup set on tape or in the image backup set file on the disk.

/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 is 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 in the volume's index file. (BRU supports up to 65,500 files on a volume.) 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 that is different in size from the input disk.

If you want to copy files from a disk with a single-header index file (structure level 401) to a disk with a multiheader index file (structure level 402), specify a number of files with /MAXIMUM and a number of file headers with the /HEADERS qualifier that are both large enough to make the output disk contain a multiheader index file. See the description of the INI command in the RSX-11M/M-PLUS MCR Operations Manual for a table of maximum and default values.

/MOUNTED

Allows you to back up files from a disk that is mounted as a Files-11 volume (by means of the MCR or DCL MOUNT commands). If you use the /MOUNTED qualifier when the input device is a tape, BRU issues a syntax error.

BRU does not use the file system to read files from the input disk. Instead, it issues logical queue I/Os (such as IO.RLB). To issue these QIOs to a mounted Files-11 disk, BRU must be built as a privileged task (PR:0). (BRU does not have to be privileged for operations on unmounted volumes or volumes mounted foreign.) However, when you are restoring to a mounted disk (and you have specified the /NOINITIALIZE qualifier), BRU uses the file system to access the output disk. Therefore, a restore operation to a mounted disk is slower than a restore to an unmounted disk.

BRU must also be privileged to back up a disk that is being accessed by other users. The /MOUNTED qualifier must be specified in the command line.

When backing up files from a mounted volume, disk activity (changes to or addition or deletion of files) 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 in the file on the output volume.

- 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 reused 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 or the directory in the backup set file on the disk 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.

- If the file or data are being changed during a transfer operation, BRU will not be able to verify the accuracy of the operation. Do not attempt a verify operation in this case.

  Note that this restriction also includes the file being used by the error logger. The error logger file changes when any hardware errors occur, which can cause the verify operation to fail. To ensure that the verify operation succeeds, switch the error logger file to a different disk or exclude it with the /EXCLUDE qualifier.

/NEW_VERSION

Resolves file specification conflicts that occur during restore operations and during backups to a mounted disk using the /IMAGE:SAVE qualifier. When a file already exists on the output disk volume, /NEW_VERSION creates a new version of the file.

/NOINITIALIZE

Specifies that you do not want to initialize the output disk because it is already in Files-11 format. The output disk must be mounted as a Files-11 volume. You cannot enter any of the initialization qualifiers when you specify /NOINITIALIZE. If you enter any of these qualifiers, BRU issues an error message.

/NOPRESERVE

Specifies that you do not want to preserve file-IDs (file-IDs are generally preserved). If you specify the /NOPRESERVE qualifier, BRU suppresses the message that file-IDs are not being preserved. Note that in restoring to a mounted disk, not preserving file-IDs is BRU's default action. /NOPRESERVE is used only with the /INITIALIZE qualifier.

When file-IDs are not preserved, BRU assigns new file-IDs, incrementing them sequentially.

/NOSUPERSEDE

Specifies that when file specifications on the mounted output disk are identical to those on the input volume, the file on the input volume is not transferred. That is, the file on the output disk is not superseded by the file on the input volume. /NOSUPERSEDE is the default.

/OUTVOLUME:name

Specifies the volume label of the output disk. This label can be up to 12(10) characters long.

For disk-to-tape backup operations, the name of the input disk volume stored on the output tape volume is changed to the name specified with the /OUTVOLUME qualifier.

For tape-to-disk 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:

- In backup operations, the input disk volume name is used as the volume name stored on the output tape volume.
- In restore operations, the volume name stored on the input tape volume is used as the name of the output disk volume.
- 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

Specifies the location of the index file on the output disk volume being initialized, usually to minimize access time. The BEGINNING, MIDDLE, and END options specify the beginning, middle, and end of a volume. The BLOCK:number option specifies a block number where the index file is to be placed. The BEGINNING or END option is generally used only when a disk mostly contains large contiguous files. MIDDLE is recommended to minimize access time.

When you use the BLOCK:number option, the number is decimal by default (the period is optional). To specify an octal number, place a pound sign (#) in front of the number. If there are any conflicts, BRU issues a warning message.

When you do not use the /POSITION qualifier, BRU places the index file in the same location as that on the input volume.

/PROTECTION: SYSTEM:value
OWNER:value
GROUP:value
WORLD:value

Specifies the default protection status for all files created on the output volume being initialized. This protection value does not apply to files being transferred by BRU, but rather to subsequent files created on the output volume by an ACP when the volume is mounted. If you do not specify any values, they default to the protection values of the input disk.

The protection value can be R(ead), W(rite), E(xtend), or D(elete), or any combination of the four. See the RSX-11M/M-PLUS MCR Operations Manual for an explanation of file protection.

/REVISED: **BEFORE:(dd-mmm-yy hh:mm:ss)**
BEFORE:dd-mmm-yy
BEFORE:hh:mm:ss
AFTER:(dd-mmm-yy hh:mm:ss)
AFTER:dd-mmm-yy
AFTER:hh:mm:ss

Backs up or restores files revised or created on, before, or after the specified date and time.

If you use the BEFORE option, BRU copies any files revised or created at or before the specified date and/or time.

If you use the AFTER option, BRU copies any files revised or created on or after the specified date and/or at or after the specified time.

As with the /CREATED qualifier, if you specify both a date and time, the date and time must be enclosed in parentheses. If you specify only a date or time, the parentheses are not necessary. If you specify only a time, BRU uses the current date as a default. If you specify only a date, the time defaults to 00:00.

/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 a backup set.

When specified with the /APPEND qualifier, BRU rewinds the output tape and then searches for the logical end-of-tape before executing the backup operation.

/SUPERSEDE

Specifies that when file specifications on the mounted 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.

For an /IMAGE operation, if you create a backup set file on a mounted volume and a file with the same name exists, /SUPERSEDE replaces this file.

/NOSUPERSEDE is the default.

/TAPE_LABEL:label

Specifies the 6-character volume identifier on the American National Standard (ANS) X.327-1978 label to be placed on a tape during a backup operation or to be compared with the label on the tape for append and restore operations. This allows you to check that you are using the correct tape.

/UFD

Directs BRU to create UFDs (if they do not already exist) on a mounted output volume, then copy into it the files from the same UFD on the input volume. If you do not specify /UFD, BRU does not copy the files. /UFD is used only with the /NOINITIALIZE qualifier.

/VERIFY

Copies data from the input volume, performs a compare operation between the input volume and the output volume after the transfer, and reports any differences.

/WINDOWS:number

Specifies the default number of mapping pointers to be allocated for file windows when initializing an output disk. 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 for the input disk.

Choosing a large number of mapping pointers may speed up file access. However, a large file window also 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 the IAS/RSX-11 I/O Operations Reference Manual for more information.

7.5 STAND-ALONE BRU

You can also run BRU stand-alone. On RSX-11M, the stand-alone system is called BRU64K. On RSX-11M-PLUS, it is BRUSYS. The difference between the BRU task contained in these stand-alone systems and the on-line BRU described in the preceding sections is that stand-alone BRU does not support a restore operation to a mounted volume. Therefore, BRU will always initialize the output disk volume. There is no /INITIALIZE qualifier and the BRU task does not ask if you want to initialize the output volume.

Other qualifiers that cannot be used in the stand-alone BRU systems are /NEW_VERSION, /NOINITIALIZE, /NOSUPERSEDE, /SUPERSEDE, and /UFD.

Unlike other stand-alone systems, the BRU systems contain other tasks besides the BRU task. You invoke these other tasks before invoking the BRU task. (One task, CNF, is first invoked automatically when you boot the stand-alone BRU system.)

On RSX-11M, BRU64K contains two other tasks besides the BRU task -- BAD and CNF. BAD is the Bad Block Locator Utility (described in Chapter 6) and CNF is the Stand-alone Configuration and Disk Sizing Program (described in this section). You invoke BRU and BAD with the MCR RUN command. CNF is invoked automatically when you boot BRU64K. (You can use CNF again by also invoking it with the RUN command.)

On RSX-11M-PLUS, BRUSYS contains four other tasks besides the BRU task. These other tasks are FMT, BAD, DSC, and CNF. FMT is the Disk Volume Formatter Utility (described in Chapter 5) and DSC is the Disk Save and Compress Utility Program (described in Chapter 8). You invoke BRU, FMT, BAD, and DSC with the MCR RUN command. As on RSX-11M, CNF is invoked automatically when you boot the system. (You can use CNF again by also invoking it with the RUN command.)

If you need to use FMT or BAD, you must run them before running BRU. When you are finished with these other tasks, issue the RUN BRU command to begin using BRU.

After you boot the stand-alone BRU system, the CNF task runs automatically. It lists the switches available for your use and then prompts you for the devices you will be using. It is recommended that you first specify /DEV to find out the status of the devices on your system. For example:

```
     Enter first device:  /DEV

      Device                  CSR       Vector        CSR Status

      DB                      176700     254          Present
      DK                      177404     220          Present
      DL                      174400     160          Not Present
      DM                      177440     210          Present
      DP                      176714     300          Present
      DR                      176300     150          Present
      DU                      172150     154          Not Present
      MF  FOR=0               175400     260          Not Present
      MM  FOR=0               172440     224          Present
      MS                      172522     330          Not Present
      MT                      172522     320          Not Present

     Enter first device:
```

You can also use CNF and its switches to set the CSR and vector addresses of devices present on your system or to change the default formatter number (FOR=n) for some of the magnetic tape devices. For example:

```
     Enter first device: DM2:/CSR=177450/VEC=274

     Enter second device: DL:
```

The CNF switches are:

/CSR=nnnnnn

Changes the default CSR for the device.

/DEV

Lists the default CSR and vector addresses for all of the devices. It is recommended that you use this switch first to find out what devices are present on your system.

/FOR=n

Changes the default formatter number for some of the magnetic tape devices. The switch is valid for only the MF:- and MM:-type devices. The initial default for n is 0.

/VEC=nnn

Changes the default vector for the device.

7.5.1 Locating and Booting Stand-Alone BRU

On RSX-11M, the BRU64K system on the distribution kits requires 64K words of memory to run the three tasks (BRU, BAD, CNF). The BRU64K system image (BRU64K.SYS) and symbol table (BRU64K.STB) are located in UFD [1,51] on the following disk volumes:

```
BIG DISK KIT  - RSXM35
RK06/RK07 KIT - CLISRC
RL01/RL02 KIT - RLUTIL
```

On RSX-11M-PLUS, the BRUSYS system on the distribution kit requires 124K words of memory to run the five tasks (BRU, BAD, FMT, DSC, CNF). The BRUSYS system image and symbol table are located in UFD [6,54] on the distribution disk. The name of the disk is RSX11MPBL15.

On both RSX-11M and RSX-11M-PLUS, you can bootstrap the stand-alone BRU system in one of two ways:

1. Software boot stand-alone BRU by using the privileged MCR BOOT command as follows:

   For RSX-11M mapped systems

   ```
   >INS $BOO
   >BOO [1,51]BRU64K
   ```

   For RSX-11M-PLUS

   ```
   >INS $BOO
   >BOO [6,54]BRUSYS
   ```

2. Hardware boot stand-alone BRU following the hardware bootstrap procedure for your processor.

   To create a hardware-bootable stand-alone BRU tape from the distribution disk, use the Virtual Monitor Console Routine (VMR) to save the system image to tape as follows:

   For RSX-11M

   ```
   >SET /UIC=[1,54]
   RUN [1,54]VMR
   ENTER FILENAME: BRU64K
   VMR>SAVE MT:BRU64K
   VMR> CTRL/Z
   ```

   For RSX-11M-PLUS

   ```
   >SET /UIC=[6,54]
   RUN VMRM41
   ENTER FILENAME: BRUSYS
   VMR>SAVE MT:BRUSYS
   VMR> CTRL/Z
   ```

   This tape contains a hardware-bootable image of the stand-alone BRU system. (See the RSX-11M/M-PLUS System Management Guide for information on VMR.)

7.6 ON-LINE BRU BAD BLOCK PROCESSING

After you have formatted a disk with the Disk Volume Formatter Utility (FMT; see Chapter 5) and marked any bad blocks on it with the Bad Block Locator Utility (BAD; see Chapter 6), you can initialize it with BRU.

If you specify the /BAD qualifier with the /INITIALIZE qualifier, BRU uses the bad block information from running BAD on the disk to create the file BADBLK.SYS. This file has allocated to it all of the bad blocks on the disk so that other files will not try to use them.

The /BAD qualifier has three options: AUTOMATIC, which is the default option, OVERRIDE, and MANUAL. The following sections describe how to use these options.

7.6.1 Using the AUTOMATIC Option

The AUTOMATIC option 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.

7.6.2 Using the OVERRIDE Option

The OVERRIDE option 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 the BAD /OVR switch 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.

7.6.3 Using the MANUAL Option

The MANUAL option accepts the addresses of bad blocks you enter interactively at your terminal. It also specifies that BRU use either the manufacturer-written bad block information and the software-detected bad sector file (for last-track devices) or the bad block (for nonlast-track devices) descriptor block to create BADBLK.SYS. If there is no software-written bad block information, a message will be displayed informing you that BAD has not processed the disk.

When you specify /BAD:MANUAL, BRU issues 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[.]]
```

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 number sign (#). If you do not specify count, it defaults to 1.

If you enter a bad block that is already in the bad block file, BRU generates an error message.

To get a list of the LBNs you have entered so far, type one slash (/). To copy the LBNs into BADBLK.SYS, type two slashes (//).

When you have finished entering bad blocks, press the RETURN key to return to BRU command level.

7.7 USING BRU TO COPY DISKS CONTAINING SYSTEM IMAGES

When you copy a bootable system disk to a disk of the same controller type, BRU automatically produces a bootable output disk for you.

If, however, you are copying an unsaved (virgin) system or copying a saved system to a smaller disk or to a disk of a different controller type, you can use the procedures described in the following sections to ensure that BRU produces a bootable output disk.

7.7.1 Copying an Unsaved (Virgin) System

In an unsaved system, installed tasks are pointed to by the physical LBN of the task image on the disk. When you copy an unsaved system with BRU, BRU assigns new LBNs for the task images on the output disk. Therefore, if you want to be able to software boot the copied system, you must first use VMR to remove and reinstall any tasks. (VMR, the Virtual Monitor Console Routine, is described in the RSX-11M/M-PLUS System Management Guide.)

7.7.2 Copying a Saved System

In a saved system, installed tasks are pointed to by the file-ID of the task image on the disk. To copy a saved system to a smaller disk or to a disk of a different controller type, use the following procedures.

7.7.2.1 **Copying to a Smaller Disk** - If you want to copy a disk with a saved system to a smaller disk, the most common method is to first use the /MAXIMUM and /HEADERS qualifiers (described in Section 7.4) to decrease the size of the index file. BRU is then unable to preserve the file-IDs of the files, so you must use VMR to remove and reinstall any tasks in the copied system image (the image on the output disk).

7.7.2.2 **Copying to a Different Controller Type** - If you have used BRU to copy a hardware-bootable disk to a disk of a different controller type and you want the output disk to also be hardware-bootable, you must use the MCR BOOT command to boot the saved system on the output disk. Then you use the MCR SAVE /WB command to write the correct boot block on the output disk.

On an RSX-11M-PLUS system, you can copy a DB:-, DM:-, DU:-, EM:- or DR:-type disk to any other of these controller types without having to re-SAVE the system because the boot block for these devices is common on RSX-11M-PLUS. Use the following command to write the correct boot block on the output disk:

```
     MCR SAV /WB
```

7.8 BRU FILE TREATMENT

The following sections describe how BRU treats file dates, file headers, file synonyms, and lost files.

7.8.1 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 a disk being initialized, and also when the /UFD qualifier is specified, the creation date of the UFD is the date on which BRU created it.

7.8.2 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 correctly reflect the file's size.

7.8.3 File Synonyms

File synonyms are files that have different names but share the same file-ID and data. They can be created with the PIP utility but also occur when a file-ID from a deleted file is reused in a UFD that BRU has not yet copied. 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.

7.8.4 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 File Structure Verification Utility (VFY) with its /LOst switch before using BRU to back up the disk (VFY is described in Chapter 9).

7.9 EXAMPLES

This section gives examples of various BRU operations and command lines. Note that the qualifiers used in the command lines have been truncated to three characters. All of the BRU qualifiers are unique to three characters.

Examples 1 through 6 and examples 11 and 12 also show informational messages that BRU returns during some operations. The remaining examples do not include these messages.

The following list is a summary of the operations and command lines shown in the examples:

1. Disk-to-tape backup (with verification) and tape-to-disk restore operations.

2. Disk-to-disk backup operation.

3. Disk-to-disk backup operation including changing the maximum number of files and initial header allocation for the output disk.

4. Disk-to-disk multivolume backup operation.

5. Disk-to-disk multivolume restore operation.

6. Disk-to-disk multivolume backup (with appending) and disk to disk restore operation.

7. Disk-to-tape incremental backup operation (by date) with tape verification.

8. Disk-to-tape selective backup operation (by file specification).

9. Mounted disk-to-tape backup and tape-to-mounted disk restore operations.

10. Disk-to-tape backup (with appending) and tape-to-mounted disk restore operations.

11. Exclusion of certain files during a backup operation.

12. Manually entering bad blocks and displaying them.

13. Continuation command lines on RSX-11M and RSX-11M-PLUS.

14. Continuation command lines on RSX-11M-PLUS only.

Example 1 shows how to use BRU to back up an entire disk volume onto two 1600 bpi magnetic tapes and then how to restore the disk. For the backup operation, BRU verifies the output volumes as part of the operation. (Verifying volumes is specified with the /VERIFY qualifier.) If files do not verify, BRU returns an error message.

To back up DM2: onto the magnetic tapes on MM0: and MM1:, use the following command line:

```
BRU>/DEN:1600/VER DM2: MM0:,MM1:(RET)
BRU - STARTING TAPE 1 ON MM0:

BRU - END OF TAPE 1 ON MM0:

BRU - STARTING VERIFY PASS 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 - STARTING VERIFY PASS TAPE 2 ON MM1:

BRU - END OF TAPE 2 ON MM1:

BRU - COMPLETED

BRU> (CTRL/Z)
>
```

To restore the entire disk and rewind the first input tape, use the following command line (the /INI qualifier specifies that DM2: will be initialized before the restore operation begins):

```
BRU>/REW/DEN:1600/INI MM:,MM1: DM2:(RET)

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 - STARTING TAPE 2 ON MM1:

BRU - END OF TAPE 2 ON MM1:

BRU - COMPLETED

BRU>(CTRL/Z)
>
```

Example 2 shows how to do a disk-to-disk backup operation for an entire disk. The characteristics of the output disk default to those of the input disk. This operation (and every other BRU operation) can be done in two ways.

```
>BRU/INI DM: DM1:(RET)
BRU - COMPLETED
```

or

```
>BRU(RET)
BRU>(RET)
FROM: DM:(RET)
TO:   DM1:(RET)
INITIALIZE OUTPUT DISK [Y/N]:Y(RET)
BRU - COMPLETED

BRU>(CTRL/Z)
```

These two command lines tell BRU to initialize the output disk (DM1:) and then back up all of the files on the input disk (DM0:) onto it.

Example 3 shows another disk-to-disk backup operation. This time, the maximum number of files and initial file header allocation for the output disk are changed. This information is contained in the index file, which can be placed at different locations on the disk.

```
BRU>/INI/MAX:10000/HEA:5000/POS:BEG DM: DM1:(RET)
BRU - COMPLETED
```

This command initializes the output disk (DM1:) and tells BRU that the maximum number of files allowed on the disk will be 10000(10) and the initial file header allocation will be 5000(10) headers. The index file, which contains this information, will be placed at the beginning of the disk. When the output disk has been initialized, all of the files on the input disk (DM:) will be copied onto it.

Example 4 shows a multiple disk-to-disk backup operation. You must use the SAVE option with the /IMAGE qualifier when doing a multiple disk-to-disk backup operation.

```
     BRU>/INI/IMA:SAV/VER/MOU DL: DY: RET

     BRU - MOUNT DISK 1 ON DY0:. PRESS "RETURN" WHEN DONE

     BRU - STARTING DISK 1 ON DY0:

     BRU - END OF DISK 1 ON DY0:

     BRU - STARTING VERIFY PASS DISK 1 ON DY0:

     BRU - END OF DISK 1 ON DY0:

     BRU - MOUNT DISK 2 ON DY0: . PRESS "RETURN" WHEN DONE

     BRU - STARTING DISK 2 ON DY0:

     BRU - END OF DISK 2 ON DY0:

     BRU - STARTING VERIFY PASS DISK 2 ON DY0:

     BRU - END OF DISK 2 ON DY0:

     BRU - MOUNT DISK 3 ON DY0: . PRESS "RETURN" WHEN DONE

     BRU - STARTING DISK 3 ON DY0:

     BRU - END OF DISK 3 ON DY0:

     BRU - STARTING VERIFY PASS DISK 3 ON DY0:

     BRU - END OF DISK 3 ON DY0:

     BRU - COMPLETED

     BRU>CTRL/Z
```

Example 5 shows a multiple disk-to-disk restore operation. You must specify the RESTORE option on the command line with the /IMAGE qualifier.

```
     BRU>/INI/IMA:RES/VER DY: DL: RET

     BRU - MOUNT DISK 1 ON DY0: . PRESS "RETURN" WHEN DONE

     BRU - STARTING DISK 1 ON DY0:

     BRU - THIS DISK WILL NOT CONTAIN A HARDWARE BOOTABLE SYSTEM

     BRU - END OF DISK 1 ON DY0:

     BRU - MOUNT DISK 2 ON DY0: . PRESS "RETURN" WHEN DONE

     BRU - STARTING DISK 2 ON DY0:

     BRU - END OF DISK 2 ON DY0:

     BRU - MOUNT DISK 3 ON DY0: . PRESS "RETURN" WHEN DONE
```

```
BRU - STARTING DISK 3 ON DY0:

BRU - END OF DISK 3 ON DY0:

BRU - MOUNT DISK 1 ON DY0: . PRESS "RETURN" WHEN DONE

BRU - STARTING VERIFY PASS DISK 1 ON DY0:

BRU - END OF DISK 1 ON DY0:

BRU - MOUNT DISK 2 ON DY0: . PRESS "RETURN" WHEN DONE

BRU - STARTING VERIFY PASS DISK 2 ON DY0:

BRU - END OF DISK 2 ON DY0:

BRU - MOUNT DISK 3 ON DY0: . PRESS "RETURN" WHEN DONE

BRU - STARTING VERIFY PASS DISK 3 ON DY0:

BRU - END OF DISK 3 ON DY0:

BRU - COMPLETED

BRU> CTRL/Z
```

Example 6 show how to append backup sets on a multivolume disk, plus restore the multivolume disk set. If your multivolume backup disk contains a backup set that does not occupy the entire disk, you can append backup sets on the same disk using the /APPEND qualifier.

```
BRU>/APP/IMA:SAVE/BACKUP:SECOND/INI

FROM: DB: RET

TO:   DK:,DK1: RET

BRU - MOUNT DISK 1 on DK0: . PRESS "RETURN" WHEN DONE.

BRU - STARTING DISK 1 on DK0:

BRU - END OF DISK 1 on DK0:

BRU - MOUNT DISK 2 ON DK1: . PRESS "RETURN" WHEN DONE.

BRU - STARTING DISK 2 ON DK1:

BRU - END OF DISK 2 ON DK1:

BRU - COMPLETED

BRU> CTRL/Z
```

To restore the appended backup set from the multivolume disk set, you have to specify the following:

```
BRU>/IMA:RES/BACKUPSET:SECOND/INI DK:,DK1:  DB0:

BRU - MOUNT DISK 1 ON DK0: . PRESS "RETURN" WHEN DONE.

BRU - STARTING DISK 1 ON DK0:

BRU - END OF DISK 1 ON DK0:

BRU - MOUNT DISK 2 ON DK0: . PRESS "RETURN" WHEN DONE.
BRU - STARTING DISK 2 ON DK1:

BRU - END OF DISK 2 ON DK1:

BRU - COMPLETED

BRU>CTRL/Z
```

Examples 7 and 8 show how to do incremental backups with BRU. Example 7 (following) shows a backup-by-date operation (and tape verification) and Example 8 shows a backup-by-file-specification operation.

```
BRU>/REV:AFT:(14-FEB-83 17:00)/VER RET

FROM: DM: RET

TO:   MT: RET
```

This command line backs up all files on the disk that were revised after 5:00 P.M. on February 24, 1981. After all the files have been copied onto the tape, BRU verifies the tape. If files on the tape do not verify, BRU returns an error message.

Example 8 shows a backup-by-file-specification operation:

```
BRU>DB:[7,10],[301,304]*.MAC,*.CMD RET
TO:   MM: RET
```

In this case, BRU backs up all the files in UFD [7,10] and all the .MAC and .CMD files in UFD [301,304] on the input disk to a magnetic tape.

Example 9 shows how to back up files from a mounted disk and then two ways to restore files to a mounted disk.

```
BRU>/MOU DB:[304,303],[7,326] MM: RET
```

This command line informs BRU that the input disk is mounted as a Files-11 device.

```
BRU>/NOI MM:[304,303] DB: RET
```

This command line restores the files in UFD [304,303] on the magnetic tape to the mounted disk volume without first initializing it. In this case, any file on the tape that is identical to a file already on the disk is not superseded (the input file is not copied). Not superseding files is the default operation for BRU.

```
BRU>/NOI/NEW MM:[7,326] DB: RET
```

This command line restores the files in UFD [7,326] on the magnetic tape to the mounted disk volume without first initializing it. The /NEW (/NEW_VERSION) qualifier tells BRU to create a new version of any duplicate files.

Example 10 shows how to append files from a disk to a tape with a backup set already on it and then how to restore the set back to a mounted disk.

```
BRU>/APP/VER/BAC:TODAY RET

FROM: DB:[7,326] RET

TO:    MM: RET
```

This command line appends the files in UFD [7,326] on the input disk to a magnetic tape. The name of the backup set being written on the tape is "TODAY". After the backup operation is completed, BRU verifies the tape.

```
BRU>/REW/BAC:TODAY/NOI MM:  DB: RET
```

This command line rewinds the magnetic tape containing the backup set "TODAY". All of the files in TODAY are then copied back onto a mounted output disk. If a file already exists on the disk, BRU defaults to /NOSUPERSEDE to resolve the conflict.

Example 11 shows how to exclude certain files from being copied during a backup operation.

```
BRU>/MOU/EXC DM:[1,6] MM1: RET
BRU - STARTING TAPE 1 ON MM1:

BRU - END OF TAPE 1 ON MM1:

BRU - COMPLETED

BRU> CTRL/Z
```

This command line backs up all of the files on the input disk, except for those in UFD [1,6], onto the magnetic tape.

Example 12 shows how to enter bad blocks manually and how to display them.

```
BRU>/REW/INI/BAD:MAN MM1: DM: RET
BRU - STARTING TAPE 1 ON MM1:

BRU>LBN(S)=/ RET
  053768:022
BRU>LBN(S)=10500:2
BRU>LBN(S)=12000
BRU>LBN(S)=/ RET
  053768:022
  010500:002
  012000:001
BRU>LBN(S)=// RET
BRU -- *WARNING* -- THIS DISK WILL NOT CONTAIN A HARDWARE BOOTABLE SYSTEM

BRU - END OF TAPE 1 ON MM1:

BRU - COMPLETED

BRU> CTRL/Z
```

This command line first initializes the output disk and then requests you to enter the locations of any bad blocks before copying the files from the magnetic tape. The first slash (/)

displays the bad blocks in the bad sector file on the last track of the disk. The second slash displays those blocks and the ones that have been entered manually. Two slashes (//) returns you to the BRU command level.

In this example, two locations of bad blocks for the disk were entered: there are 2 bad blocks starting at LBN 10500 and 1 block at LBN 12000. You can enter the LBN in either decimal (the default) or octal (precede the number with #), but the number of bad blocks must be in decimal. When you do not specify the number of bad blocks, it defaults to 1.

Examples 13 and 14 show BRU continuation command lines on RSX-11M and RSX-11M-PLUS (see Section 7.2.2.2 for more information).

RSX-11M and RSX-11M-PLUS

```
>RUN BRU
BRU>/REWIND-
BRU>/INVOLUME:BACKUP-
BRU>/BACKUP_SET:25MAY81-
BRU>/TAPE_LABEL:BRU123
FROM:   DM0:
TO:     MM0:
```

RSX-11M and RSX-11M-PLUS support continuation lines when you invoke BRU and then respond to the BRU> prompt. The command line can be 256(10) characters long.

RSX-11M-PLUS only (for command lines no longer than 80 characters)

```
>BRU/REWIND-
->/INVOLUME:BACKUP-
->/BACKUP_SET:25MAY81-
->/TAPE_LABEL:BRU123 DM0: MM0:
```

RSX-11M-PLUS also supports continuation lines when you use BRU from the MCR command level. In this case, the command line can only be 80 characters long (including the initial line).

7.10 MESSAGES

This section lists BRU information and error messages, describes the meaning of the message, and suggests actions to correct the errors. A WARNING message indicates an error that may or may not terminate the BRU operation. A FATAL message indicates an error that always terminates the operation.

BRU -- *WARNING* -- ALLOCATION FAILURE [ufd]filename.type;version

Explanation: During a copy to a mounted volume, there was not enough free space to copy the specified file.

User Action: Create enough free space on the volume by using the Peripheral Interchange Program (PIP; see Chapter 3) to delete or truncate some files and then reenter the command line.

BRU -- *FATAL* -- ALLOCATION FOR SYSTEM 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: 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 OPTION

Explanation: An option specified with a qualifier is not unique. For example, the "B" in /POSITION:B could mean either BEGINNING or BLOCK.

User Action: Use a form of the option that is unique. All BRU options are unique to two characters.

BRU -- *FATAL* -- AMBIGUOUS QUALIFIER

Explanation: A qualifier is not unique. For example, /RE could mean either /REVISED or /REWIND.

User Action: Use a form of the qualifier that is unique. All BRU qualifiers are unique to three characters.

BRU -- *WARNING* -- APPENDING AT DEFAULT BPI ON ddnn:
or
BRU -- *WARNING* -- APPENDING AT 1600 BPI ON ddnn:

Explanation: The wrong tape density was specified with the /APPEND qualifier. BRU performs an append operation only at the density at which the tape was previously written. The default bpi in the first message is either 800 or 6250, depending on the type of tape drive.

User Action: None. BRU continues at the correct density.

BRU -- *FATAL* -- ATTACH FAILED ON ddnn:

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 -- *FATAL* -- BACKUP DISK READ ERROR

Explanation: An unrecoverable read error occurred on the output backup disk, possibly caused by an undetected bad block, or an error occurred while BRU was sizing the input or output disk for a multivolume backup operation.

User Action: Use the BAD utility to locate all bad blocks. Then use BRU with the /BAD:AUTOMATIC qualifier to use the existing bad block information on the disk to create the file BADBLK.SYS.

BRU -- *FATAL* -- BACKUP DISK WRITE ERROR

Explanation: An unrecoverable write error occurred on the output backup disk during a multivolume backup operation. The error could have been caused by an undetected bad block.

User Action: Use the BAD utility to locate all bad blocks (see Chapter 6). Then use BRU with the /BAD:AUTOMATIC qualifier to use the existing bad block information on the disk to create the file BADBLK.SYS.

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 FULL

Explanation: The manual addition of bad blocks has resulted in more than 204(10) sets of contiguous bad blocks.

User Action: None. You cannot enter more bad blocks than the file will hold. You may not want to use the disk anymore.

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 output disk will not be hardware-bootable.

User Action: None. BRU continues 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 continues the operation.

BRU -- *WARNING* -- BOOT BLOCK READ ERROR

Explanation: An error occurred while BRU was reading the boot block.

User Action: None. BRU continues the operation.

BRU -- *WARNING* -- BOOT BLOCK VERIFY ERROR ON ddnn:

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 continues the operation.

BRU-- *FATAL* -- CANNOT APPEND ON A MOUNTED DISK

Explanation: A multivolume backup operation to disk is not possible because you cannot append to a mounted disk. You can only use the /APPEND qualifier when doing a multivolume disk backup operation to an unmounted disk.

User Action: You must use another disk to proceed with your backup operations.

BRU-- *FATAL* -- CANNOT MIX DIFFERENT TYPES OF DISKS

Explanation: The input disk and the output disk are of different types for a multivolume disk operation.

User Action: You must specify the same type of disk for the input and/or output disk when you do a multivolume restore or backup operation.

BRU-- *WARNING* -- CANNOT RESTORE CONTIGUOUSLY [ufd]filename.type;version

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 Peripheral Interchange Program (PIP; see Chapter 3) to make the file contiguous again. Use the PIP switches /DE and /TR to reclaim disk space by deletion or truncation.

BRU -- *WARNING* -- CLOSE OR WRITE ATTRIBUTES ERROR [ufd]filename.type;version
I/O ERROR CODE number

Explanation: During a copy to a mounted volume, BRU encountered an error while attempting to close the specified file.

User Action: If possible, determine the cause of the error from the I/O code. (Refer to the IAS/RSX-11 I/O Operations Reference Manual.) If it is correctable, delete the portion of the file that BRU has copied, and reenter the command line.

BRU -- COMPLETED

Explanation: The BRU operation is complete.

User Action: Enter another BRU command line or exit with a CTRL/Z.

BRU -- *FATAL* -- CONFLICTING QUALIFIERS

Explanation: Two or more of the specified qualifiers are mutually exclusive: for example, /SUPERSEDE and /NOSUPERSEDE.

User Action: Reenter the command line.

BRU -- *WARNING* -- DATA ID RECORD VERIFY ERROR

Explanation: An error occurred while BRU was verifying an 80-byte data-ID record.

User Action: None. BRU continues 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 continues 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 are 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 continues 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 in the system.

User Action: Reenter the command line, specifying the correct device specification.

BRU -- *FATAL* -- DEVICE NOT SUPPORTED

Explanation: The specified device was not a tape or a disk, or it was a disk that is not supported by BRU.

User Action: BRU supports only certain disk and magnetic tape devices. See Table 7-2 for a list of supported devices. Reenter the command line, specifying supported devices.

BRU -- *FATAL* -- DIRECTIVE ERROR

Explanation: An internal error occurred in BRU.

User Action: Reenter the command line. If the error persists, submit a Software Performance Report (SPR) that includes a hard copy of the BRU operations and error messages.

BRU -- *WARNING* -- DIRECTORY VERIFY ERROR

Explanation: A directory record on the input device did not match a directory record on the output device.

User Action: None. BRU continues 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: Reenter the command line, using a different output volume.

BRU --* -- DISK LABEL ERROR

Explanation: An I/O error occurred while BRU was reading or writing a disk label. A write error is fatal; a read error is not fatal as long as BRU can continue reading the disk. See the _IAS/RSX-11 I/O Operations Reference Manual_ for the definition of the I/O error code number.

User Action: If a write error occurred, reenter the command line, specifying a different disk.

BRU -- *WARNING* -- DISK OUT OF SEQUENCE. PLEASE MOUNT CORRECT DISK.

Explanation: The wrong disk volume was mounted on the disk drive during a restore-from-disk operation from a multivolume backup set.

User Action: Mount the correct disk on the drive.

BRU -- *FATAL* -- DISK READ ERROR

Explanation: An unrecoverable read error occurred on the output disk, possibly caused by an undetected bad block, or an error occurred while BRU was sizing the input or output disk.

User Action: Use the BAD utility to locate all bad blocks. Then use BRU with the /BAD:AUTOMATIC qualifier to use the existing bad block information on the disk to create the file BADBLK.SYS.

BRU -- *FATAL* -- DISK WRITE ERROR

Explanation: An unrecoverable write error occurred on the output disk. The error could have been caused by an undetected bad block.

User Action: Use the BAD utility to locate all bad blocks (see Chapter 6). Then use BRU with the /BAD:AUTOMATIC qualifier to use the existing bad block information on the disk to create the file BADBLK.SYS.

BRU -- *FATAL* -- DOUBLY DEFINED QUALIFIER

Explanation: The same qualifier was specified more than once on the command line.

User Action: Reenter the command line, specifying the qualifier once.

BRU -- *WARNING* -- DUPLICATE BLOCKS FOUND

Explanation: A manually entered bad block was already in the bad block file.

User Action: None. BRU continues the operation.

BRU -- END OF DISK Number ON ddnn:

Explanation: BRU has finished transferring data or verifying a disk.

User Action: None. This is an informational message.

BRU -- END OF TAPE number ON ddnn:

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 or disk volume.

User Action: Mount the correct tape or disk volume or reenter the command line, specifying the correct backup set name.

BRU -- *WARNING* -- EOT MARKER ERROR

Explanation: During a backup operation, an error occurred while BRU was writing or verifying the end-of-tape label on the output tape.

After a restore operation, an error occurred while BRU was positioning the tape at the end of a backup set for a subsequent operation.

User Action: On a write error, BRU rewinds the current tape and places it off-line. BRU then requests that a new tape be mounted and rewrites the data on the new tape.

On a verify error, BRU continues the operation.

On a positioning error, BRU finishes the operation. If you want to perform another BRU operation on the tape, use the /REWIND qualifier to position the tape to beginning-of-tape.

BRU -- *WARNING* -- ERROR ACCESSING FILE
I/O ERROR CODE number
FILE ID number

Explanation: An error occurred while BRU was writing data into a file, or BRU tried to do a compare read on a file that was already opened. BRU will continue with the next file.

See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: After BRU has finished, delete the file and then enter a command line, specifying the file.

BRU -- *WARNING* -- ERROR ACCESSING UFD. SKIPPING [ufd]
I/O ERROR CODE number

Explanation: During a copy to a mounted volume, an error occurred when BRU attempted to access a directory. See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: If possible, determine the cause of the error from the I/O error code. If correctable, try the copy operation again.

BRU -- *FATAL* -- ERROR LIMIT EXCEEDED

Explanation: BRU has reached the specified number of tape read errors and terminated execution.

User Action: Reenter the command line, using a different tape drive, or reenter the command line after cleaning the tape drive heads on the original drive.

BRU -- *FATAL* -- ERROR READING COMMAND FILE

Explanation: An I/O error occurred while BRU was reading the indirect command file.

User Action: Reenter the command line.

```
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 that 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 re-reading the disk.

See the _IAS/RSX-11 I/O Operations Reference Manual_ for the definition of the I/O error code.

User Action: None. BRU continues the operation.

BRU -- *WARNING* -- ERROR READING UFD [ufd]

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 are not backed up.

User Action: Reenter the command line. If the error still occurs, you can find the lost files by using the VFY utility (refer to Chapter 9).

BRU -- *WARNING* -- ERROR READING UFD HEADER [ufd]

Explanation: An error occurred while BRU was reading the header of the specified UFD. Files in this UFD are not backed up.

User Action: Reenter the command line. If the error still occurs, use the VFY utility to find the lost files (see Chapter 9).

BRU -- *WARNING* -- EXTENDING INDEX FILE

Explanation: The initial number of file headers was too small. Either 256(10) or 16(10) more headers will be allocated, depending on the number of blocks on the output disk.

User Action: None. BRU continues 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: Reenter the command line, using the /BAD:OVERRIDE qualifier.

```
BRU -- *WARNING* -- FILE 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. That file is not backed up. See the _IAS/RSX-11 I/O Operations Reference Manual_ for the definition of the I/O error code number.

User Action: None. BRU continues the operation.

BRU -- *WARNING* -- FILE HEADER VERIFY ERROR [ufd]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 continues 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 continues the operation.

BRU -- *FATAL* -- FILE ID EXCEEDS MAXIMUM NUMBER OF FILES

Explanation: You specified a maximum number of files with the /MAXIMUM qualifier that was smaller than a file-ID encountered on the input volume.

User Action: Reenter the command line, specifying a larger value with the /MAXIMUM qualifier.

BRU -- *WARNING* -- FILE ID SEQUENCE NUMBER ERROR [ufd]filename.type;version

Explanation: The two possible sources of this error are:

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 Master File Directory (MFD).

Therefore, the file or UFD is not valid and is not copied.

User Action: None. BRU continues the operation.

BRU -- *WARNING* -- FILE IDS WILL NOT BE PRESERVED

Explanation: File-IDs cannot be preserved because the index file 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 continues the operation without preserving file-IDs. If your input disk had a hardware-bootable system on it, your output disk will not be hardware-bootable.

If you want the disk to be hardware-bootable, perform the BRU operation again, specifying a larger value with the /MAXIMUM qualifier.

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 is not copied.

User Action: None. BRU continues the operation.

BRU -- *WARNING* -- FILE NOT FOUND [ufd]filename.type;version

Explanation: During a backup operation, BRU cannot find the header for the specified file or directory in the index file. The file is not copied.

During the verify or compare pass of a restore operation, BRU cannot find the specified file on the output device.

User Action: None. BRU continues the operation.

BRU -- *FATAL* -- FILE NOT FOUND

Explanation: BRU could not find the specified indirect command file.

User Action: Reenter the command line, correctly specifying the indirect command file.

BRU -- *WARNING* -- FILE NOT SUPERSEDED [ufd]filename.type;version

Explanation: During a copy to a mounted volume with /NOSUPERSEDE specified (or defaulted), the specified file was not restored because it already existed on the output disk.

User Action: If you want the file to be restored, enter a command line, specifying the file and either /SUPERSEDE or /NEW_VERSION.

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 reenter the command line, specifying the correct device name.

BRU -- *WARNING* -- HEADER ID RECORD VERIFY ERROR

Explanation: The BRU-generated header-ID record on the output device is different from the one on the input device.

User Action: None. BRU continues 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.

See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

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. See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: Reenter the command line.

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 continues, 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 the BAD utility to find all the bad blocks on the disk before initializing it.

BRU -- *FATAL* -- ILLEGAL USE OF DIRECTORY QUALIFIER

Explanation: Possible sources for this error are:

1. The /DIRECTORY qualifier was specified with an output device.
2. The /DIRECTORY qualifier was specified with a device other than a tape or backup disk with the BRU container file.
3. The /INITIALIZE qualifier or any of its related qualifiers were specified with the /DIRECTORY qualifier.

User Action: Refer to Section 7.4 for a description of valid uses of the /DIRECTORY qualifier.

BRU -- *FATAL* -- INCONSISTENT INITIALIZE QUALIFIERS

Explanation: The /INITIALIZE qualifier or any of its related qualifiers were specified for the output disk, but the /NOINITIALIZE qualifier was also used.

User Action: Reenter 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. See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: Reenter the command line.

BRU -- *FATAL* -- INDEX FILE WRITE ERROR

Explanation: An I/O error occurred while BRU was writing the index file on the output disk.

User Action: Use the BAD utility (see Chapter 6) to identify the bad blocks on the output disk, then reenter the command line.

BRU -- *FATAL* -- INDEXF.SYS IS FULL

Explanation: The index file cannot map any more file headers.

User Action: Reenter the command line, specifying a larger value with the /MAXIMUM qualifier.

BRU -- *FATAL* -- INITIALIZE QUALIFIERS INVALID WHEN OUTPUT IS TAPE

Explanation: The /INITIALIZE qualifier and the other qualifiers that you can specify with it may be used only when the output device is a disk.

User Action: Reenter the command line.

BRU -- *FATAL* -- INPUT DEVICE EQUALS OUTPUT DEVICE

Explanation: The input and output devices must be different.

User Action: Reenter 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(10) characters.

User Action: Truncate qualifiers and options to shorten the line. Make sure the truncated forms are unique. All BRU qualifiers are unique to three characters; all options are unique to two characters.

BRU -- *WARNING* -- INPUT VOLUME STRUCTURE LEVEL DIFFERS FROM OUTPUT VOLUME

Explanation: You have initialized the output volume, specifying with the /MAXIMUM qualifier that the number of files allowed on the volume be greater than 25593. This causes the index file on the output volume to have more than one file header.

User Action: None.

BRU -- *FATAL* -- INTERNAL ERROR

Explanation: BRU has detected an error within itself. This should not normally occur.

User Action: Please submit a Software Performance Report (SPR) with a hard copy of the BRU operations and error messages.

BRU -- *FATAL* -- INVALID DATE OR TIME

Explanation: In the command line, a date or time was specified incorrectly or is out of range.

User Action: Specify the correct date or time.

BRU -- *WARNING* -- INVALID DATE OR TIME [ufd]filename.type;version

Explanation: An invalid date or time was encountered in a file header during an incremental backup.

User Action: None. BRU continues the operation. The file is copied.

BRU -- *FATAL* -- INVALID DENSITY OR TAPE FORMAT

Explanation: You specified a density that was neither the default bpi (800 or 6250) nor 1600 bpi or you attempted to use both 7-track and 9-track tapes in a multivolume tape set.

User Action: In the former case, reenter the command line, specifying the correct density. In the latter case, only use all 7-track or all 9-track tapes for a multivolume tape set.

BRU -- *FATAL* -- INVALID DISK FORMAT

Explanation: The disk that was mounted for an /IMAGE restore operation is not a BRU multivolume backup disk.

User Action: Mount the correct disk.

BRU -- *FATAL* -- INVALID FILENAME

Explanation: The name of the indirect command file is not syntactically correct.

User Action: Reenter 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 is not restored.

BRU -- *FATAL* -- INVALID VALUE OR NAME

Explanation: A value or name specified for a qualifier has illegal syntax or is out of range.

User Action: Refer to Section 7.4 to determine the legal values for the particular qualifier.

BRU -- *FATAL* -- MANUFACTURER BAD SECTOR FILE IS CORRUPT

Explanation: BRU was unable to read the bad block information from a last-track output disk.

User Action: Reenter the command line, specifying 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: Reenter the command line. If the header still cannot be read, the files on the disk are lost and may be recovered using the VFY utility (see Chapter 9).

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: Reenter the command line. If the block cannot be read, use the VFY utility to recover the lost files. (Refer to Chapter 9 for information on VFY.)

BRU -- *FATAL* -- MISSING COLON

Explanation: A qualifier option that accepts a value was not followed by a colon.

User Action: Reenter 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: Reenter the command line.

BRU -- MOUNT DISK n ON ddnn:. PRESS "RETURN" WHEN DONE

Explanation: This message is issued each time BRU requests a disk for an image backup or restore operation.

User Action: Mount the disk specified on the drive specified and then press "RETURN".

BRU -- MOUNT TAPE n ON ddnn:

Explanation: There is no tape on the specified drive or the tape is not at load point. This message prints every two minutes until the tape is mounted.

User Action: Mount the tape specified on the drive specified.

BRU -- MOUNT ANOTHER DISK

Explanation: BRU is requesting that a new disk be mounted after encountering a fatal disk write error.

User Action: Mount a new disk on the drive.

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 EXCEEDS MAXIMUM ALLOWED LENGTH

Explanation: A name, such as a backup set name, is longer than 12(10) characters. An exception to this rule is during an /IMAGE backup operation to a mounted disk. You may specify more than 12(10) characters if you are adding the name of the backup_set filename to the command line. For additional information, see Section 7.4 for a description of the /BACKUP_SET qualifier.

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 continues the operation. Refer to Section 7.6 for information on bad block processing by BRU.

BRU -- *WARNING* -- NO FILES FOUND

Explanation: During a backup or restore operation, BRU did not find any files to transfer.

User Action: None.

BRU -- *WARNING* -- NONFATAL QUALIFIER CONFLICTS BEING IGNORED

Explanation: You entered a qualifier that conflicts with the rest of the command line, but is not fatal if ignored. For example, you used the /REWIND qualifier on a disk-to-disk operation.

User Action: None. BRU continues the operation.

BRU -- *WARNING* -- NO SUCH UFD EXISTS. SKIPPING [ufd]

Explanation: During a copy to a mounted volume, BRU encountered one or more files in the specified UFD on the input volume, but there is no corresponding UFD on the output volume.

User Action: Reenter the command line, specifying the /UFD qualifier to create the UFD.

BRU -- *FATAL* -- NUMBER OF HEADERS INCONSISTENT WITH MAXIMUM FILES

Explanation: During an attempt to initialize an output volume, BRU found that the maximum number of files specified with the /MAXIMUM qualifier was inconsistent with the number of headers initially allocated to the index file with the /HEADERS qualifier.

User Action: See the RSX-11M/M-PLUS MCR Operations Manual (the INI command) for the legal ranges of values for the /MAXIMUM and /HEADERS qualifiers.

BRU -- *WARNING* -- OPEN ERROR
I/O ERROR CODE number
FILE I/D number or [ufd]filename.type;version

Explanation: During a copy operation to a mounted volume, an error occurred while BRU was attempting to open the specified file. See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: Determine the cause of the error from the I/O error code. If correctable, delete any portion of the file already copied by BRU, then reenter the command line.

BRU -- *FATAL* -- OUTPUT DISK TOO FRAGMENTED TO RESTORE

Explanation: The internal tables in BRU have overflowed due to the 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 DEVICE 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, reenter the command line, specifying only the files you want. If you were doing an append to a tape that is already full, reenter the command line, specifying a new tape.

BRU -- *FATAL* -- OVERRIDE INVALID WITH NON LAST TRACK DEVICE

Explanation: The OVERRIDE option may be used only when the output disk is a last-track device.

User Action: Refer to Section 7.4.

BRU -- PLEASE ANSWER YES OR NO

Explanation: BRU requires a YES or NO response.

User Action: Enter YES or NO at your terminal.

BRU -- *WARNING* -- PRIVILEGE VIOLATION [ufd]filename.type;version

Explanation: During a backup operation, you attempted to copy a file that you do not have read access to.

User Action: None. BRU does not copy the file.

BRU -- *FATAL* -- PRIVILEGE VIOLATION

Explanation: The mount status of one of the devices is inconsistent with the qualifiers specified in the command line.

User Action: See Table 7-1 for the correct combinations of mounted devices and qualifiers, then reenter the command line.

BRU -- *FATAL* -- RAN OUT OF SPARE FILE IDS

Explanation: The output disk required more file headers than the input disk, but no free headers were available. The lack of headers is probably due to one of the following reasons:

1. The output disk is too fragmented because of bad blocks.

2. There are no free file headers on the input disk.

User Action: If you do not need to preserve file-IDs, reenter the command line, specifying the /NOPRESERVE qualifier.

If you want to preserve file-IDs, do one of the following:

1. If the output disk is too fragmented, run BAD (see Chapter 6) on it to display the number of bad blocks. If it contains a large number of bad blocks, you may want to use a different disk.

2. Use the PIP /FR switch (see Chapter 3) to display the number of free file headers on the input disk. If there are fewer than 4 free headers, delete some of the files and then reenter the command line. If you still do not have enough file headers, specify the /NOPRESERVE qualifier in the command line.

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 continues 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: Reenter 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: Reenter the command line.

BRU -- REWIND ERROR ON ddnn:

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 a verify operation. The error is not fatal if BRU is rewinding a tape it is finished with.

User Action: If the error is fatal, reenter the command line. If the error is not fatal, no action is required.

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 in Files-11 format.

User Action: Check to see that you have the correct disk.

BRU -- STARTING TAPE n ON ddnn:

Explanation: This message tells you which tape is being copied to or from which drive.

User Action: None. This is an informational message.

BRU -- STARTING VERIFY PASS

Explanation: This message tells you that the verify pass of a disk-to-disk operation is beginning.

User Action: None. This is an informational message.

BRU -- STARTING VERIFY PASS TAPE n ON ddnn:

Explanation: This message tells you which tape is being verified during a backup or restore operation.

User Action: None. This is an informational message.

BRU -- *FATAL* -- SYNTAX ERROR

Explanation: The command line is invalid.

User Action: Reenter the command line.

BRU -- TAPE LABEL ERROR ON ddnn:
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. See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: If a write error occurred, reenter the command line, specifying a different tape.

BRU -- *WARNING* -- TAPE LABEL VERIFY ERROR

Explanation: BRU detected an error in the tape label of the input or output tape volume during a verify operation.

User Action: None. BRU continues the operation.

BRU -- *WARNING* -- TAPE NOT AT BOT. NO REWIND OR APPEND SPECIFIED

Explanation: For a backup operation to tape, BRU will not process a tape that is not at BOT unless the /APPEND qualifier was specified.

User Action: If you want to start writing at the beginning of the tape, use the /REWIND qualifier.

You can append to a tape only at the end of the last backup set on it. If the tape is already positioned there, specify /APPEND in the command line. If it is not, specify both /REWIND and /APPEND in the command line.

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 the append operation.

BRU -- *FATAL* -- TAPE POSITIONING ERROR. NO EOV ENCOUNTERED
I/O ERROR CODE number

Explanation: The tape spacing operation to find the end-of-volume for an append operation failed. See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: Reenter the command line.

BRU -- *WARNING* -- TAPE READ ERROR

Explanation: An I/O error occurred while BRU was reading a tape.

User Action: None. BRU continues the operation.

BRU -- *FATAL* -- TAPE TO TAPE NOT SUPPORTED

Explanation: BRU does 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 BRU was writing to tape. BRU rewinds the tape and then requests that another tape be mounted. See the IAS/RSX-11 I/O Operations Reference Manual for the definition of the I/O error code number.

User Action: If the error is related to the tape drive, terminate BRU and start over on another drive.

BRU -- 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 copied
3. Copying to a disk of different size or type

NOTE

This message is not issued when BRU is restoring to a mounted volume.

User Action: None.

BRU -- *FATAL* -- TOO MANY DEVICES

Explanation: For a conventional backup a disk may be specified only once as an input or output device. However, up to eight tape drives or eight disks in an image backup may constitute the input or output.

User Action: Reenter the command line, specifying only one disk or no more than eight tape drives.

BRU -- *FATAL* -- TOO MANY FILE SPECIFICATIONS

Explanation: More than 16(10) file specifications were specified on the command line.

User Action: Reenter the command line. You can 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 a Software Performance Report (SPR) with a hard copy of the BRU operations and error messages.

BRU -- *WARNING* -- UFD RECORD VERIFY ERROR

Explanation: There is a difference between input and output devices on a UFD record.

User Action: None. BRU continues the operation.

BRU -- *FATAL* -- UNKNOWN OPTION

Explanation: An option was specified that was not recognized by BRU.

User Action: Reenter the command line. See Table 7-3 for a list of legal options.

BRU -- *FATAL* -- UNKNOWN QUALIFIER

Explanation: A qualifier was specified that was not recognized by BRU.

User Action: Reenter the command line. See Table 7-3 for a list of legal command qualifiers and their options.

BRU -- *FATAL* -- UNSUPPORTED STRUCTURE LEVEL

Explanation: The file structure level on the input disk is not supported by BRU.

User Action: Ensure that you have the correct disk. (See the descriptions of the /HEADERS and /MAXIMUM qualifiers in Section 7.4 for information on structure levels.)

BRU -- *WARNING* -- VBN NOT IN FILE

Explanation: A file-ID was encountered that is larger than the maximum file-ID in the index file. The file is ignored. This error message occurs if a UFD entry was corrupted on the input disk.

User Action: None. BRU continues the operation.

BRU -- *FATAL* -- VERIFY LOST

Explanation: During the verify pass of a disk-to-tape backup operation, BRU has lost synchronization between the input and the output. This is usually caused by the tape position being lost or by backing up from a disk that is mounted and then changed during the backup operation.

User Action: Reenter the command line.

BRU -- -FATAL* -- VOLUME NOT A BACKUP DISK

Explanation: The disk mounted for an append or restore operation does not contain a backup set file generated by BRU.

User Action: Check to see that you have the correct disk and reenter the command line.

BRU -- *FATAL* -- VOLUME NOT A BACKUP TAPE

Explanation: The tape mounted for an append or restore operation was not generated by BRU, or the tape is not positioned correctly for an append operation.

User Action: Check to see that you have the correct tape, or reenter the command line, specifying the /REWIND qualifier to position the tape.

BRU -- *FATAL* -- VOLUME NOT READY

Explanation: The device is not on-line.

User Action: Put the device on-line and reenter 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 to make it write-enabled. If it is a disk, press the Write Enable switch on the disk drive.

BRU -- *FATAL* -- WRONG BACKUP SET

Explanation: During a restore operation from a multireel tape set or a multivolume disk backup set, BRU found that one of the tapes or disks does not contain the correct backup set.

User Action: Reenter the command line, specifying the correct tape.

BRU -- *FATAL* -- WRONG INPUT VOLUME LABEL

Explanation: The input volume label specified with the /INVOLUME qualifier does not match the volume label of the input device.

User Action: Reenter the command line, specifying the correct input volume label.

CHAPTER 8

DISK SAVE AND COMPRESS (DSC)

The Disk Save and Compress (DSC) utility copies a Files-11 structured disk 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, therefore, reduces the number of retrieval pointers and file headers required for the same files on the new 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 reduces 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 in compressed form on another disk. The operation can use one command (for a disk-to-disk cycle) or two commands (for a disk-to-tape and tape-to-disk cycle). You can use DSC on-line or in either of its stand-alone versions (DSCSYS.SYS or DSC64K.SYS).

After a DSC copy operation, individual files are written in available contiguous blocks and the blocks available for new files are located in a contiguous area at the end of the new volume. 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.

DSC reads and writes data to its two buffers when it performs copy or compare operations. (See Figures 8-1 and 8-2.) 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. The maximum blocking factor is 4 for both stand-alone versions.

In a disk-to-disk copy operation, DSC:

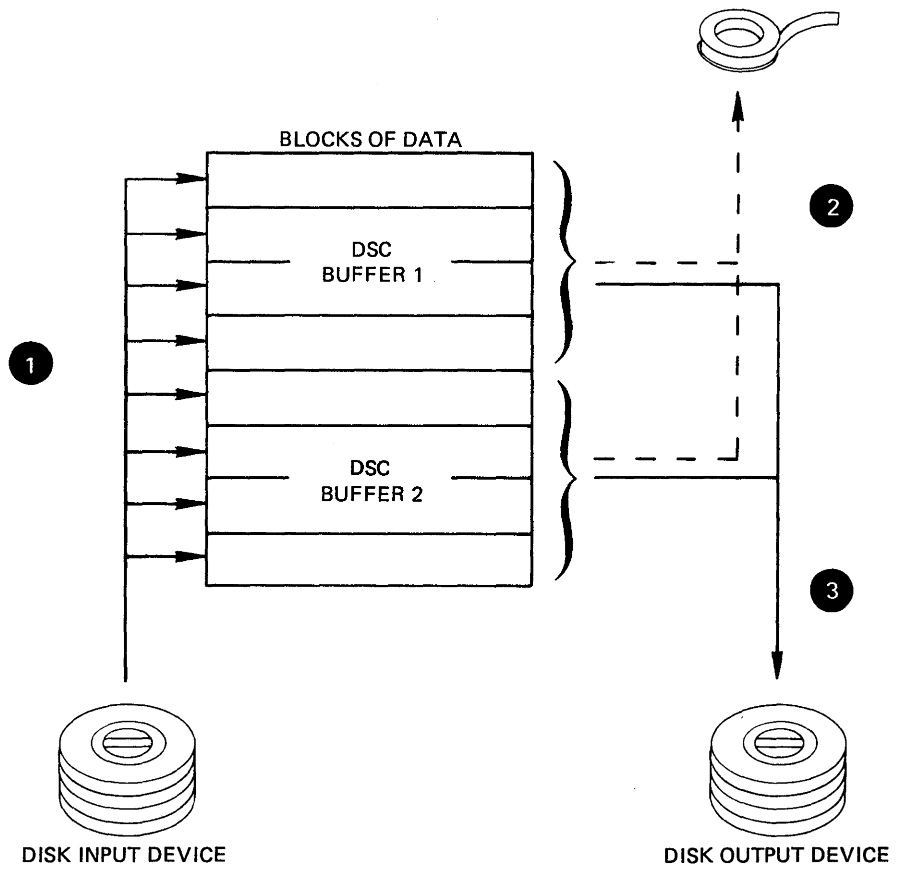
1. Copies data from disk to a DSC buffer

2. Copies data from the DSC buffer to another disk

In a disk-to-tape and tape-to-disk operation, DSC:

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.



❶ DSC reads eight (default) or more blocks of data from the disk input device to two buffers.

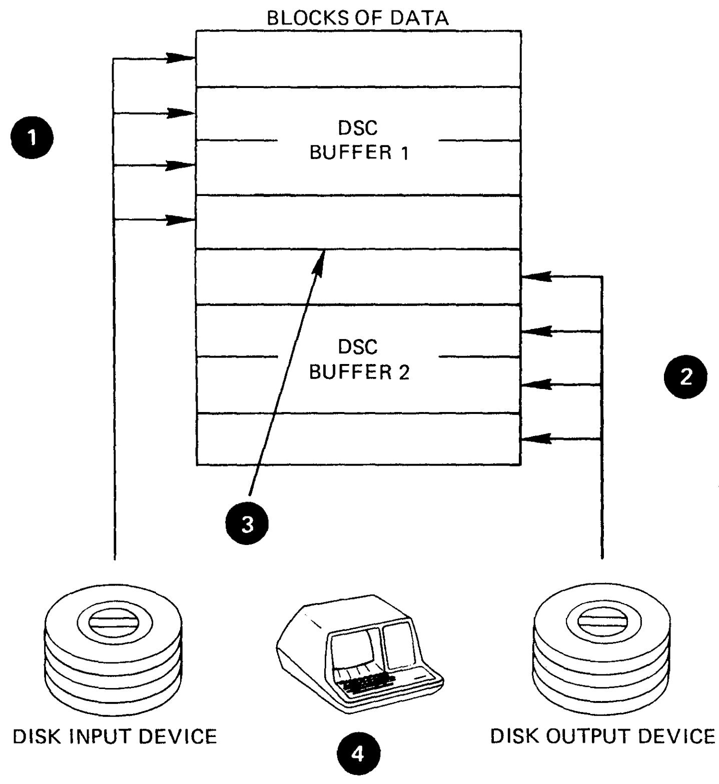
❷ In disk-to-tape copy operations, DSC writes data from the buffers to magnetic tape.

❸ In disk-to-disk copy operations, DSC writes data from the buffers to the disk output device.

DSC repeats steps ❶ and ❷ or ❸ until it copies the entire input device.

ZK-182-81

Figure 8-1 Data Transfer for DSC Copy Operation



1 DSC reads four blocks (default) of data from the disk input device to a buffer.

2 DSC reads four blocks (default) 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.

ZK-183-81

Figure 8-2 Data Transfer for DSC Compare Operation

After a disk-to-disk copy operation, you can access the data 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 DSC 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), such a saved system can function normally when it is rebooted.

You can also use DSC to recover from hardware malfunctions that have made a portion of a disk volume unreadable. If the contents of a block allocated to a data file cause a read error, you can use DSC to copy the garbled contents to the output device and to generate a warning message labeling the garbled data block. You can then access the block and correct its contents.

8.1 DSC-SUPPORTED VOLUMES

You can use DSC with a variety of mass storage or magnetic tape devices. The status DSC requires for the devices varies with the operating system.

On RSX-11M systems, DSC requires unmounted volumes.

On RSX-11M-PLUS systems, DSC requires volumes mounted within foreign characteristics.

Table 8-1 lists the devices that can be used with DSC operations.

Table 8-1
DSC-Supported Devices

Abbreviation	Type	Class
DB	RP04/RP05/RP06 disk pack	Block structured
DD[1]	TU58 cassette (DECtape II)	Block structured
DF[1]	RF11/RS11 fixed head disk	Block structured
DK	RK05/RK05F cartridge disk	Block structured
DL	RL01/RL02 cartridge disk	Block structured
DM	RK06/RK07 cartridge disk	Block structured
DP	RP02/RP03 disk pack	Block structured
DR	RM02/RM03/RM05/RM80/RP07 disk pack	Block structured
DS[1]	RH70/RS03/RS04 and RH70/RS03/ fixed-head disk	Block structured
DT[1]	TU56 DECtape	Block structured
DU[2]	RA80 Fixed media disk	Block structured
DX[1]	RX01 floppy disk	Block structured
DY[1]	RX02 floppy disk	Block Structured
MM	TE16/TU16/TU45/TU77 9-track magnetic tape	Tape
MS	TS11 magnetic tape	Tape
MT	TU10/TE10 7- or 9-track magnetic tape and TS03 9-track magnetic tape	Tape
MF[2]	TU78 magnetic tape	Tape

1. Indicates that the device cannot be used with either stand-alone DSC.

2. Indicates that the device cannot be used with DSCSYS.SYS.

8.2 INITIATING AND TERMINATING ON-LINE DSC

You can initiate the on-line DSC in any of the ways explained in Chapter 1 of this manual. To terminate on-line DSC, type CTRL/Z.

8.3 INITIATING AND TERMINATING STAND-ALONE DSC

You can bootstrap stand-alone DSC (DSCSYS.SYS or DSC64K.SYS) from disk or from tapes supplied with the operating system.

You can bootstrap stand-alone DSC in one of two ways:

1. Software boot stand-alone DSC by entering the privileged MCR BOOT command as follows:

   For Unmapped

   ```
   >INS $BOO
   >BOO[T] [1,50]DSCSYS.SYS
   ```

   For Mapped

   ```
   >INS $BOO
   >BOO[T] [1,51]DSC64K.SYS
   ```

2. Hardware boot stand-alone DSC by loading the appropriate beginning bootstrap address.

   To create a hardware-bootable stand-alone DSC tape from the distribution disk, use the Virtual Monitor Console Routine (VMR) to save the system image to tape.

When stand-alone DSC is booted, it displays the message:

```
RSX-11S V3.0 BL32 DISK SAVE AND COMPRESS UTILITY V4.0
DSC>
```

The prompt indicates that DSC is ready to accept commands. Terminate stand-alone DSC by halting the processor.

When DSC64K.SYS is booted, the Stand-alone Configuration and Disk Sizing Program (CNF) is active. Type the following for the DSC prompt:

```
>RUN DSC
```

Section 8.8 describes stand-alone DSC64K.SYS.

8.4 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 dd[nn]: where dd 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 required syntax for a device specification. If you omit the unit 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 tape drives available in round-robin fashion. Up to eight tape drives, separated by commas, can be specified as output devices in an on-line DSC operation. Stand-alone DSC permits only two tapes to be used as output devices.

DSC ignores multiple disk specifications.

file label

Identifies the output disk's volume-ID, the tape file, or tape set that 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, and you copy a disk to tape, DSC records the volume ID of the input disk on the tape. When the copy is from tape to disk, the output volume ID defaults to the ID recorded on the tape. In a disk to disk copy operation, the output volume ID will default to the ID of the input disk.

switch

One or more of the optional switches described in Section 8.5.

Input Parameters

indev:

The physical volume(s), in the format dd[nn]:, from which data is copied (see outdev:).

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-created tape set contains the contents of the disk copied by DSC.

switch

One or more of the optional switches described in Section 8.5.

8.5 DSC FILE LABELS, SWITCHES, AND OPTIONS

DSC commands can contain file labels and switches. Some switches also use options to specify values. Table 8-2 summarizes the DSC switches and options. Note that all of these switches can be used with both on-line and stand-alone DSC. See Table 8-5 for switches available only for stand-alone DSCSYS.

Table 8-2
DSC Switches and Options

Switch	Format	Description
Append	/AP	Appends a DSC file to the first volume of a tape set that already contains a DSC file. The latter file is currently the last file of the set.
Bad Block	$/\text{BAD=}\left\{\begin{array}{l}\text{MAN}\\\text{NOAUTO}\\\text{MAN:NOAUTO}\\\text{OVR}\\\text{MAN:OVR}\end{array}\right\}$	Allows manual entry of bad block locations; can supplement, override, or ignore the disk's bad block file.
Block Factor	/BL=n or /BL:n	Sets the number of 256-word blocks DSC can include in each of its two buffers.
Compare	/CMP	Compares input and output volumes for differences.
Density	/DENS=1600 /DENS=800:1600 or /DENS:1600 /DENS:800:1600 /DENS = 6250	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, TU77, TE16, or TU45 drives. The DENS=800:1600 switch is valid with TU16 or TU45 drives when they are not controlled by the TM03 formatter. The /DENS = 6250 switch is only valid with TU78 drives.
Rewind	/RW	Rewinds all magnetic tapes before DSC executes the current command.
Verify	/VE	Copies data from the input volume and compares it with data in the output volume.

8.5.1 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, when copying to tape, DSC uses 9 characters to identify the file it creates which contains the disk's 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 line to identify the file on tape that will contain 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 volume label of the output disk. If you do not specify an output file label, the default file label is that of the original input disk, (as recorded in its Home Block).

For example:

```
DSC>DB1:=MM01:,MM02:SYSFILE
```

In this command line, 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 command line 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.

8.5.2 /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.

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 line, 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 ddnn:  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 ddnn:  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 ddnn:  IS NOT THE ONLY REEL IN ITS SET
```

For example:

```
DSC>MM01:,MM:SYSFILE/RW/AP=DX1:
```

This command line appends the contents of DX1: to the last DSC-created file already present on the first output volume specified, MM01:. Since the /Rewind switch is specified (see Section 8.5.7), 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.

You can only use the /Append switch with output tape volumes. Any other use of the switch causes DSC to generate an error message and terminate the operation.

8.5.3 /Bad Block Switch

Use the /Bad Block switch (/BAD) with output disk volumes to control the way DSC uses bad block information.

The options for the /BAD switch allow you to:

1. Supplement the output disk bad block file with manually entered bad block data.

2. Ignore or override the bad block file on last track (manufacturer's list of bad blocks) or non-last-track devices.

3. Use only manually entered bad blocks.

The bad block descriptor of the disk is never altered by DSC.

If the /BAD switch is not specified, DSC will access the bad block descriptor area (the last good block on non-last-track disks or the entire last track on last-track disks) to obtain the information to create the bad block file, BADBLK.SYS. If DSC determines that the descriptor area is invalid, DSC displays a warning message (see Message 59).

The format for the /BAD switch and its options are:

```
/BAD=MAN
/BAD=NOAUTO
/BAD=MAN:NOAUTO
/BAD=OVR
/BAD=MAN:OVR
```

MAN

Supplements BADBLK.SYS with manually entered bad block data. This option may be combined with either NOAUTO or OVR to produce two more options.

NOAUTO

Ignores the bad block descriptor area on the disk. Note that in this case, DSC will attempt to write in any block it selects. This option may be combined with MANUAL to produce the option MAN:NOAUTO.

MAN:NOAUTO

Enters only manually entered bad block data in the bad block file BADBLK.SYS. Thus, DSC bypasses only manually entered bad blocks when selecting blocks to write in.

OVR

Ignores the bad block descriptor area and accesses the substitute descriptor area (the last good block on the next to the last track on the disk) to obtain the data for the creation of BADBLK.SYS. This option is valid only on last-track devices. This option may also be combined with MAN to produce the option MAN:OVR.

MAN:OVR

Allows manual entry of bad block data to the bad block file BADBLK.SYS.

When you specify MAN, MAN:NOAUTO, or MAN:OVR with the /Bad 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 after the LBN(S)= prompt as follows:

```
DSC>LBN(S)=n:m.
```

n

The logical block number (LBN), in octal, of the initial bad block in the group.

m

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.

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.

After you enter the first group of bad blocks, DSC reissues the LBN(S)= prompt. At this point, you can enter additional bad blocks.

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)=<RET>
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 previous example, the following blocks will not be used:

```
702     644     4057    5001
703     645             5002
704                     5003
705
706
707
710
```

Compare the previous 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.

If you specify /BAD=OVR on a last track device, DSC reads the last good block written by BAD on the next to the last track of the device. The information in this substitute descriptor block is used to create the bad block file. If MAN:OVR is specified, manually entered blocks will be added to the bad block file.

8.5.3.1 **Obtaining Bad Block Information** - You can obtain bad block information in two ways:

1. Running the Bad Block Locator program (BAD), described in this manual
2. Running the DIGITAL Field Service stand-alone diagnostic

The BAD utility automatically provides bad block information 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.

8.5.3.2 **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 LBNS. Use 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:

	Octal	**Decimal**
Cylinder Number =	536(8)	350.
Track Number =	16(8)	(14.)
Sector Number =	13(8)	(11.)

The LBN for the example is calculated as follows:

```
(((350.*19.)+14.)*22.)+11.=146619.
```

8.5.4 /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 with the /BL switch.

8.5.4.1 **Using the /BL Switch** - The format for the /Block Factor switch is:

```
outdev:[filelabel]/BL=n=indev:[filelabel]
```

Note: /BL:n will also be accepted.

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. (See your system manager for this information.)

The /BL switch can be specified either on the input or output side of a DSC command line.

Note, if the input volume is tape, DSC determines the block factor from the header label of the input file and ignores 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 line. However, if you specify the /BL switch only on the output side of a command line, 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 line requires more memory than DSC has available, DSC displays the message BAD BLOCKING FACTOR and exits.

For example:

```
    DSC>DB1:/BL=11=DB0:  or 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 BAD BLOCKING FACTOR.

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 to the value of the /BL switch, that value becomes the default value. DSC does not reduce its task image size if a command line is executed at a lower block factor. However, if you specify a lower block factor in a subsequent command line, DSC will create that volume at the lower factor.

8.5.4.2 **System-Dependent Requirements for /BL Switch** - On-line DSC on a mapped system expands automatically if memory is available. (See the _RSX-11M System Generation and Installation Guide_ for details of building DSC with additional memory.)

Table 8-3
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 beyond 4.

NOTE

If you create DSC tapes at a blocking factor greater than 10, they will be unusable on unmapped RSX-11M systems. Such systems can only use DSC tasks of 20K words.

8.5.5 /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. The /Compare switch is always specified on the output side of the DSC command line. If the comparison involves tape and disk, specify the tape as the input device. 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 8.5.8).

Specify the /Compare switch as follows:

```
     outdev:[filelabel]/CMP=indev:[filelabel]
```

When DSC finds a difference between the volumes it is comparing, it displays a warning message on your terminal. This warning message 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 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 begins a comparison involving tape, it first positions the specified or implied file as described in Section 8.5.7. 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 compare operation for it is completed. DSC then resumes the comparison using the next volume of the set.

8.5.6 /Density Switch

The Density switch, with its two options, allows you to override the DSC default storage density of 800 bpi for TU16, TE16, TU77, and TU45 tape drives and 6250 bpi for the TU78. The following two sections discuss these options. Although you can use other tape drives with DSC, only these drives can support the /Density switch.

You do not have to specify the /Density switch when a tape is the input device. DSC determines the density of all input tapes by first reading the tape at 800 bpi and then, if that fails, reading it at 1600 bpi. In the case of the TU78, DSC first reads the tape at 6250 bpi, then, if that fails, it reads the tape at 1600 bpi.

If you specify the /Density switch with a disk, DSC issues an error message and halts the operation.

If you specify the /Density switch with tape drives other than those above, DSC ignores the switch and does not alter the default density. Note that TS11 (TS04) drives write all tapes at 1600 bpi and cannot support 800:1600. The TS11 (TS04) ignores the /Density switch, therefore do not use it with these devices.

8.5.6.1 **1600 bpi Option** - The 1600 bpi Option directs the TU16, TE16, TU77, TU78, or TU45 drive to operate as an output volume at a density of 1600 bpi. 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 bpi density.

8.5.6.2 **Split Density Option** - The Split Density Option (/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 the first volume, at 1600 bpi. The first block on the file contains the volume label 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 bpi and the remainder of the file at 1600 bpi.

```
DSC>MM01:,MM02:SYSFILE/RW/DENS=800:1600=DB1:
```

NOTE

Magnetic tapes created using the Split Density Option do not comply with American National Standard X3.27-1978.

You cannot use the Split Density Option with the TE16 magnetic tape drive. Tape drives controlled by a TM03 also cannot use the split density option. The TM02 controller, however, does support the split density option.

8.5.7 /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 operation. You can use it to rewind either input or output volumes (see Table 8-4).

The /RW switch can be used only with magnetic tapes. If you use it with any other volume, DSC prints an error message.

INPUT

If you enter the /RW switch as part of the input specification, DSC rewinds only the first tape before the DSC operation begins. The other tapes are rewound before they are about to be accessed. 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 a 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.

OUTPUT

If you enter the /RW switch as part of the output specification, DSC rewinds the output tape before beginning a copy or compare function. The default is no rewind and the tape is not moved.

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 last DSC-created EOF it finds on that volume.

After the output tape has been rewound, DSC determines if the tape is positioned at the beginning (BOT). For a compare function, a search for the next file or a specific file begins at the current tape position. For a copy function, if the /Append switch was specified or if the tape is not positioned at BOT, the search for the current end of DSC created files begins (see Section 8.5.2); otherwise, the copy operation will overwrite any data previously stored on the tape.

Table 8-4 summarizes the use of the /Rewind switch with various DSC operations, with and without a file label.

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 BOT 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
```

Table 8-4
The /Rewind Switch and DSC Operations

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 before a copy/compare operation begins.
/RW	Input	Without	DSC copies/ compares the first file it encounters on the first volume.
/RW	Output/With	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, unless /AD is specified.
No Rewind Switch	Output		If the tape is not at BOT, DSC writes data, beginning after the last end-of-file block it encounters. (If tape is already at BOT, and the /AP switch is not specified, DSC starts there.) During copy operations to multiple tapes, DSC rewinds the tape as it is filled and takes it offline.

In this example, DSC restores a volume (DB1:) by using a tape set 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 search 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. If SYSFILE ended on MM02: the first time it was accessed, the tape is not rewound and unloaded but is positioned to access the next file.

NOTE

> When you refer to tapes after your system is booted, you must use the /Rewind switch. If you do not use the switch, the tape driver will return an error message.

8.5.8 /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:, then writes MM02: and compares MM02:, after which 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 an input tape set, DSC will copy the first file it finds on 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 volume to the beginning of the current file segment 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.

8.6 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. DSC then accesses the remaining primary file headers in numerical order. For example:

```
     DSC>DB1:=DB2:
```

In this example, DSC copies all the files on DB2: to DB1:.

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 reduces the number of file extensions required as the 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 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.

Before the actual copying of data to a disk begins, DSC must, in effect, initialize the disk. This process might take several minutes if there is a maximum number of files allocated in the Index file. Although it might appear that DSC is in a loop during this period, it is actually zeroing out all headers in the Index file.

8.7 STAND-ALONE DSC - DSCSYS.SYS

Stand-alone DSC DSCSYS.SYS does not support all the features of the on-line version. DECtapes, floppy diskettes, DF/DS fixed-head disks, and TU58 cassettes cannot be used with the standalone version. In data transfer operations, stand-alone DSC uses all of the switches described in Section 8.5.

The system data base in the stand-alone version has a Device Control Block (DCB) for each supported device type. The DCB points to a Unit Control Block (UCB) for logical unit 0 and for logical unit 1. Except for MS tapes, the UCBs for a specific device type point to a common Status Control Block (SCB) which contains the CSR and Vector Addresses associated with the related controller as listed in Table 8-6.

Since MS0: and MS1: require unique CSR and Vector addresses, their respective UCBs point to separate SCBs. The format of the system data base imposes the following restrictions:

- Logical unit numbers are limited to 0 and 1.
- Only one controller per device type (except MS tapes) is supported per command.
- DP and DR type devices have been assigned nonstandard vector addresses to avoid possible conflict with DB devices. Similarly, MT and MS tapes have been assigned nonstandard vector addresses to avoid possible conflict with MM tapes.

You can overcome some of these limitations by using the four switches listed in Table 8-5 to alter the system data base default values to match your system. A mismatch of either the CSR or Vector address will cause the stand-alone system to fail. The switches can be used only with stand-alone DSC.

Table 8-5
Stand-Alone DSCSYS.SYS Switches

Format	Switch	Description
/CSR=xxxx	/Control Status Register switch	Specifies control status addresses for a specific SCB.
/TM02=x	/TM02/TM03 Formatter switch	Specifies the physical unit number of the formatter on the RH11/RH70 controller.
/UNIT=x	/Unit switch	Specifies the physical unit that will be referenced by the indicated UCB.
/VEC=xxx	/Vector Address switch	Specifies the vector address for a specific SCB.

The four switches supplied with stand-alone DSC can appear together in a single command line to specify the appropriate values of a single device or device type. However, you can only specify 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 command lines until you either specify new switches in a new command line or terminate DSC.

The general format for a stand-alone DSC command with switches is:

```
DSC>ddnn:/switch1=x.../switchn=y
```

ddnn:

The device identifier and unit number specifying the DCB and UCB

/switch1.../switchn

One or more of the stand-alone switches described in the following sections

x,y

The values you assign to the switch(es)

8.7.1 /Control Status Register Switch

Use the Control Status Register switch (/CSR) to alter the device Control Status Register address generated by stand-alone DSC so that it conforms to the address required by your system for a particular device.

Table 8-6 lists the CSR and vector addresses of the device types supported by stand-alone DSCSYS.SYS.

Table 8-6
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[1]
DR:	176700	320[1]
MM:	172440	224
MT:	172522	320[1]
MS0:	172522	320[1]
MS1:	172526	330[1]
MF:	175400	260

1. Indicates nonstandard vector address.

The following example illustrates the correct use of the /CSR switch:

```
     DSC:MM1:/CSR=160546
     DSC>DB0:/CSR=160646
```

In this example, DSC has set the CSR addresses of the MM1: tape drive and the DB0: disk drive to 160546 and 160646, respectively. After you enter these values, you can enter another DSC 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 switch once for each device type. (The exception is the MS: tape drive; each drive must be set to its correct CSR on the host system.)

8.7.2 /TM02 Switch

Use the /TM02 switch (/TM02) to specify the physical unit number of the TM02/TM03 formatter, associated with a particular UCB on the RH controller for your system. This switch need only be used if that physical number differs from the current value. Do not confuse this number with the physical number assigned to a particular tape drive.

Stand-alone DSC is created with a physical unit number of 0 assigned to the TM02/TM03 formatter on the RH controller. This assignment affects each of the two UCB's for MM tapes. You can change this to any octal digit from 1 to 7 for each MM: device. For example:

```
     DSC>MM1:/TM02=1
```

This command line alters the physical unit number of the formatter associated with MM1: from its current assignment on the RH controller to 1. The /TM02 switch affects only the specified device. If MM0: also requires a change, the command must be repeated specifying MM0:. If MM0: and MM1: are associated with different RH controllers, they cannot appear in the same command line. The /TM02 switch only works with MM: devices. It cannot be specified with an MT:, MS:, or a disk device.

8.7.3 /Unit Switch

You can use the /Unit switch (/UNIT) 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. This constraint can be amended somewhat with the /UNIT 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:

```
     DSC>DP1:/UNIT=5
```

This command will initiate a copy from the DP currently designated as physical unit 5 to DP1:.

```
     DSC>DP1:=DP0:
```

In this command, the output device is the DP currently designated as unit 1 unless the /UNIT switch had previously been applied to DP1:.

8.7.4 /Vector Address Switch

Use the /Vector Address switch (/VEC) to change 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:

```
     DSC>DB1:/VEC=320
```

After you enter this command line, all DB:-type 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:

- MM, MT, or MS device types such as a TU16, TE10/TU10 or a TS03 drive, for example.
- Any combination of RP02/03, RP04,05/06/07, and RM02/03/05/80 (such as an RP02 disk and an RP04 disk).

For example, before you can reference MM, MT, or MS tapes, you must use the /VEC switch to change the DSC vector setting of 320 to the correct value for your system.

```
     DSC>MT1:/VEC=224
```

After you enter this command line, all MT devices will be accessed with a vector address of 224 (instead of the DSC-generated vector address of 320).

The /VEC switch applies to all drives of the same type except in the case of MS:, where only the specified device is affected.

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.

8.8 STAND-ALONE DSC - DSC64K.SYS

DSC64K is similar to the on-line version of DSC with the following exceptions:

- DSC64K is not overlaid
- DT, DX, DY, DD, DF, DS devices are not supported
- Only one tape may be referenced either as input or output
- The maximum blocking factor is 4

This version is essentially an RSX-11M system with BAD, FMT, DSC, and CNF fixed in memory, and requires 64KW of memory. When [1,51]DSC64K.SYS is software booted, the system comes up with CNF active.

CNF is the Stand-alone Configuration and Disk Sizing Program. It lists the switches you can use and then prompts you for the first device type for which you would like the CSR and vector information. It is recommended that you first specify /DEV to find out the status of devices on your system. You can also use CNF and its switches to set the CSR and vector addresses of devices in your system or to change the default formatter number (FOR=n) for some of the magnetic tape devices. (The functions of these switches correspond to the switches listed in Table 8-5 for DECSYS.SYS.)

The CNF switches are:

/CSR=nnnnnn

Changes the default CSR for the device.

/DEV

Lists the default CSR and vector addresses for all of the devices.

/FOR=n

Changes the default formatter number for some of the magnetic tape devices. The switch is only valid for MF:-and MM:- type devices. The initial default for n is 0.

/VEC=nnn

Changes the default for the device.

CNF will prompt you for the first device you want to reference. If adjustments are required for the device, use the appropriate CNF switch(es). For example, to alter the vector and CSR values for MT:, type the following:

```
MT:/VEC=nnn/CRS=nnnnnn
```

After the system data base has been adjusted with the new values, CNF prompts you for the second device. The response will be similar to the response for the first device. CNF will then request that you press the RETURN key to return control to MCR. Use the RUN command to activate any one of the four installed utilities.

The DSC64K system image (DSC64K.SYS) and symbol table (DSC64K.STB) are located in UFD [1,51] on the following disk volumes:

BIG DISK KIT	-	RSXMBL31
RK06/RK07 KIT	-	CLISRC
RL01/RL02 KIT	-	RLUTIL
RK05 KIT	-	DCLSRC

8.9 DSC DATA TRANSFERS

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.

8.9.1 Data Transfer from Disk

After you enter a DSC command line 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 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 on line and in Files-11 format.
2. Transcribes disk index files, updated for their new status.
3. Reads the data to a DSC buffer.

8.9.2 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 backup of the disk's contents. You can only use DSC-created tapes by copying them back to a disk and restoring the disk's contents to their original form. Although the tapes contain many individual files from the input disk, DSC treats the tapes as if they contained a single file -- a file of the disk's entire contents.

When DSC begins writing the disk'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 copied 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 copy operation, first ensure that the tape devices are online. You can specify multiple tape drives in the following way:

```
     DSC>ddnn(0):,ddnn(1):,...ddnn(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 line 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 line. Although you can specify up to eight drives on the output side of the command line (two drives in stand-alone DSC), you can specify each 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 previous example, the order of replacement until an end-of-file is reached would be as follows:

```
     MM0:  MM1:  MM4:  MM2:  MM0:  MM1:  MM4:  MM2:  ...
```

In summary, to transfer data to tape, DSC:

1. Verifies that the first or only volume of a tape set is on-line and write-enabled.
2. Verifies that subsequent volumes of a tape set are at Beginning of Tape (BOT), on-line when required, and write-enabled.
3. Transcribes data from a DSC buffer to the tape.

8.9.3 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. Up to eight drives may be specified on the input side; they will be referenced in round-robin fashion as described in Section 8.9.2. As it transfers the data, DSC creates and updates directory files.

Tape drives specified as input devices must be on line. 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 on line.
2. Accesses the volumes in a tape set in round-robin order.
3. Creates directory files.
4. Reads the data to a DSC buffer.

8.9.4 Data Transfer to Disk

A DSC operation is not complete until the data involved in the transfer is restored to disk.

To receive input, a disk must be on line. Any disk large enough to contain all the input data can be specified as the output disk when the data is restored to the original disk.

The disk should have an up-to-date bad block descriptor or have bad block data entered in a DSC command line 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 program (see Chapter 6).

After identifying the bad blocks on the output disk, DSC examines that disk to ensure that it has enough free blocks to contain all the data to be transferred. DSC compares the number of blocks to be transferred from the input disk(s) with the number of blocks available on the output disk. DSC issues an error message and exits if too few blocks are available.

DSC constructs the index and storage bit map files when it begins transcribing files. DSC updates the file headers to reflect the location of the data 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 LBN available on the disk. DSC will write the primary file header, 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.

Compressing files in this manner is beneficial when retrieval pointers for a noncontiguous file header are almost used up. Because DSC creates each retrieval pointer to map as many contiguous blocks as possible (maximum is 256.), the number of pointers may 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 LBN 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 on line.

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.

8.10 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 displayed on your terminal in either a long or a short form: on-line DSC displays the long form and stand-alone DSC displays the short form. You can determine the meaning of the short form message from the number provided with the message. Use the number to find the long form message in Section 8.10.1. 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
```

Error messages which only appear with the stand-alone versions of DSC are described in Section 8.10.3.

Table 8-7 is a quick reference to the single letter codes used in general messages and in I/O messages (Section 8.10.2).

Table 8-7
General Error and I/O Error Message Codes

Type of Code	Symbol	Meaning
General Error Message	Code A	Failed to read storage bit 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 data
	J	Reading input tape labels
	K	Reading file attributes
	L	Reading file header
	M	Reading index file data
	N	Reading summary data
	O	Writing file header

When DSC identifies a file in which a problem has been detected, it provides only the file-ID (file number, file sequence number) of that file. Use the Dump utility (DMP) (described in Chapter 11) with the /FI switch and /HD switch to obtain the name of the file and other information contained in the file header. For example:

```
DMP>TI:=devid:/FI:x:y/HD/BL:0
```

This command line will cause DMP to output to a terminal the header(s) of file x, y. The variables x and y, displayed in certain DSC error messages, represent the file number and file sequence number respectively.

8.10.1 DSC General Messages

The following are the general messages DSC can return.

2 CONFLICTING DEV. TYPES

Explanation: An illegal combination of device types was specified.

User Action: Check for typographical errors in device abbreviations and make sure that the disks and tape drives are not specified on the same side of the command line.

3 MIXED TAPE TYPES

Explanation: Two different types of tape drives were specified in the command line.

User Action: Reenter the command line, specifying only one type of tape drive.

4 ILLEGAL SWITCH

Explanation: The command line was entered with a switch that cannot be used with that command line.

User Action: Reenter the command line, specifying only correct switches.

5 FILE LABEL TOO LONG

Explanation: A file label consisting of more than 12(10) alphanumeric characters was specified.

User Action: Reenter the command line, specifying a shorter file label.

6 SYNTAX ERROR

Explanation: An error occurred in the command line format.

User Action: Reenter the command, specifying the right order.

7 DUP. DEV. NAME;

Explanation: The same device was entered more than once in the command line.

User Action: Reenter the command line, 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 line, specifying no more than eight devices per side.

9 DEV. ddnn: 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 line. Reenter the command line.

10 DEV. ddnn: NOT FILES-11

Explanation: The specified input device is not formatted as a Files-11 device.

User Action: Check the input device that was entered in the command line. Reenter the command line.

11 BAD BLOCK SYNTAX ERROR

Explanation: A syntax error occurred when bad block data was manually entered.

User Action: Check the command line that was entered. Reenter it.

12 BAD BLOCK COUNT TOO LARGE

Explanation: There are too many groups of bad blocks (Maximum is 125.) on the output disk for DSC to handle.

User Action: Use a different output disk.

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. Reenter the command line.

15 OUTPUT TAPE ddnn: FULL

Explanation: The specified tape is full and files cannot be appended to it.

User Action: Change the output tape. Reenter a command line to begin a new tape set.

16 OUTPUT TAPE ddnn: 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. Reenter the command line (see message 15).

17 TAPE ddnn: NOT ANSI FORMAT

OUTPUT TAPE

Explanation: If the medium is an output tape, the /Append switch was specified and the tape is not in ANSI format.

User Action: Reenter the command line and either omit the /AP switch to write the specified tape or change to an ANSI tape.

INPUT TAPE

Explanation: If the medium 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 line.

18 OUTPUT TAPE ddnn: NOT DSC TAPE

Explanation: An /Append switch was specified with a tape that was not created by DSC.

User Action: Reenter the command line and either omit the /Append switch or change to a DSC-created tape.

19 TAPE ddnn: A CONTINUATION TAPE

Explanation: If the medium is an input tape, the tape was mounted out of sequence and is not the first of a set.

User Action: Reenter the command line 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 line.

22 FILE STRUCTURE LEVEL ON ddnn: NOT SUPPORTED

Explanation: The device is not a Files-11 Structure Level disk, so it cannot be used.

User Action: Replace the device, and reenter the command line.

23 I/O ERROR A ON ddnn:
... (Additional error information)

Explanation: The I/O error indicated explains why the index file bit map on the device could not be read.

User Action: If possible, correct the cause of the error. Reenter the command line.

24 I/O ERROR B ON ddnn:
... (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: If possible, correct the cause of the error on the device. Reenter the command line.

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 line.

26 I/O ERROR C ON ddnn:
... (Additional error information)

Explanation: The I/O error indicated explains the error that occurred when DSC read the specified file.

User Action: Reenter the command line.

27 I/O ERROR D ON ddnn:
... (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: Reenter the command line, specifying a new drive.

31 I/O ERROR E ON ddnn: file ID
... (Additional error information)

Explanation: The I/O error indicated explains why the specified file header could not be read.

User Action: If possible, correct the cause of the error. Reenter the command line.

32 INPUT DEVICE ddnn: file ID, y, y NOT PRESENT

Explanation: The specified file does not have a file header in the index file. The file is not copied.

User Action: This is a warning only. If desired, the operation can be retried on a different drive.

33 INPUT DEVICE ddnn: 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 ddnn: File ID y, y UNSUPPORTED STRUCTURE LEVEL

Explanation: The header of the file indicated by the message does not contain 000401 in its File Structure Level Field (indicates ODS-1 structure). DSC only copies ODS-1 files, therefore this file will not be copied.

User Action: This message is only a warning. DSC will attempt to copy the remainder of the input.

35 INPUT DEVICE ddnn: File ID y, y FILE NUMBER CHECK

Explanation: An incorrect file header was read from the disk, causing the specified file to be lost.

User Action: Reenter the command line.

36 INPUT DEVICE ddnn: File ID, y, y FILE HEADER CHECKSUM ERROR

Explanation: Incorrect file header contents caused the specified file to be lost.

User Action: Reenter the command line.

37 INPUT DEVICE ddnn: File ID y, y SEQUENCE NUMBER CHECK

Explanation: The sequence number is incorrect.

User Action: Replace the disk and reenter the command line.

38 INPUT DEVICE ddnn: File ID y, y SEGMENT NUMBER CHECK

Explanation: The linkage connecting file segments was broken; the specified file is lost.

User Action: Reenter the command line.

39 DIRECTIVE ERROR error number

Explanation: An internal error occurred, usually the result of a system overload.

User Action: Reenter the command line.

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 line.

41 I/O ERROR I on ddnn:
File ID y, y VBN z, z
... (Additional error information)

Explanation: An I/O error occurred that resulted in bad data being read from the specified virtual block number of the indicated file on the indicated device.

User Action: This is a warning message only. You should examine the block specified to determine the extent of the error.

42 VERIFICATION ERROR ON ddnn:
File ID y, y, y VBN 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 ddnn:
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 ddnn: AND HIT RETURN

Explanation: This is an instruction only.

User Action: Mount the volume number requested on the specified tape drive and press the RETURN key when ready.

45 STARTING VERIFY PASS

Explanation: The copy operation is complete and DSC is beginning the verify operation (specified with the /Verify switch).

User Action: This is an informational message only. No User Action is required.

46 RESUME COPYING

Explanation: The verify operation (specified with the /Verify 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 ddnn: 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 press the RETURN key. Reenter the command line.

48 INPUT FILE ON ddnn: 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: Reenter the command line.

49 OUTPUT DEVICE ddnn: 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 line, specifying a less fragmented output disk.

50 OUTPUT FILE HEADER FULL ON ddnn: 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 current header(s) of the file. The allocation of blocks to the current output header is aborted. DSC will copy as many blocks as it has mapped to that header before it continues to allocate blocks to the header of the next output file. Note that some blocks will not be copied during this operation.

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.

PIP will assign a different file number to the output, other than the original input file number, therefore the files will not compare when you use the DSC /Compare switch.

51 OUTPUT FILE HEADER ON ddnn: NOT MAPPED - FILE ID y, y, y

Explanation: Space for the specified file header was not allocated. The file is lost.

User Action: Reenter the command line; a new disk may be required.

52 I/O ERROR G ON ddnn:
... (Additional error information)

Explanation: The I/O error indicated explains why the Index File Bitmap could not be written.

User Action: Reenter the command line.

53 FAILED TO READ FILE EXTENSION HEADER ON ddnn: 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: Reenter the command line.

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 then reenter the command line.

55 INDEX FILE ALLOCATION FAILURE

Explanation: Too many bad blocks exist to allow the allocation for the specified file.

User Action: Replace the disk, then reenter the command line.

56 OUTPUT DISK ddnn: IS NOT BOOTABLE

Explanation: Logical block number 0, which is the bootstrap block, of the specified disk or tape is bad. (This message will always be proceeded by message 84 and/or 86 indicating the reason for the error.)

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 program on the disk and manually enter bad block data or reenter the command line specifying another disk.

58 BAD BLOCK FILE FULL

Explanation: Too many bad blocks exist on the output disk.

User Action: Replace the disk then reenter the command line.

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 line, specifying a new disk.

60 OUTPUT DEVICE ddnn: 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 line.

61 CODE B ON ddnn:
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 ddnn:
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: Re-create the tape or reenter the command line specifying a different tape drive.

63 CODE D ON ddnn:
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: Reenter the command line.

64 FAILED TO MAP OUTPUT FILE ON ddnn:
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: Reenter the command line.

65 OUTPUT DISK ddnn: IS TOO SMALL -- nn BLOCKS NEEDED

Explanation: The output disk is not large enough to accommodate the data to be transferred.

User Action: Reenter the command line, specifying a larger output disk.

66 I/O ERROR C ON ddnn:
... (Additional error information)

Explanation: The I/O error indicated explains why the storage bit map could not be read.

User Action: Reenter the command line.

67 I/O ERROR H ON ddnn:

Explanation: The message that follows explains why the header of the storage bit map file could not be written.

User Action: Reenter the command line.

68 I/O ERROR J ON ddnn:
... (Additional error information)

Explanation: The I/O error indicated explains why the tape labels on the specified device could not be read.

User Action: Reenter the command line, specifying a different tape drive.

69 INPUT TAPE ON ddnn: 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 operation.

User Action: If the /Verify switch was not specified, check the tape and remount at BOT.

70 WRONG INPUT TAPE ON ddnn:
EXPECTING File ID, FOUND File ID

Explanation: The input tapes were specified out of sequence.

User Action: Check the tapes, then reenter them in proper order after receiving the mount instructions.

71 CODE E ON ddnn: AFTER File ID y, y, y

Explanation: This message 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: Reenter the command line.

72 I/O ERROR K ON ddnn:
AFTER File ID y, y, y
... (Additional error information)

Explanation: The I/O error indicated explains why the attributes of the specified file could not be read.

User Action: Reenter the command line.

73 I/O ERROR L ON ddnn:
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: Reenter the command line.

74 INPUT TAPE ddnn: 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, then reenter the command specifying the correct tape and file label.

76 EXPECTED EXTENSION HEADER NOT PRESENT ON ddnn: 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. Reenter the command line. If not, the input tape is bad and should be regenerated.

77 CODE F ON ddnn: 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: Reenter the command line.

78 I/O ERROR M ON ddnn:
... (Additional error information)

Explanation: The message following the device name explains why the Index File data could not be read.

User Action: Reenter the command line.

79 INDEX FILE DATA NOT PRESENT

Explanation: When reading the input tape, DSC accessed a file other than the index file. This message is the result of 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 ddnn:
... (Additional error information)

Explanation: The I/O error indicated explains why the index and storage bitmap files from the specified input tape could not be restored.

User Action: Reenter the command line, specifying 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 line.

82 I/O ERROR O ON ddnn: - File ID y, y, y
... (Additional error information)

Explanation: The I/O error indicated explains why the specified file header could not be written.

User Action: Reenter the command line.

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 reenter the command line.

84 INPUT DISK NOT BOOTABLE

Explanation: The input disk does not have a valid boot block, therefore the output disk will not be bootable. This message will always be accompanied with message 56 stating that the output disk will not be bootable.

User Action: This is a warning only. No action is required.

85 INPUT/OUTPUT DISKS DIFFER

Explanation: The boot block is usually different for each disk type, therefore the output disk may not have a valid boot block and may not be bootable. (This message will always be accompanied by other messages pertaining to the bootability of the output disk.)

User Action: This is a warning only. If message 84 is also displayed, a copy from the output disk to another disk that is the same type as the original input disk will yield a disk that is bootable.

86 BAD LBN # 0

Explanation: The output disk has a bad LBN 0 which is the boot block; therefore, the output disk will not be bootable.

User Action: This is a warning only. A copy from the output disk to another disk with a good LBN 0 will yield a disk that is bootable.

87 OUTPUT DISK ddnn: MAY NOT BE BOOTABLE

Explanation: This message is always preceeded by message 85 which indicates the reason for the error. Other messages concerning the bootability of the output disk may proceed this message. (If message 56 is displayed, the output disk will not be bootable.

User Action: Refer to message 56 and messages 84, 85, and 86 for more detail.

8.10.2 DSC I/O Messages

In on-line and both stand-alone versions of DSC, I/O errors are identified 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: Reenter the command line.

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 and the device specification in the command line, then reenter the command line.

DEVICE WRITE LOCKED

Explanation: The disk drive is write-locked.

User Action: Write-enable the disk drive and reenter the command line.

END OF FILE DETECTED

Explanation: The tape position was lost.

User Action: Reenter the command line.

END OF TAPE DETECTED

Explanation: The tape position was lost.

User Action: Reenter the command line.

END OF VOLUME DETECTED

Explanation: The tape position was lost.

User Action: Reenter the command line.

FATAL HARDWARE ERROR

Explanation: A hardware malfunction occurred.

User Action: Reenter the command line. If the error repeats, call your DIGITAL Field Service representative.

HANDLER NOT RESIDENT

Explanation: The device driver (handler) was not loaded.

User Action: Load the appropriate device driver and reenter the command line.

INSUFFICIENT POOL SPACE

Explanation: The operating system is overloaded.

User Action: Reenter the command line.

PARITY ERROR ON DEVICE

Explanation: An uncorrectable read error occurred.

User Action: Reenter the command line.

PRIVILEGE VIOLATION

Explanation: A device was mounted as Files-11 or is allocated to a different user.

User Action:

RSX-11M Users: Dismount the disk, allocate the device to yourself, and reenter the command line.

RSX-11M-PLUS Users: Dismount the disk, mount it as a foreign device and reenter the command line.

ERROR CODE IS <Driver code>

Explanation: An I/O error that DSC cannot translate occurred.

User Action: If possible, translate the error code and reenter the command line.

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 line.

8.10.3 Stand-Alone DSC Messages

The following messages appear only with the stand-alone version DSCSYS.SYS. (Similar messages are generated by DSC64K.SYS if you specify invalid values.)

ILLEGAL VECTOR ADDRESS

Explanation: An illegal vector address was specified.

User Action: Correct the vector specification and reenter the command line. 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 line.

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 line.

SPECIFIED UNIT NUMBER EXCEEDS MAX. OF 1

Explanation: Stand-alone DSC does not accept unit numbers greater than 1.

User Action: Correct the error and reenter the command line. Specify the /Unit switch if required.

CHAPTER 9

FILE STRUCTURE VERIFICATION UTILITY (VFY)

The File Structure Verification Utility (VFY) for Files-11 volumes provides the ability to:

- Check the readability and validity of a file-structured volume (default function).
- Print the number of available blocks on a file-structured volume (/FR).
- Search for files in the index file that are not in any directory; that is, files that 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.)
- Validate directories against the files they list (/DV).
- List all files in the index file, showing the file ID, file name, 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 bitmap so that it properly reflects the information in the index file (/RE).
- Restore files that are marked for deletion (/DE).
- Delete bad file headers (/HD).
- Perform a read check on every allocated block on a file-structured volume (/RC).

The volume to be verified must be mounted as a Files-11 device.

There should be no other activity on the volume while VFY is executing. In particular, activities that 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, /DE, or /HD switch is being processed. Aborting VFY while it is modifying the storage allocation or index files can seriously endanger the integrity of that volume.

9.1 VFY COMMAND FORMAT

The command line for VFY uses the format:

```
VFY>listfile,scratchdev=indev/switch
```

The parameters of this command format are:

Output Parameters

listfile

Specifies the output listing file in the following format:

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

If you do not specify a device, the default for the output listing device is the issuing terminal (TI:). The [ufd] is the UIC under which VFY is currently running. You must, however, specify the file name and file type of the output file. The default version number will be the latest version plus one.

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 you. The scratch file is automatically deleted when VFY is terminated. If you do not specify a scratch device the default device is SY0:.

If the user's default system disk is faulty or full, use this parameter to direct the scratch file to another device. The scratch file should always be assigned to a volume other than the indev volume. The scratch file is not used with the /FR and /LI switches.

Input Parameters

indev

Specifies the volume to be verified in the format dev:. If you do not specify the volume, the default is SY0:.

/switch

Specifies the function to be performed by VFY.

The VFY switches are described in detail in Section 9.4.

9.2 VFY MODE OF OPERATION

VFY normally operates in read-only mode, where 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 or /HD 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 which is the directory containing "lost" files.

If write access to the volume index or bitmap files is required for the desired operation, the user must mount the volume using the /UNLOCK switch with the MCR or DCL MOUNT command.

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.

9.3 VFY VALIDITY CHECK

VFY checks the readability and validity of the volume mounted on the specified device. This function is the default function and 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 allocated to that file in the volume bitmap.

The volume may be write-protected if it is not the system volume, or if the required scratch file is directed to another file-structured volume.

A validity check is specified in the following format:

```
listfile,scratchdev=indev<RET>

     or

indev<RET>
```

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.
```

9.4 VFY SWITCHES

VFY functions are specified with switches appended to the VFY command line. The switches and their functions are summarized in Table 9-1.

Table 9-1
VFY Switches and Functions

Switch	Format	Description
Delete	/DE	Resets the marked-for-delete indicators.
Directory Validation	/DV	Validates directories against the files they list.
Free	/FR	Prints out the available space on a volume.
Header Delete	/HD	Deletes bad file headers on a volume. The subswitch /AL allows the /HD switch to delete bad file headers without prompting the user.
Identify	/ID	Identifies the VFY version. This switch may be specified on a command line by itself at any time.
List	/LI	Lists the index file by file ID.
Lost	/LO	Scans the file structure looking for files which are not in any directory.
Read Check	/RC	Checks the volume to see if every block of every file can be read.
Rebuild	/RE	Recovers blocks that appear to be allocated but are not contained in a file.
Update	/UP	Allocates blocks that appear to be available but have been allocated to a file.

9.4.1 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 deleted.

VFY must be running under a system UIC and the volume must be mounted with the /UNLOCK switch.

9.4.2 Directory Validation Switch (/DV)

The Directory Validation Switch (/DV) examines each directory on the volume. (VFY considers any file on the volume with the file type .DIR and a fixed record length of 16 bytes to be a directory.) It then reports any errors found that could be attributed to a corrupt directory or a nonexistent file listed in the directory. For example:

```
>VFY DX:/DV
THE FOLLOWING DIRECTORY ENTRIES WERE INVALID
[301,333] FILE ID 13,2,0 DELETED.FIL;1 - FILE NOT FOUND
[301,333] FILE ID 12345,3,0 CORRUPTED.FID;1 - FILE NOT FOUND
[301,333] FILE ID 14,2,0 GARBAGE.VER;123456 - INVALID VERSION NUMBER
[301,333] FILE ID 15,1,444 RELVOLNEZ.ERO;1 - RESERVED FIELD WAS NON-ZERO

        4. INVALID DIRECTORY ENTRIES WERE FOUND
```

Directory entries may be invalid due to the following conditions:

FILE NOT FOUND

The file was either deleted without the corresponding directory entry being removed or the file ID field in the directory entry was corrupted. If the file does exist, it cannot be accessed with this directory entry.

Remove the directory entry using the PIP /RM command.

INVALID VERSION NUMBER

The directory entry was corrupted. If the file does exist, it cannot be accessed with this directory entry.

Remove version zero of the file with the PIP /RM command.

RESERVED FIELD WAS NON-ZERO

The third word of the file ID field in a directory entry is a reserved field and should always be zero. Remove the directory entry with PIP /RM and then reenter it with the PIP /EN command.

9.4.3 Free Switch (/FR)

The Free switch (/FR) displays the available space on a specified volume with the following message:

```
dev: HAS nnnn. BLOCKS FREE, nnnn. BLOCKS USED OUT OF nnnn.
```

9.4.4 Header Delete Switch (/HD)

The Header Delete switch (/HD) recognizes all bad file headers on a volume. If you specify the /AL subswitch, all bad file headers will automatically be deleted.

If you do not specify the /AL subswitch, VFY prompts you as follows:

```
>VFY DX1:/HD

CONSISTENCY CHECK OF INDEX AND BITMAP ON DX1:

FILE ID 000015,000003 007334.DIR;1 OWNER [7,334]
        BAD FILE HEADER
DELETE THIS HEADER [Y/N/Q/G]?
```

You may respond as follows:

Y RET	deletes the header and proceeds
N RET	does not delete the header and proceeds
Q RET	does not delete the header and does not proceed
G RET	deletes the header and all subsequent bad headers
RET	does not delete the header and proceeds

If you give any other response, the following message will appear:

```
VFY -- ILLEGAL RESPONSE - TRY AGAIN
```

9.4.5 List Switch (/LI)

The List switch (/LI) lists the index file. The output for each file specifies the file number, file sequence number, file name, and owner UIC, as shown in the following example:

```
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]
```

9.4.6 Lost Switch (/LO)

The Lost switch (/LO) scans the file structure looking for files that are not in any directory and cannot be referenced by file name. (VFY considers any file on the volume with the filetype .DIR and a fixed record length of 16 bytes to be a directory.) A list of the files is produced, and if the "lost file directory" [1,3] exists on that volume, the files will be entered in that directory. If an I/O error occurs, however, on a directory file operation, the files will not be entered into [1,3]. The following error message will appear:

```
FAILED TO OPEN DIRECTORY FILE
ERROR CODE -16.  - DIRECTORY [301,333]
AS A RESULT, NO FILES WILL BE ENTERED IN [1,3]
```

9.4.7 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 optional parameter [:n] is the blocking factor that 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. The buffer area available may be increased by installing VFY in a larger partition. Four blocks are available when VFY is installed in an 8K partition, and four blocks are added for each 1K increment.

For the fastest read check, the maximum block factor should be used. Whenever an error is encountered, each block of the portion in error is reread to determine which data block(s) cannot be read.

When an error is detected, a file identification line is displayed in the following format:

```
FILE ID nn,nn filename.typ;ver.  n blocks used/n blocks allocated
```

Following this line, an error message is displayed. 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 the _IAS/RSX-11 I/O Operations Reference Manual_.

9.4.8 Rebuild Switch (/RE)

The Rebuild switch (/RE) recovers lost blocks; that is, blocks that appear to be allocated but are not contained in any file.

Multiply-allocated blocks must be deleted from the file structure before the Rebuild switch can take effect.

You must run VFY under a system UIC and write-enable the volume. The scratch file should be on another volume.

9.4.9 Update Switch (/UP)

The Update switch (/UP) allocates blocks that appear to be available but are actually allocated to a file.

Files with multiply-allocated blocks must be deleted from the file structure before the Update switch can take effect.

You must run VFY under a system UIC and write-enable the volume. The scratch file should be on another volume.

9.5 FILE ERROR REPORTING

As VFY verifies a volume, error conditions are reported. Errors for a given file are preceded by a line that identifies the file in error. This line is formatted as follows:

```
FILE ID nn,mm filename.filetype;version OWNER [uic]
```

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.

[uic]

Represents the UIC for 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: VFY 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 9.5.2).

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 9.5.4).

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 9.5.1).

HEADER MAP ERROR

Explanation: VFY detected an error in the header map area that 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 errors encountered in the the map area of the file header.

If the output for VFY is directed to a terminal and you do not wish to see the error messages for a given file, enter CTRL/O. This terminates the listing of error messages for that file that is, all messages but the summary line.

9.5.1 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. Once files have been restored or deleted, run VFY with the /RE switch to assure the consistency of the volume's storage allocation bitmap.

9.5.1.1 **Restoring a File Marked-for-Delete** - To restore a file marked-for-delete, mount the disk volume using the MCR or DCL MOUNT command and the /UNLOCK switch. 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 command.

9.5.1.2 **Deleting a File Marked-for-Delete** - Files that are marked-for-delete can be deleted directly with PIP, once their unique file ID has been obtained by doing a validity check. The file ID appears as the first entry in the file identification line that precedes each list of file errors (see Section 9.5). The following example shows 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 deleted.

9.5.2 Deletion of Bad File Headers

If the volume contains bad file headers, it is advisable to delete them first by using the /HD switch before you address the problem of multiply-allocated or free blocks. Deleting bad file headers may free the blocks that were contained in the files with the bad headers. See Section 9.4.4 for a description of the /HD switch.

9.5.3 Deletion of Multiply-Allocated Blocks

If the file structure contains multiply-allocated blocks, it is necessary to delete files until there are no such blocks. An automatic rescan of the volume identifies which files share which blocks. This rescan lists the files which contain the multiply-allocated blocks. Use this information to determine which, if any, of the files can be saved and then delete the rest with the PIP delete function.

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.

9.5.4 Elimination of Free Blocks

Once there are no multiply-allocated blocks, the next concern is the elimination of blocks that are marked as free in the storage allocation bitmap but are actually allocated to a file. To reallocate these blocks in the storage allocation bitmap, run the validity check with the /UP switch. This allocates all blocks that should have been marked as allocated. See Section 9.4.9 for a description of the /UP switch.

When you have no multiply-allocated blocks and no blocks marked as free that are actually in use, the file structure is safe for writing new files and extending existing files. Files may have data blocks that have been overwritten as the result of multiple allocation.

9.5.5 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 (Section 9.3). The last two lines of output give the free space on the volume. The first line reports the amount of available space according to the index file (that is, the number of blocks that are not in use by any file in the index file). The second line reports the amount of available space according to the storage allocation bitmap.

If there are no errors, these two figures should agree. If the index file indicates that more blocks are free than the storage allocation bitmap indicate, 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 and specifying the /RE switch. See Section 9.4.8 for a description of the /RE switch.

9.6 VFY ERROR MESSAGES

The VFY error messages, their explanations, and suggested user actions are described below.

VFY -- COMMAND SYNTAX ERROR

Explanation: The command as entered does not conform to command syntax rules.

User Action: Reenter the command line with the correct syntax specified.

VFY -- CLOSE FAILURE ON BIT MAP

or

VFY -- CLOSE FAILURE ON INDEX FILE

or

VFY -- CLOSE FAILURE ON TEMPORARY FILE

or

VFY -- CLOSE FAILURE ON LISTING FILE

or

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 -- I/O ERROR WRITING FILE HEADER

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 -- FAILED TO ALLOCATE SPACE FOR TEMP FILE

Explanation: The volume specified for the temporary scratch file is full.

User Action: Use PIP to delete unnecessary files and rerun VFY, or specify another volume as the scratch device when you reenter the command line.

VFY -- FAILED TO ATTACH DEVICE

or

VFY -- FAILED TO DETACH DEVICE

Explanation: The list file specified a terminal device. VFY was not able to attach or detach the device.

User Action: Reenter the command line with a list file device that can be attached or detached.

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 conditions caused the message and correct that condition. Reenter the command line.

VFY -- FAILED TO FIND INDEXF.SYS;1 IN MFD - WILL OPEN INDEX BY FILE ID 1,1

or

VFY -- FAILED TO FIND BITMAP.SYS;1 IN MFD - WILL OPEN BITMAP BY FILE ID 2.2

Explanation: The Master File Directory has been corrupted.

User Action: Copy the disk using the BRU utility (see Chapter 7).

VFY -- FAILED TO OPEN DIRECTORY FILE (See OPEN FAILURE error messages)

VFY -- ILLEGAL DEVICE

Explanation: The input device specified is something other than a disk or DECtape.

User Action: Reenter the command line with a mounted Files-11 device specified.

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 -- 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

or

VFY -- FAILED TO OPEN FILE FOR READ CHECK

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 the specified directory.
- The volume is not mounted.
- The specified file is read protected.
- The specified file does not exist.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

VFY -- STORAGE CONTROL BLOCK (VBN1 of BITMAP.SYS) IS CORRUPTED

Explanation: The Storage Control Block is corrupt. This is harmless, because only VFY and PIP /FR can examine the block.

User Action: Copy the disk using BRU or DSC (see Chapter 7 or 8, respectively).

VFY -- THEY ARE STILL LOST, COULD NOT FIND DIRECTORY

Explanation: UFD [1,3] did not exist on the volume. UFD [1,3] is the "lost files" directory. VFY enters all files found by the /LO switch into this directory.

User Action: Use the MCR UFD command to enter UFD [1,3] on the volume.

CHAPTER 10

LIBRARIAN UTILITY PROGRAM (LBR)

The Librarian Utility Program (LBR) allows you to create, update, modify, list, and maintain library files. A library file is a direct access file that contains a collection of related files. LBR organizes files, usually having the same file type, into library modules so that you have rapid and convenient access to your files.

Library files contain two directory tables: the entry point table (EPT) and the module name table (MNT). The EPT contains entry point names that consist of global symbols defined as entry points in MACRO source programs. The MNT contains names of the modules in the library. Both tables are alphabetically ordered.

The following paragraphs describe the three types of libraries: object, macro, and universal.

Object library files (.OLB) contain object files (.OBJ). The module names are derived from .TITLE directives, while the entry point names are derived from global symbols defined in the module. LBR references the module code in the library by the module name. The source program references object library modules by the entry point name. Entry points apply only to object libraries.

You use object module libraries as input to the Task Builder. The Task Builder (TKB) searches for definitions of all global symbols referenced in a program in the following manner. First, TKB searches the other modules specified, then it searches the specified user-written object module library, and finally, it searches the system library.

Macro library files (.MLB) contain source macro files (.MAC). The module names are derived from .MACRO directives. From each macro definition, LBR extracts the name and creates an entry in the module name table. The entry in the module name table is the means by which the assembler finds the associated macro definition in the library.

You use macro library modules as input to the MACRO-11 Assembler. The assembler searches the specified library for macros listed in .MCALL statements and called in the source program before searching the system macro library.

Universal library files (.ULB) contain modules inserted from any kind of file whether it be a program or text. The module names are either user-specified in the Insert (/IN) switch or derived from the file name at the time of insertion.

Primarily, you use universal libraries to package related files together. You can reference a universal library module in a program by using the Universal Library Access ($ULA) system library routine. $ULA, specified in the macro source program, establishes the necessary conditions for access (read only) to a universal library module. (For more information on $ULA, see the IAS/RSX-11 System Library Routines Reference Manual.)

You can invoke LBR using any of the methods for invoking a utility described in Chapter 1.

10.1 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 any available space. The entry point table has zero length for macro and universal libraries. Figure 10-1 illustrates object and macro library file format and Figure 10-2 illustrates universal library file format.

10.1.1 Library Header

The header section is a full block in which the first 24(10) words are used to describe the current status of the library. The header's contents are updated as the library is modified. This allows LBR to access the necessary information to perform its functions (for example, Insert, Compress, and Delete). The twenty-fourth word in the library header is the default insert file type for universal libraries and is undefined for macro and object libraries. See Figure 10-3.

10.1.2 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 10-4.) 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 applies only to object library files.

10.1.3 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 10-5. The module name table is sequenced in order of ascending module names.

10.1.4 Module Header

Each module starts with a header of eight words for object and macro modules and 32(10) words for universal modules. The module header contains information about the module such as the type and status of the module, its length (number of words), and its attributes. See Figure 10-6 and Figure 10-7.

In object and universal modules, the low-order bit of the attributes byte is set if the module has the selective search attribute. In universal modules, bit 1 of the attributes byte is set if the input file was contiguous. Also, in object modules, the two words of type-dependent information contain the module identification defined by the .IDENT directive at assembly time. In macro modules, these two words are undefined.

For universal modules, type-dependent identification is derived from the file type and version number of the input file.

Universal libraries allow you to change the module header, which contains optional descriptive information, by means of the Modify Header switch (/MH).

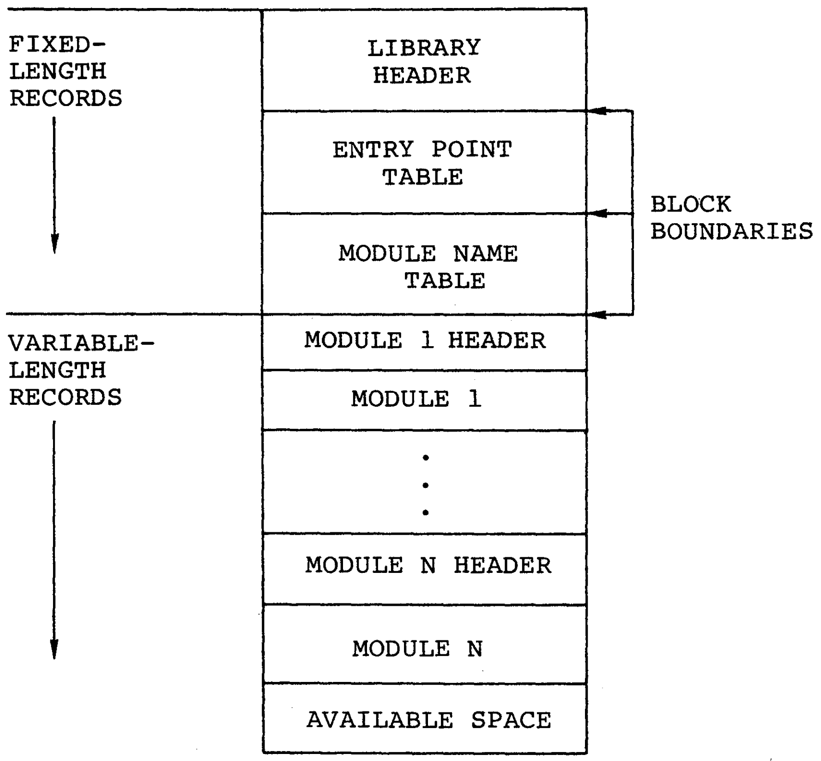
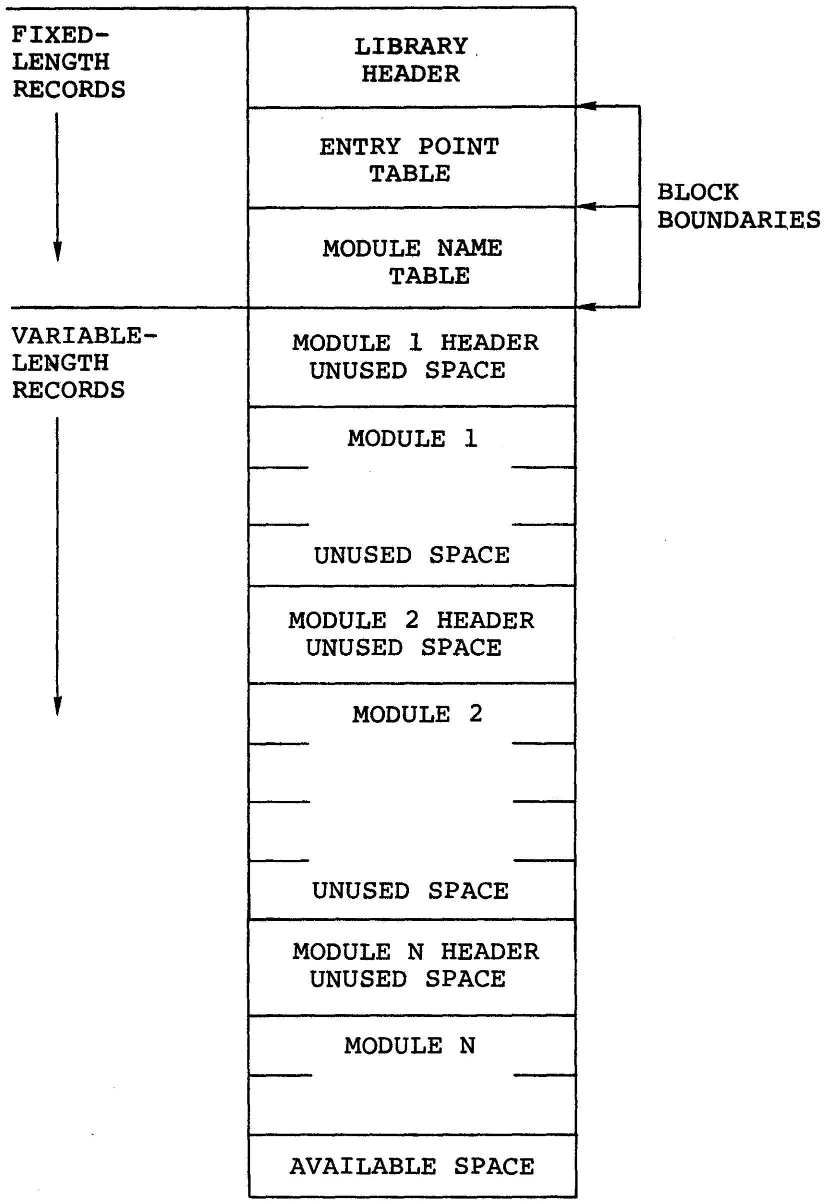


Figure 10-1 General Format for Object and Macro Library Files



NOTE

All universal module headers and the first record of each universal module will start on a block boundary.

ZK-185-81

Figure 10-2 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[1]	

[1]UNDEFINED FOR MACRO AND OBJECT LIBRARIES

ZK-186-81

Figure 10-3 Contents of Library Header

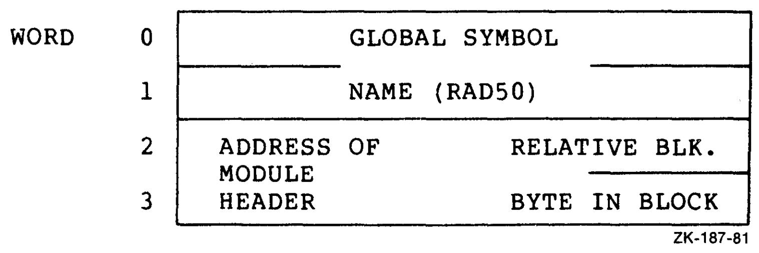


Figure 10-4 Format of Entry Point Table Element

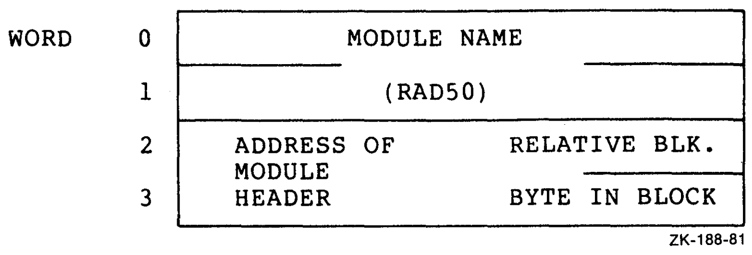


Figure 10-5 Format of Module Name Table Element

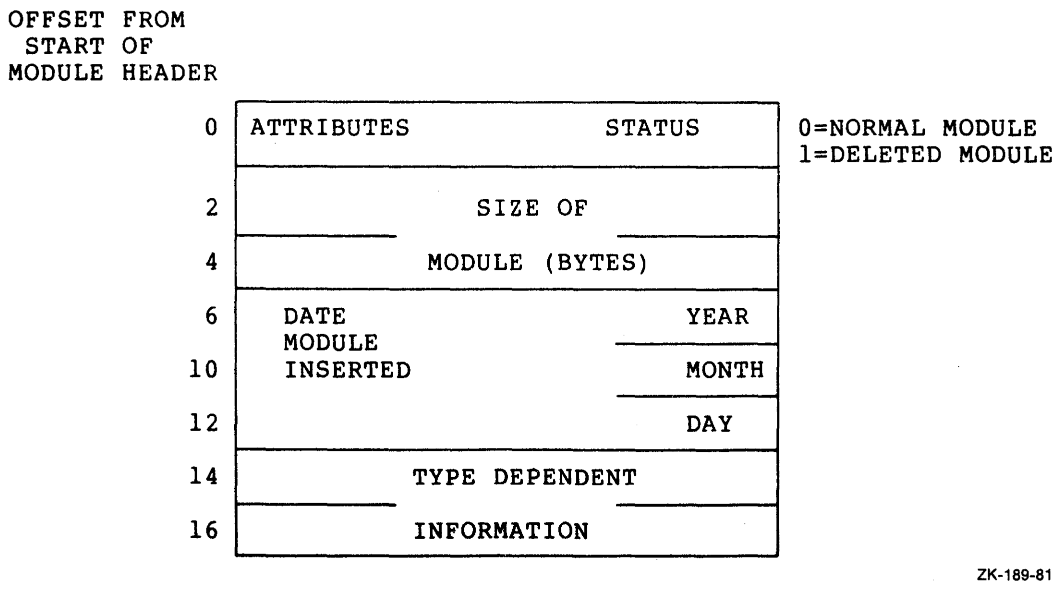


Figure 10-6 Module Header Format for Object and Macro Libraries

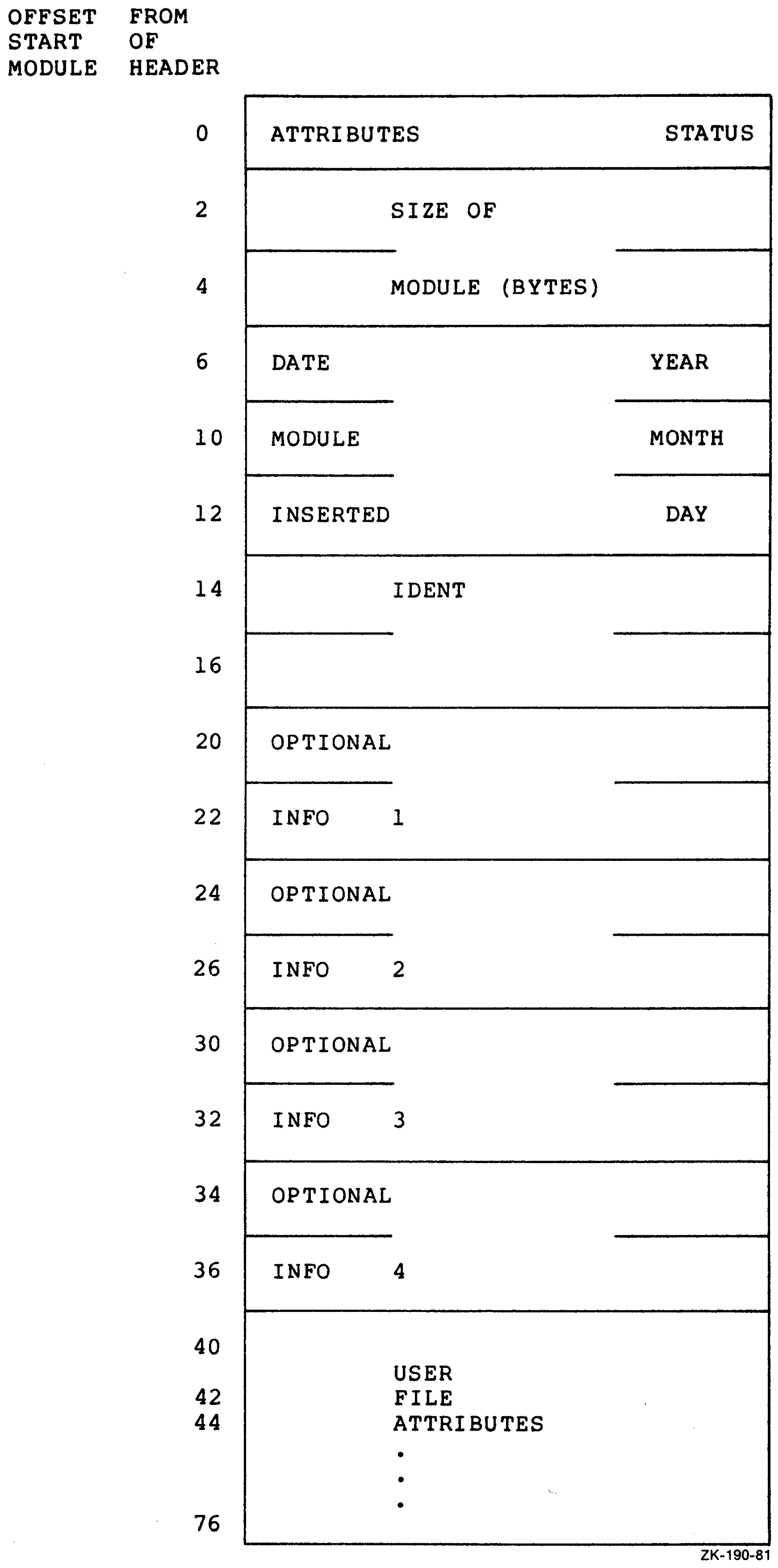


Figure 10-7 Module Header Format for Universal Libraries

10.2 LBR RESTRICTIONS

The following restrictions apply when using LBR:

- Limit of 65,536(64.K) words per module.
- Limit of 65,536(64.K) blocks per library.
- Tables should be allocated their anticipated maximum size. Expanding table allocations requires using the Compress switch (/CO) 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 10.5.8.
- The use of wildcards in file specifiers is not allowed (that is, forms such as *.OBJ, where the * indicates all modules with type .OBJ).

The library's tables must contain enough space for both the modules being replaced and their replacements because the new modules are entered and the old modules are only logically (not physically) deleted.

10.3 LBR COMMAND LINE

LBR accepts command lines in the following general format:

```
outfile[,listfile]=infile1[,infile2,...infilen]
```

LBR allows only one level of indirect command file nesting. For a complete description of file specifiers, see Chapter 1.

10.4 DEFAULTS FOR LBR FILE SPECIFIERS

Table 10-1 describes the defaults for LBR file specifiers.

Table 10-1
LBR File Specifiers Defaults

Specifier	Default
dev:	Output File SY0: Listing File The device that was specified for the output file; otherwise, the default for the output file. Input File For the first input file specifier, SY0:. For subsequent input file specifiers, the device specified in the previous input file specifier; otherwise, the default for the previous input file specifier.

(continued on next page)

Table 10-1 (Cont.)
LBR File Specifiers Defaults

Specifier	Default
dev:	Output File [ufd] Output File The UIC under which LBR is currently running. Listing File The UFD that 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 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 10.5.4), except when the Compress (/CO) or Create (/CR) switch is specified (see Sections 10.5.1 and 10.5.2, respectively). Listing File .LST Input File Refer to the descriptions of Compress (Section 10.5.1), Insert (Section 10.5.8), and Replace (Section 10.5.11) switches.
;ver	Latest version of the file, or latest version plus 1 for the output file when the Compress (/CO), Create (/CR), or Extract (/EX) switches are specified.
/switch	Output File /IN (Insert) List File /SP/LI (Spool and List module names) Input File None.

10.5 LBR SWITCHES

LBR uses switches appended to file specifications to invoke functions. These switches are summarized in Table 10-2.

Table 10-2
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.
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 the selective search attribute in the module header.
Squeeze	/SZ	Reduce the size of the macro source.
	/-SZ	Do not reduce the size of a specific macro source.

10.5.1 Compress Switch (/CO)

Use the Compress switch (/CO) to physically delete all logically deleted records, to put all free space at the end of the file, and to make 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 specification. The format for specifying the /CO switch is:

```
outfile/CO:size:ept:mnt=infile
```

outfile

Specifies the file that is to become the compressed version of the input file. The default file type is .OLB if the input file is an object library, .MLB if the 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. The 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. The maximum number of entries is 4096(10).

infile

Specifies the library file to be compressed. The 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 file type (see Section 10.5.4).

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 one version greater than the latest version for the file.

Both files, RICKLIB.OLB and SHEILA.OLB, reside in the default directory file on SY0:.

10.5.2 Create Switch (/CR)

Use the Create switch (/CR) to allocate a contiguous library file on a direct access device (for example, a disk). The switch 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 /CR 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 new library file in disk (256(10)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 zero for macro libraries and universal libraries. The 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). The 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 10.5.4), 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
```

Secondly, 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 (/IN) is the default switch for input files (see Section 10.5.8).

10.5.3 **Delete Switch (/DE)**

Use the Delete switch (/DE) to logically delete library modules and their associated entry points (global symbols) from a library file. Up to 15(10) 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 specification.

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 that 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 10.5.1).

The format for specifying the /DE 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.

10.5.4 Default Switch (/DF)

Use the Default switch (/DF) to specify the default library file type. Acceptable default 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 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. Exceptions to this occur when you use /CO or /CR. When you specify /CO, the default applies to the library input file. When you specify /CR, 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 filetype of the file to be opened. After that, 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 specification. The format for specifying the /DF 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
```

The file RICKLIB.MLB is opened for insertion.

```
LBR>/DF:MAC
LBR>RICKLIB/DF:OBJ=infile
```

OBJ replaces MAC as the default filetype. The file RICKLIB.OLB is opened for insertion.

```
LBR>/DF:MAC
LBR>RICKLIB/CR
```

The macro library RICKLIB.MLB is created.

```
LBR>/DF:MAC
LBR>RICKLIB/CR::::OBJ
```

Because OBJ is specified, it overrides the default (MAC). The object library RICKLIB.OLB is created.

```
LBR>/DF:OBJ
LBR>TEMP/CO=RICKLIB.MLB
```

Because RICKLIB.MLB is a macro library, MAC overrides the default (OBJ). The macro library file TEMP.MLB is created to receive the compressed output.

```
LBR>/DF:UNI
LBR>RICHLIB=TEST
```

The file RICHLIB.ULB is opened for insertion.

10.5.5 Delete Global Switch (/DG)

Use the Delete Global switch (/DG) to delete a specified entry point (global symbol) from the EPT. Up to 15(10) entry points may be deleted with one command. This command does not affect the object module that 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 specification.

The format for specifying the /DG 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.

10.5.6 Entry Point Switch (/EP)

Use the Entry Point switch (/EP) to control (include or exclude) the placement of global symbols in a library entry point table. The switch can be specified in either a positive or negative format:

/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 format (/EP) causes all entry points in a module or modules to be entered in the library entry point table.

Either negative format (/-EP or /NOEP) provides for a module to be included in a library, but excludes 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 /EP switch has no effect on macro or universal libraries.

The format for specifying the /EP 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 /EP 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.

NOTE

Although not reflected in the command formats, the positive and negative forms of the switch may be applied to both the output and input file specifications. For example, the effect of /EP applied to the output file can be overridden by applying /-EP to a specific input file.

The /-EP switch is useful for including modules that contain duplicate entry point names in the same library. The /-EP switch provides the means for entering a module in the library without having its entry points included in the library entry point table.

The /-EP switch 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. The /-EP switch can be used to exclude entry points from being entered in the library entry point table.

Depending on whether the /EP switch is applied to the output specification or to an input specification, it has either a global or local effect.

When applied to the output file specification, the /EP 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 specification, 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.

Entry points in an object module are not affected by the /EP switch. The switch only affects entries in the library entry point table.

10.5.7 Extract Switch (/EX)

Use the Extract switch /EX 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 in 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 specifications. The format for specifying the /EX switch 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 in alphabetical order 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.

10.5.8 Insert Switch (/IN) for Object and Macro Libraries

Use the Insert switch /IN to insert 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. (The Insert switch for Universal Libraries, is explained in Section 10.5.9.) The /IN switch is the default library file option and can be appended only to the library file specification.

If you attempt to insert an input module that 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 /IN 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 10.5.4). 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 the modules to be inserted into the library file. The default file type is .OBJ if the outfile is an object library and .MAC if the 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 file type for files SHEILA, LAURA, and JENNY is .OBJ if RICKLIB is an object module library, or .MAC if RICKLIB is a macro library.

10.5.9 Insert Switch (/IN) for Universal Libraries

The Insert switch (/IN) 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 applied to the input file rather than the output file. You can also specify module name and descriptive information as switch values in the command line. In addition, LBR copies input file attributes to the module header.

The high block indicator (F.HIBK of the file's descriptor block) 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 to it as are used.

The format for specifying the /IN switch for universal libraries is:

```
outfile=infile/IN:name:op:op:op:op
```

outfile

Specifies the universal library into which the infile is to be inserted.

infile

Specifies the input file to be inserted into the outfile. The default for the file type is the value indicated at the universal library's creation time. (See Section 10.5.2.)

/IN

Specifies the Insert switch.

:name

Optionally specifies the module name (up to six Radix-50 characters). The default is the first six characters of the input file name.

:op

Specifies optional descriptive information (up to six 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 /IN switch and its values. As a result, all the information normally specified by the switch values defaults as described in this example.

10.5.10 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.

These switches can be appended only to the output file specification or the list file specification.

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 line.

The format for specifying list switches is:

```
outfile[,listfile]/switch(es)
```

outfile

Specifies the library file whose contents are 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 in file SY0:RICKLIB.OLB, and a full description of each one contained 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, which lists the module names, their entry points, and a full description of each module for file RICKLIB.

10.5.11 Modify Header Switch (/MH)

The Modify Header switch pertains only to universal libraries and allows you to modify the optional user-specified information in the module header.

The format for specifying the /MH 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 six 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"
```

10.5.12 Replace Switch (/RP) For Macro and Object Libraries

Use the Replace switch /RP 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 uppercase 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 specification and all of the input files are assumed to contain replacement modules.
- Local format - The /RP switch is appended to an input file specification 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 file type depends on the current default (see Section 10.5.4). 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.

The Global format 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 specification.

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.

infile

Specifies the input file that contains replacement modules for the output library file. The local format default is the same as the global format default.

/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 (/IN) and instructs LBR to treat the module(s) contained in the specified file as replacement modules.

Examples

The files used in the following four examples, and the modules contained within each file, are depicted in Figure 10-8. 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 10-8.

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 10-9.

2. `LBR>RICKLIB=CHRIS,SHEILA/RP`

   ```
   MODULE "SHEILA" REPLACED
   ```

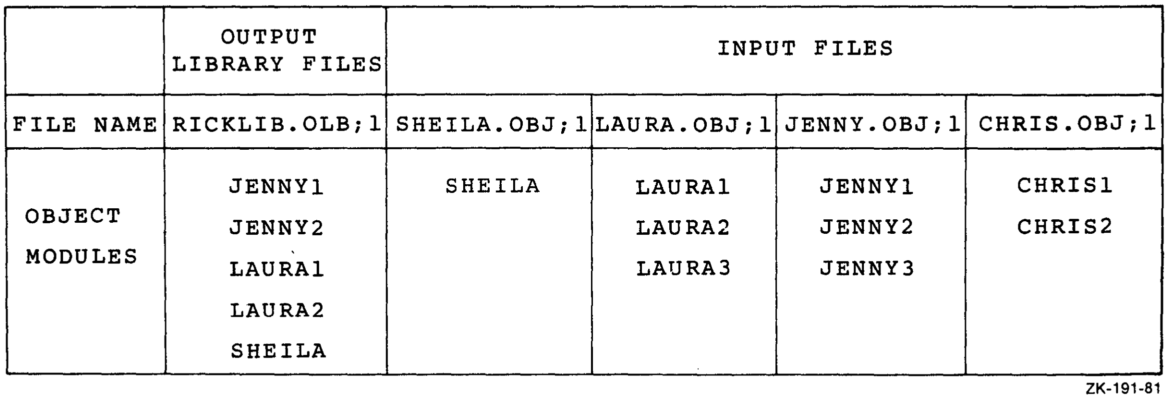
   In this example, the local format of the /RP switch is used. The object module SHEILA from file SHEILA is replaced in the library file RICKLIB. The object modules in the file CHRIS are inserted in the library file. (See the description of the /IN switch in Section 10.5.8.) The resulting library file is shown in Figure 10-10.

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 that contains modules to be inserted. The resulting library file is shown in Figure 10-11.

4.
```
LBR>RICKLIB/RP=SHEILA,LAURA/-RP,JENNY

MODULE "SHEILA" REPLACED
LBR -- *FATAL* -- DUPLICATE MODULE "LAURA1" IN LAURA.OBJ;1
```

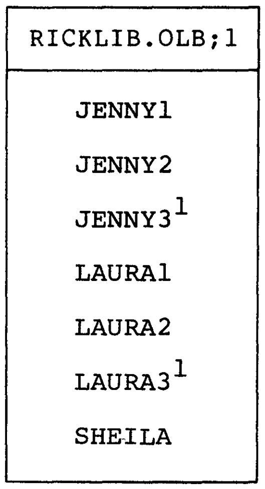
   In this example, only module SHEILA from file SHEILA was replaced. The user specified that the modules in file LAURA not be replaced (/-RP), but inserted. One of the modules contained in file LAURA duplicated an already existing module in file RICKLIB (see Figure 10-8). Therefore, LBR issued the fatal error message and terminated the processing of the current command line.



	OUTPUT LIBRARY FILES	INPUT FILES			
FILE NAME	RICKLIB.OLB;1	SHEILA.OBJ;1	LAURA.OBJ;1	JENNY.OBJ;1	CHRIS.OBJ;1
OBJECT MODULES	JENNY1 JENNY2 LAURA1 LAURA2 SHEILA	SHEILA	LAURA1 LAURA2 LAURA3	JENNY1 JENNY2 JENNY3	CHRIS1 CHRIS2

ZK-191-81

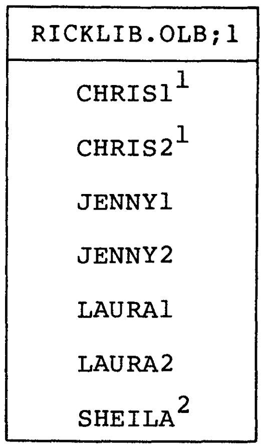
Figure 10-8 Sample Files Used in LBR Examples 1-4



1. These modules did not exist in the library file prior to the execution of this example, but they did exist in the input files. LBR, therefore, assumed that they were to be inserted. Since LBR handled these modules as a normal insert, no message was printed on the input terminal.

ZK-192-81

Figure 10-9 Output Library File After Execution of Example 1



1. These modules are inserted.

2. This module is replaced.

ZK-193-81

Figure 10-10 Output Library File After Execution of Example 2

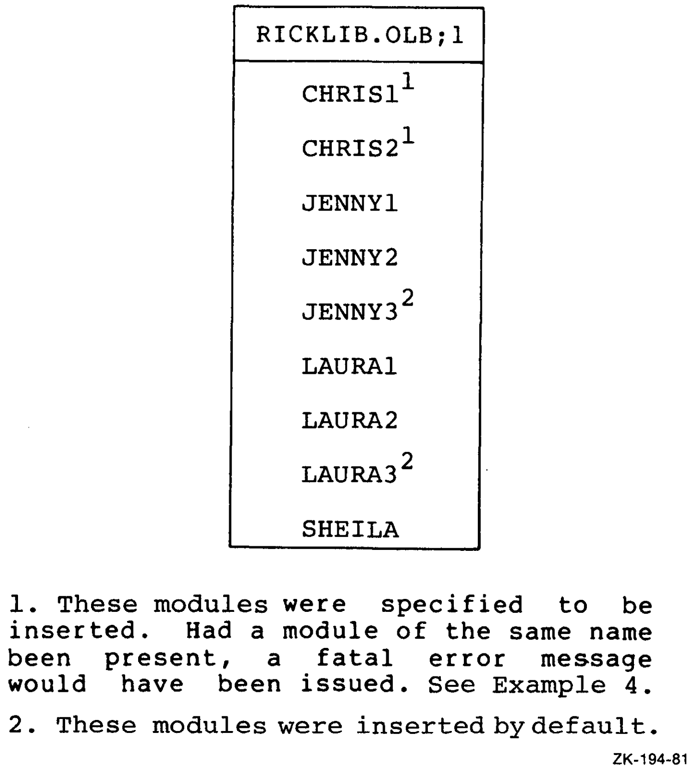

RICKLIB.OLB;1
CHRIS1[1]
CHRIS2[1]
JENNY1
JENNY2
JENNY3[2]
LAURA1
LAURA2
LAURA3[2]
SHEILA
1. These modules were specified to be inserted. Had a module of the same name been present, a fatal error message would have been issued. See Example 4.
2. These modules were inserted by default.
ZK-194-81

Figure 10-11 Output Library File After Execution of Example 3

10.5.13 Replace Switch (/RP) for Universal Libraries

Use the /RP switch for universal libraries in the same way as for macro and object libraries. However, you can also specify the same values for the /RP switch as for the /IN switch for universal libraries (see Section 10.5.9).

As with macro and object libraries, you can specify the /RP switch with either the output file specification or with the input file specifications.

The global format of the /RP switch for universal libraries is:

```
outfile/RP:name:op:op:op:op=infile[,infile2,....infilen]
```

The local format of the /RP switch 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 that contains replacement modules for the library file. The default for the file type is the value indicated at the universal library's creation time. (See Section 10.5.2).

/RP

Specifies the Replace switch.

:name

Optionally specifies the module name to be replaced (up to six Radix-50 characters). The default is the first six characters of the infile name.

:op

Specifies optional descriptive information (up to six 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 module DEBBIE in the universal library TEXT.ULB with an updated module from file DEBBIE.TXT. The date of replacement is specified by the optional user 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 10-12. The resulting library file is shown in Figure 10-13.

	OUTPUT LIBRARY FILE	INPUT FILES
FILE NAME	TEXT.ULB;1	DEBBIE.TXT
MODULES	DEBBIE BERNIE	

ZK-195-81

Figure 10-12 Sample Files for Universal Library Replace Example

TEXT.ULB;1
DEBBIE[1] BERNIE

1.The module DEBBIE was replaced. If a different infile were specified, that file would become module DEBBIE and occupy the same location in TEXT.ULB.

ZK-196-81

Figure 10-13 Output Library File After Execution of Universal Library Replace Example

10.5.14 Spool Switch (/SP)

The Spool switch (/SP) 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 (see the RSX-11M/M-PLUS Batch and Queue Operations Manual for a description of the spooler task).

The automatic printing of the listing file can be inhibited by specifying /-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 /SP 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. Because the /-SP switch is specified, the file is not automatically printed.

10.5.15 Selective Search Switch (/SS)

Use the Selective Search switch (/SS) 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 with input files for insertion or replacement operations only, 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 /SS 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.

Example

```
LBR>ANGEL=JOHN,JILL/SS,MARK/SS,MARY
```

The object files JOHN.OBJ, JILL.OBJ, MARK.OBJ, and MARY.OBJ are inserted into object library ANGEL.OLB. The selective search attribute bit is set in both the JILL and MARK object module header.

10.5.16 Squeeze Switch (/SZ)

Use the Squeeze switch (/SZ) 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 /SZ switch has no effect on object libraries or universal libraries.

The /SZ switch can be specified in either of two formats:

- Global format - The /SZ switch is appended to the library file specification. All of the input files are assumed to contain modules to be squeezed.
- Local format - The /SZ switch is appended to an input file specification. The /SZ switch works only on the file to which you append it.

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 the global 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 specification.

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 noncomment text and you want comments squeezed, code a dummy comment for that line. The /SZ switch will use only the rightmost comment during squeeze processing.

Example

Figure 10-14 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

        .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

        .MACRO  MOVSTR RX,RY,?LBL
;***    - - NOTE :
;       BOTH ARGUMENTS MUST BE REGISTERS
LBL:    MOVB    (RX)+,(RY)+
        BNE     LBL
        DEC     RY
        .ENDM
```

ZK-197-81

Figure 10-14 MACRO Listing Before
and After Running LBR with /SZ Switch

10.6 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. Default switch (/DF)
2. Create switch (/CR)
3. Delete switch (/DE)
4. Delete Global switch (/DG)
5. Modify Header switch (/MH)
6. Insert (/IN), Replace (/RP), Selective Search (/SS), Squeeze (/SZ), Entry Point (/EP) switches
7. Compress switch (/CO)
8. Extract switch (/EX)
9. List switches (/LI), /LE, /FU), Spool switch (/SP)

Example

```
LBR>FILE/DE:XYZ:$A,LP.LST:/LE/FU=MODX,MODY/RP
```

Functions, performed in order, are:

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.

10.7 LBR ERROR MESSAGES

LBR returns two types of error messages: diagnostic and fatal.

Diagnostic error messages describe a condition that requires consideration, but the nature of the condition does not warrant termination of the command. Diagnostic messages are issued to your terminal in the format:

```
LBR -- *DIAG* - message
```

Fatal error messages describe a condition 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 your terminal 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 your terminal and LBR returns to the highest level of command input.

10.7.1 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 insertion process, the library is rewritten after the insertion of all modules in each individual input file, that is, between input files.

10.7.2 LBR Error Messages

LBR -- BAD LIBRARY HEADER

Explanation: The file is not a library file or it 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, you should run the file structure verification utility (VFY) against the volume to determine if it is corrupted (see Chapter 9).
- 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 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 was 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, you can delete the duplicate entry point from the library and reenter the command line.

LBR -- DUPLICATE MODULE NAME "name" IN filename

Explanation: An attempt was made to insert (without replacing) 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 with the /RP switch appended to the input file specification.

LBR -- EPT OR MNT EXCEEDED IN filename

Explanation: The EPT or MNT table limit was reached during the execution of an insert or replace operation.

User Action: Copy the library, increasing the table space by means of 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 input library file, or more than one 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 (SPR) along with the related console dialog and any other pertinent information.

LBR -- ILLEGAL DEVICE/VOLUME
command line

Explanation: The Device specifier entered does not conform to syntax rules. A device specifier consists of two ASCII characters, followed by one or two optional octal digits.

User Action: Reenter the command line with the correct device syntax specified and followed by a colon.

LBR -- ILLEGAL DIRECTORY
command line

Explanation: The 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 ([ggg,mmm]).

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 (SPR) along with the related console dialog 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 10.6.

User Action: Reenter the command line, specifying the switches in the proper combination.

LBR -- INDIRECT COMMAND SYNTAX ERROR
command line

Explanation: An indirect command 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 was made to exceed one level of indirect command files.

User Action: Rerun the job with only one level of indirect command file specified.

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 exists in the Executive.

User Action: Determine which of the 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 for the task size.

User Action: Remove the task (LBR), install it in a larger partition, and reenter the command line. (See the MCR INSTALL 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 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 LIBRARY TYPE SPECIFIED

Explanation: An invalid library type was specified when using the Create or Default switch. The values OBJ, MAC, and UNI are the only valid specifications. See Sections 10.5.2 and 10.5.4.

User Action: Reenter the command line with OBJ, MAC, or UNI specified.

LBR -- INVALID MODULE FORMAT in 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 NAME -- "name"

Explantion: A module name that contains a non-Radix-50 character was specified for deletion, insertion, or replacement of a module in a universal library or in a macro module; or a module name was 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 OPERATION FOR OBJECT AND MACRO LIBRARIES

Explanation: Module header information was supplied for an object library or macro library in an insert or replace operation.

User Action: No action required. The command will be executed as if the information had not been supplied.

LBR -- INVALID RAD50 CHARACTER IN "character string"

Explanation: A character supplied as part of information when using the Insert, Replace, or Modify Header switches 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 conditions caused the message and correct that condition. Reenter the command line.

LBR -- LIBRARY FILE SPECIFICATION MISSING

Explanation: A command line 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 9).

LBR -- MULTIPLE MODULE EXTRACTIONS NOT PERMITTED FOR UNI 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 or the library file 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 exists in the 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 exists in the Executive.

User Action: Determine which of the 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: A positioning error has occurred on the input file. One of the following conditions exist:

- A problem exists on the physical device (for example it is not cycled up).
- The file is corrupted or the format is wrong.

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

LBR -- RMS MODULES CANNOT BE EXTRACTED TO RECORD ORIENTED DEVICES

Explanation: An attempt was made to extract a module inserted from a nonsequential RMS file to a record-oriented device. 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) had not been specified.

LBR -- VIRTUAL STORAGE REQUIREMENT EXCEEDS 65536 WORDS

Explanation: This error may occur if you are working with maximum size libraries and you specify a single command line that first logically deletes a large number of modules and entry points, then replaces them with an equally large number of modules and entry points that have names much different from those being replaced. 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 LBR. 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 11

FILE DUMP UTILITY (DMP)

The File Dump Utility (DMP) enables the user to examine the contents of a specific file or volume of files. The output may be formatted in ASCII, octal, decimal, hexadecimal, or Radix-50 form and dumped to any suitable output device such as a line printer, terminal, magnetic tape, DECtape, or disk.

You can dump the header and/or virtual blocks of a file or only the virtual records of a file. If you are dumping a volume, a range of logical blocks may be specified (see Sections 11.1 and 11.2). DMP normally handles blocks of up to 256(10) words in length. If the maximum block size exceeds 256(10) words, DMP's buffer size must be increased as follows:

```
    >RUN $DMP /INC=n
```

The value of the variable n must be equal to a multiple of 8 that is equal to or greater than the maximum length block or record minus 256(10) words.

The same restriction applies when dumping records greater than 512(10) bytes. In this case, you must increase DMP's buffer size by altering the DMPBLD.CMD or DMPANSBLD.CMD file as required, then rebuilding the task. See the appropriate system generation manual and your system manager for more specific information on altering various features of DMP.

You can invoke DMP by using any of the methods described in Chapter 1. After DMP is invoked, it prompts:

```
    DMP>
```

DMP is now ready for user input in the form of a command line.

DMP operates in two basic modes: file mode and device mode. File mode is used to dump virtual records or virtual blocks, and device mode is used to dump logical blocks.

11.1 FILE MODE

In file mode, one input file is specified, and all or a specified range of virtual blocks are dumped. You can also dump all the virtual records of a specified file in this mode. The input device must be either a Files-11 formatted disk or a magnetic tape. The volume must be mounted using the MCR MOUNT Command.

NOTE

If the input medium is magnetic tape, DMP must be built with ANSLIB instead of SYSLIB.

In file mode, you can specify that data be dumped one record or one block at a time. A virtual block or record refers to one block or record of data in a file. Virtual blocks and records are numbered sequentially from 1 through n, where n is the total number of blocks or records in the file. Virtual block 0 contains the header of the file. Use the /BL:n:m switch to dump virtual blocks and the /RC switch to dump virtual records. The /BL and /RC switches are mutually exclusive. (DMP switches are listed in Table 11-1.)

11.2 DEVICE MODE

In device mode, only the input device is specified, and a specified range of logical blocks is dumped. The /BL:n:m switch is a required parameter in this mode.

A logical block refers to a physical 512-byte block on disk or DECtape and to physical records on magnetic tape or cassette. Logical blocks are numbered from 0 to n-1, where n is the total number of logical blocks on the device.

On RSX-11M, when DMP is in device mode, do not mount the volume.

On RSX-11M-PLUS, however, the volume must be mounted foreign.

11.3 DMP COMMAND FORMAT

The command line for DMP is in the following format:

```
[outfile][/sw][/sw...]=inspec[/sw][/sw...]
```

outfile

Specifies the output file. If the output file name and file type are unspecified, DMP creates the file DMPFIL.DMP. If the file type is .DMP, the file will be deleted after it is printed. TI: and LP: (for terminal and line printer) are also acceptable outfile specifications.

On RSX-11M, if you specify LP:, the file will not print if the print despooler is active.

On RSX-11M-PLUS, transparent spooling is implemented. 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.

/sw

Specifies one of the switches listed in Table 11-1. 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. DMP will allow multiple dumps in a single command line. Therefore, any or all of the current format switches may be specified. Certain switches are mutually exclusive. For example, the /HX, /LW, and /WD switches, are mutually exclusive hexadecimal dump switches. The first one in the following order will be the only one executed: /LW, /WD, /HX.

inspec

Specifies the input device and file or input device only. In file mode, the equal sign and the input file name and file type are required because DMP does not provide a default for either of them. However, the input file version number defaults to the latest version and the device defaults to SY: and the current UIC.

In device mode, the equal sign and input device are required as is the /BL:n:m switch which specifies the range of logical blocks to be dumped.

For a complete description of file specifications, see Chapter 1.

11.4 DMP SWITCHES

DMP switch specifications consist of a slash (/) followed by a switch name, optionally followed by a value. The value is separated from the switch by a colon (:). DMP functions are implemented by the switches described in Table 11-1.

Table 11-1
DMP Switches

Switch	Description
Default	The default is a word mode octal dump, which is spooled to the line printer.
/AS	Specifies that the data should be dumped one byte at a time in ASCII mode. The control characters (0-37) are printed as a circumflex (^), followed by the alphabetic character corresponding to the character code plus 100. For example, bell (code 7) is printed as ^G (code 107). Lowercase characters (140-177) are printed as a percent sign (%), followed by the corresponding uppercase character (character code minus 40), unless the /LC switch is specified. Note that the /AS and /OCT switches are mutually exclusive when dumping bytes.
/BA:n:m	Specifies a 2-word base block address (the initial base address is 0,0), where n is the high-order base block address (octal), and m is the low-order base block address (octal). The address may also be specified in decimal by using a period after the number. All future block numbers specified by the /BL switch will be added to this value to obtain

(continued on next page)

Table 11-1 (Cont.)
DMP Switches

Switch	Description

/BA:n:m (Cont.)	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. Once the /BA switch is specified, it remains in effect until it is used again to set a new base address.
	
	When the /BA switch appears in a command line, no blocks are dumped. The only result of the command line 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. The values of n and m must not exceed 16 bits. In file mode only, the /BL switch is not required. If the /BL switch is not specified, DMP will dump all blocks of the specified file, relative to the current base address. If /BL:n:m is specified in file mode, it specifies the range of virtual blocks to be dumped. If /BL:n:m 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 switch and the /RC switch are mutually exclusive. 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. The value n represents the block number of the first

(continued on next page)

Table 11-1 (Cont.)
DMP Switches

Switch	Description
/BL:n:m (Cont.)	block dumped. Successive blocks are labeled with a block number one higher than the preceding block number. The dump will continue until the block labeled m is dumped.
/BY	Specifies that the data be dumped in octal byte format.
/DC	Specifies that the data be dumped in decimal word format.
/DENS:n	Specifies the density of an input magnetic tape with 800, 1600, and 6250 bpi capability, when DMP is in device mode only. The value for n can be 800, 1600, or 6250. DMP does not automatically determine the density of an input tape. If the /DENS switch is not specified in a DMP command line, DMP attempts to read an input tape at the density currently set in the tape controller. (See the <u>RSX-11M/M-PLUS MCR Operations Manual</u> and the <u>RSX-11M/M-PLUS Command Language Manual</u> for descriptions of the MOUNT command and its /DENS switch.)
/FI:file-number:sequence-number	In File Mode, the file number can be used instead of a file name as a file specification for input.

/HD[:F or :U]	This switch is an optional parameter used in File Mode. If specified, the /HD switch causes the file header as well as the specified or implied portion of the file to be dumped.
	
	Example:
	
	`DMP>TI:=JMF.DAT/HD/BL:5:6`
	
	This example dumps the header of JMF.DAT in header format and virtual blocks 5 and 6 in octal format.
	
	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`

(continued on next page)

Table 11-1 (Cont.)
DMP Switches

Switch	Description
/HD[:F or :U] (Cont.)	File headers are described in the <u>IAS/RSX-11 I/O Operations Reference Manual</u>.
/HF	Specifies the format for data blocks that have the 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.
/HX	Specifies that the data be dumped in hexadecimal byte format. Note that a hexadecimal dump reads from right to left. (See also the /LW and /WD switches.)

/ID	Causes DMP's version to be identified. This switch may be specified on a command line by itself at any time.
	
	Example:
	
	`DMP>/ID`
	`DMP--DMP VERSION M07.1B`
	
	For DMP built with ANSLIB, the response is
	
	`DMP--DMP VERSION M07.1B (ANSI)`

/LB	Requests logical block information for a file. The starting block number and a contiguous or noncontiguous indication for the file are displayed.
	
	Example:
	
	`DMP>TI:=DK0:RICKSFILE.DAT;3/LB`
	`STARTING BLOCK NUMBER = 0,135163 C`
	
	The file RICKSFILE.DAT, version 3, is a contiguous file starting at block number 0,135163. (See /BA:n:m for block number description.)

/LC	Specifies that the data should be dumped in lowercase characters. This switch can only be used if the output device has lowercase capability.

(continued on next page)

Table 11-1 (Cont.)
DMP Switches

Switch	Description
/LW	Specifies that the data be dumped in hexadecimal double-word format.
/MD[:n]	Specifies a memory dump and allows control of line numbers. Line numbers are normally reset to zero whenever a block boundary is crossed. The /MD 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 specification.
/OCT	Specifies that the data should be dumped in octal format in addition to other formats specified. If no DMP format switches are specified, the default is octal. The /AS switch and the /OCT switch are mutually exclusive when dumping bytes.
/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 any of these format switches: /AS, /DC, /HX, /LW, /R5, or /WD. The largest record that DMP can process is limited by the amount of space available to the DMP task. DMP's task image has 512(10) bytes allocated to it initially. To increase the amount of space available, use the MCR INSTALL command /INC switch. For example, to dump a file with 1024-byte records, you must specify /INC=256. (at least). If the input is a tape and the block size is greater than 512 bytes, $FSR1 must be expanded as described in the DMPBLD.CMD or DMPANSBLD.CMD file. The /RC switch and the /BL switch are mutually exclusive.
/RW	Causes DMP to issue a rewind command before referencing a specified tape. This switch may be specified at any time to reposition a tape at the beginning-of-tape (BOT).

(continued on next page)

Table 11-1 (Cont.)
DMP Switches

Switch	Description
/SB:n or /SB:-n	Specifies the number of blocks DMP spaces forward (n) or backward (-n) on tape. DMP stops when it senses end-of-tape (EOT). DMP will space only single blocks beyond EOT.
/SF:n or /SF:-n	Specifies the number of end-of-file (EOF) marks DMP spaces forward (n) or backward (-n) on tape. DMP stops when it senses EOT or BOT. DMP will space only single EOF marks beyond EOT.
/SP	Causes the dump output file to be spooled to the line printer. The /SP switch 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.

11.5 DMP EXAMPLES

Three examples of dump listings are included in this section to illustrate how the various DMP switches can be used. DMP edits blocks or records 16(10) bytes at a time. The dump includes the indicated number of valid bytes in the block or record. The remaining number of bytes are listed as null bytes (0).

11.5.1 A Multiple Format Dump

The command line shown in Example 11-1 dumps virtual blocks 5 and 6 of DSC.MAC in hexadecimal, Radix-50, and decimal format. Each line of the output file will appear in three different formats.

Example 11-1 Dumping Virtual Blocks in Hexadecimal, Radix-50, and Decimal Format

```
DMP>DOC.DMP=[301,357]DSC.MAC/HX/R5/DC/BL:5:6
```

The contents of DOC.DMP are:

```
DUMP OF DB0:[301,357]DSC.MAC;1 - FILE ID 17725,11,0
           VIRTUAL BLOCK 0,000005 - SIZE 512. BYTES


4E 41 4D 4D 4F 43 20 41 20 3B 00 1E 53 45 52 49    0000    ;Hexadecimal
000000    MFY ML7   O EFK EFQ L$K LN/ LT3                  ;Radix-50
0.  21065. 21317. 00030. 08251. 08257. 20291. 19789. 20033.;Decimal

53 45 53 53 45 43 4F 52 50 20 44 4E 41 20 2C 44    0010
000020    GCL JP2 J7F L22 L$Z KCK MMK ML7
16. 11332. 16672. 17486. 20512. 20306. 17731. 21331. 21317.

53 52 49 46 20 3B 00 39 00 3B 00 01 2E 54 49 20    0020
      .
      .
      .

DUMP OF DB0:[301,357]DSC.MAC;1 - FILE ID 17725,11,0
           VIRTUAL BLOCK 0,000006 - SIZE 512. BYTES


20 44 4E 46 55 42 24 20 51 45 20 30 52 20 46 49    0000
000000    KI3 MEX EF  M E E1H MYZ LT8 EFT
0.  17993. 21024. 08240. 20805. 09248. 21826. 20038. 08260.

44 4E 45 09 00 09 50 4F 4F 4C 20 45 56 41 45 4C    0010
000020    KCT M2A EFU L$T L39   I KA3 J7F
16. 17740. 22081. 08261. 20300. 20559. 00009. 17673. 17486.

54 53 24 20 54 45 4C 09 00 2B 00 50 4F 4F 4C 20    0020
      .
      .
      .
```

11.5.2 A Record Dump

The command line shown in Example 11-2 dumps all of the virtual records of YACHT.SEQ in ASCII and decimal word format.

Example 11-2 Dumping Virtual Records in ASCII and Decimal Word Format

```
DMP>REC.DMP=[301,357]YACHT.SEQ/RC/AS/DC
```

The contents of REC.DMP are:

```
DUMP OF DB0:[7,337]YACHT.SEQ;1 - FILE ID 15451,35,0
          RECORD NUMBER 01. - SIZE 41. BYTES

000000     A  L  B  E  R  G              3  7     M  K
0.       19521. 17730. 18258. 08224. 08224. 14131. 19744. 08267.

000020     I  I        K  E  T  C  H     3  7     2  0  0
16.      18761. 08224. 17739. 17236. 08264. 14131. 12832. 12336.

000040     0  0  1  2  3  6  9  5  1 ^@ ^@ ^@ ^@ ^@ ^@ ^@
32.      12336. 12849. 13875. 13625. 00049. 00000. 00000. 00000.

          RECORD NUMBER 02. - SIZE 41. BYTES

000000     A  L  B  I  N                 7  9
0.       19521. 18754. 08270. 08224. 08224. 14647. 08224. 08224.

000020                 S  L  O  O  P     2  6     0  4  2
16.      08224. 08224. 19539. 20303. 08272. 13874. 12320. 12852.

000040     0  0  1  0  1  7  9  0  0 ^@ ^@ ^@ ^@ ^@ ^@ ^@
32.      12336. 12337. 14129. 12345. 00048. 00000. 00000. 00000.
     .
     .
     .
```

11.5.3 A Header Dump

The command line shown in Example 11-3 dumps only the header of DSC.MAC.

Example 11-3 Dumping the File Header of a File

```
DMP>DHR.DMP=[301,357]DSC.MAC/HD/BL:0
```

The contents of DHR.DMP are:

```
DUMP OF DB0:[301,357]DSC.MAC;1 - FILE ID 17725,11,0
                                   FILE HEADER

HEADER AREA
   H.IDOF          027
   H.MPOF          056
   H.FNUM,
   H.FSEQ          (17725,11)
   H.FLEV          401
   H.FOWN          [301,357]
   H.FPRO          [RWED,RWED,RWED,R]
   H.UCHA          000=
   H.SCHA          000 =
   H.UFAT
           F.RTYP  002 = R.VAR
           F.RATT  002 =   FD.CR
           F.RSIZ  116 = 78.
           F.HIBK  H:0 L:000040 = 32.
           F.EFBK  H:0 L:000040 = 32.
           F.FFBY  532 = 346.
           (REST)
           000000 000000 000000 000000 000000 000000 000000 000000
           000000
IDENTIFICATION AREA
   I.FNAM,
   I.FTYP,
   I.FVER          DSC       .MAC;1
   I.RVNO          1
   I.RVDT          13-OCT-80
   I.RVTI          09:52:46
   I.CRDT          13-OCT-80
   I.CRTI          09:52:45
   I.EXDT          --
MAP AREA
   M.ESQN          000
   M.ERVN          000
   M.EFNU,
   M.EFSQ          (0,0)
   M.CTSZ          001
   M.LBSZ          003
   M.USE           014 = 12.
   M.MAX           314 = 204.
   M.RTRV
   SIZE    LBN
   12.     H:000 L:036215 = 15501.
   3.      H:000 L:036235 = 15517.
   1.      H:000 L:036250 = 15528.
   2.      H:000 L:036272 = 15546.
   3.      H:000 L:036313 = 15563.
   11.     H:000 L:036411 = 15625.
CHECKSUM
   H.CKSM          122620
```

11.6 DMP ERROR MESSAGES

DMP -- BAD DEVICE NAME

Explanation: An incorrect device name was entered 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 must be specified.)

User Action: Reenter the command line specifying the /BL switch.

DMP -- CANNOT FIND INPUT FILE

Explanation: The requested file cannot be located in the specified directory.

User Action: Reenter the command line specifying the correct file name and UFD.

DMP -- COMMAND SYNTAX ERROR

Explanation: A command line was entered in a format that does not conform to syntax rules.

User Action: Reenter the command line specifying the correct syntax.

DMP -- FAILED TO ASSIGN LUN

Explanation: An illegal device was entered in a file specification.

User Action: Reenter the command line specifying the correct device.

DMP -- FAILED TO READ ATTRIBUTES

Explanation: A file was specified for which you did not have read access privileges.

User Action: Rerun DMP under a UIC that has read access privileges to the file.

DMP -- ILLEGAL DENSITY VALUE

Explanation: A value other than 800, 1600, or 6250 bpi was specified with the DMP /DENS switch.

User Action: Reenter the /DENS switch with the proper value.

DMP -- ILLEGAL SWITCH

Explanation: A switch was specified that is not a valid DMP switch, or a legal switch was used in an invalid manner.

User Action: Reenter the command line specifying the correct switch.

DMP -- ILLEGAL USE OF /RC SWITCH

Explanation: The /RC switch can be used only in file mode (see Section 11.1).

User Action: Reenter the command line specifying a file name.

DMP -- ILLEGAL VALUE ON HD SWITCH

Explanation: An option was entered other than F or U for the /HD switch.

User Action: Reenter the command line specifying the correct option.

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, the device cycled down).
- The file is corrupted or the format is incorrect.
- The 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 specifying an input file.

DMP -- NO LISTS OR WILD CARDS ALLOWED

Explanation: Either a command line 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, specifying 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 exists:

- The file is protected against access.
- A problem exists on the physical device (for example, the 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 exists:

- The file is protected against access.
- A problem exists on the physical device (for example, the device cycled down).
- The named file does not exist in the specified directory.
- The volume is not mounted.
- The specified file directory does not exist.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

CHAPTER 12

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, 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 Source Language Input Program utility (SLP). 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 13.)

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 specify the number of lines required for CMP to consider that a match has been 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 in the command line. The defaults are:

SY0:	The user's default system device
[current UIC]	The UIC that CMP is running under
FILCOM	The default file name
.LST	The default file type

However, if you do not specify an output file, the output defaults to your terminal. For example:

```
CMP>FILE1.MAC,FILE2.MAC
```

CMP lists the differences between FILE1.MAC and FILE2.MAC on your terminal. If you type the equal sign but give no output file specification, only the total number of differences is output to your terminal. For example:

```
CMP>=FILE1.MAC;1,FILE2.MAC;1
10 differences found
```

/sw...

Switches that you apply to the output file specification. Some of the switches can be negated and some are mutually exclusive. Section 12.1 contains this information.

infile1

The file specification for the input file to be compared to infile2. 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. You do not have to have a complete file specification. The specifications for infile1 are used as defaults for any unspecified portions of infile2. For example:

```
CMP> DB2:[42,10]EXEC,;2
```

CMP interprets the second input file as DB2:[42,10]EXEC.MAC;2.

If you do not specify a file version number, the default is the most recent version of the file.

You can invoke CMP using any of the methods for invoking a utility described in Chapter 1.

12.1 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 after the output file in 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 switch.
/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. When a section of lines in infile1 has been deleted in infile2 (the output listing file), the first line not deleted is marked. /-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 later.
/CO /-CO	Specifies that CMP include comments (that is, text preceded by a semicolon) in compare processing. /CO is the default switch.

/DI /-DI	Specifies that CMP list the differences between the two files (rather than marking the lines in infile2). /DI is the default 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 switch.
/LI:n	Specifies that a number (n) of lines must be identical before CMP recognizes a match. /LI:3 is the default switch. When it encounters a match, CMP lists 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, /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 after the /SL switch (:au), an audit trail is specified for SLP input. Section 12.2.3, gives an example of how CMP generates SLP command input. (For information on how SLP processes source files, refer to Chapter 13.) /-SL is the default switch.
/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 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 switch.
/VB:nnn	Specifies an octal character code for the character you want to 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). /VB:041 for an exclamation mark is the default switch.

CMP default switch settings are listed in Table 12-1.

Table 12-1
Summary of CMP Default Switch Settings

/-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 blank 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.

12.2 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 infilel 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 the examples in the subsequent sections, the following files are used as infilel (TEST1.DAT;1) and infile2 (TEST2.DAT;1):

```
DB0:[7,7]TEST1.DAT;1     DB0:[7,7]TEST2.DAT;1

      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
```

12.2.1 Differences Format

If you enter a command line and do not specify any switches, CMP lists the differences between the two files on your terminal or in an output file. 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 format of output generated without any switches. The output file is generated with the CMP command:

```
CMP>TESTDIF.DAT=TEST1.DAT,TEST2.DAT

*********************************************
   1)   DB0:[7,7]TEST1.DAT;1
     6    12345
     7    23456
     8    34567
     9    LINE9
*****************
   2)   DB0:[7,7]TEST2.DAT;1
     6    45678
     7    56789
     8    67891
     9    LINE9
*********************************************
   1)   DB0:[7,7]TEST1.DAT;1
*****************
   2)   DB0:[7,7]TEST2.DAT;1
    13    EXTRA
    14    EXTRA
    15    EXTRA

       2 differences found
```

The input files are TEST1.DAT and TEST2.DAT, which are shown in Section 12.2. 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 a long line of asterisks.) The short line of asterisks separates the pair of differences that comprise the set.

Note that because /LI:n was not specified, the number of lines required for a match defaults to 3. Thus, CMP found two differences.

12.2.2 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 the format of output with change bars applied to lines from two files that do not match line for line. The output file is generated with the CMP command:

```
CMP>TESTDIF.DAT/CB=TEST1.DAT,TEST2.DAT
```

Notice that the change bar is applied to the first line of match (line 9).

```
 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
```

12.2.3 SLP Command Input Format

You 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 13 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:

```
CMP>TESTDIF.DAT/SL:BLS001=TEST1.DAT,TEST2.DAT

-6,8,/;BLS001/
45678
56789
67891
-12,,/;BLS001/
EXTRA
EXTRA
EXTRA
/
```

12.3 CMP MESSAGES

This section lists the CMP 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 syntax.

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: Reenter the command line.

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 reenter the command line.

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 line using a legal switch or switch value.

CMP -- Open failure on input file #1

Explanation: CMP could not open the first input file.

User Action: Check the file specification for first input file and reenter the command line using the correct file specification.

CMP -- Open failure on input file #2

Explanation: CMP could not open the second input file.

User Action: Check the file specification for second input file and reenter the command line using the correct file specification.

CMP -- Open failure on output file

> **Explanation:** CMP could not open the specified output file.
>
> **User Action:** Check the file specification for the output file and reenter the command line using the correct file specification.

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 description of the INSTALL command in the _RSX-11M/M-PLUS MCR Operations Manual_.)

CHAPTER 13

SOURCE LANGUAGE INPUT PROGRAM (SLP)

The Source Language Input Program (SLP) is a utility used to maintain and audit source files. The optional audit trail in the output files allows you to keep a record of maintenance changes.

SLP is invoked by 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 a 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 line numbers or character strings that you specify. Command input can come directly from your terminal or from an indirect command file containing commands and text lines to be inserted into the file. SLP accepts data from any Files-11 volume.

Output from SLP is a listing file and the 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 file and permanently applied to the output file.

You can control SLP processing with SLP control switches. These switches allow you to:

- 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
- Spool files to a Files-11 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, you should not specify TI: as your output file, because when you finish editing, you will not have a copy of the output file and the input file will be the same as before you began editing.

13.1 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 made to the file.

13.1.1 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, these changes are noted in the output files.

13.1.2 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, where the first line entered must 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, where 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/
CMP (R1)+,B
-4,4
DEC R2
/
```

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 8(10); no more than 14(10) characters may be specified. (See Section 13.4.2 for more information about the audit trail.)

The first line is the command line, where you define 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 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 command line. Wildcards cannot be used in any of the file specifications.

You can also calculate the checksum value for the edit commands. Specify the checksum switch with either the input or output file specification in the format:

```
/CS[:n]
```

The checksum value can be calculated for all SLP edit command lines. The checksum value cannot be calculated for the following:

- 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 the first character of the last audit trail string before the current 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 in Section 13.5.2)

13.1.3 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 this chapter contain samples of listing files.

13.1.4 The SLP Output File

The SLP output file is the updated input file. All of the updates specified by 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.

13.2 HOW SLP PROCESSES FILES

Figure 13-1 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
CMP (R1)+,B
-4,4
DEC (R2)
/
```

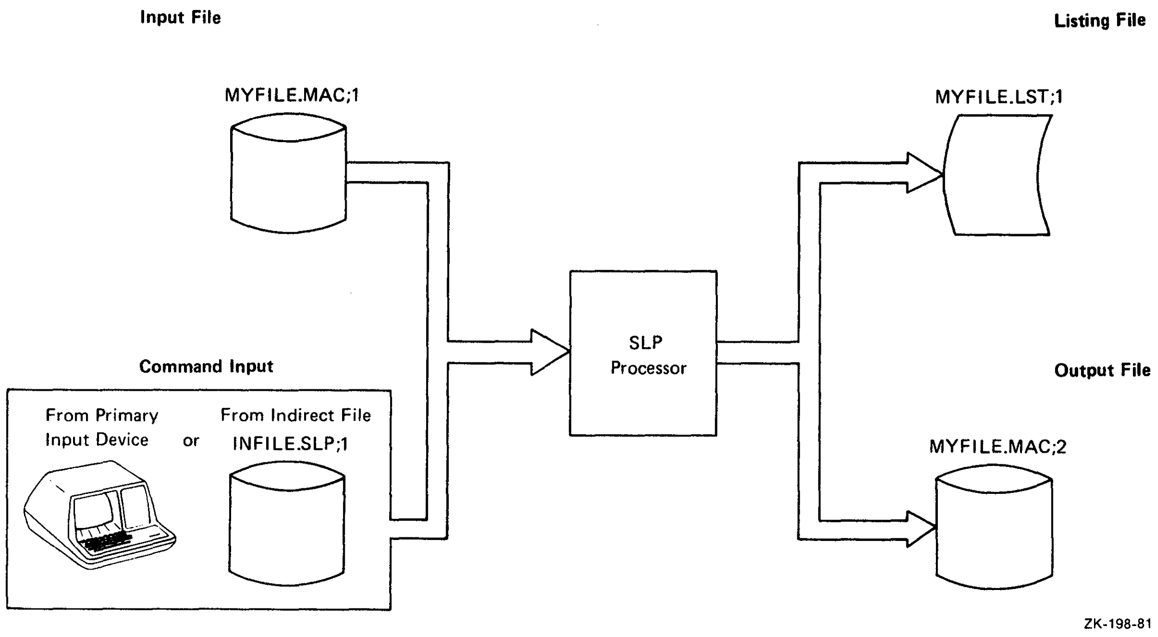


Figure 13-1 Input Files and Output Files Used During SLP Processing

This is the input file (MYFILE.MAC;1) before SLP processes the files:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (R0)+,A
BNE     20$
INC     R1
```

The following is the listing file (MYFILE.LST;1) resulting from SLP processing of these files:

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  CMP     (R1)+,B          ;**NEW**
 5.  DEC     (R2)             ;**NEW**
 6.  BEQ     END              ;**-1
 7.  CLR     R1
 8.  MOV     R2,NUMC
 9.  CMPB    (R0)+,A
10.  BNE     20$
11.  INC     R1
```

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 13-1, 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 modifies the line, notes the change by means of the audit trail, and then continues writing lines to the output file until another command is encountered or until end-of-file is reached.

13.3 USING SLP

This section describes how to:

- Specify the SLP edit commands
- Update files using the SLP edit commands
- Enter SLP commands interactively and by means of indirect command files
- Create a source file using SLP

13.3.1 Specifying SLP Edit Commands

The SLP edit commands allow you to update source files by adding, deleting, and replacing lines in a file. 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. 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)

    Identifies 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 later in this section.

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 later in this section.

/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 default 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 where the 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 where the two character strings delimit a larger character string abbreviated by an ellipsis (...).

number

    A decimal line number where the current line pointer is to be moved. The largest line number that can be specified is 9999.

. (period)

The current line.

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.

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 lines.

13.3.2 Entering SLP Edit Commands

Once you have invoked SLP, you can enter SLP edit commands interactively or by specifying 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:

```
MOV     #$SWTCH,R3
CLR     $ERFLG
CLR     $CRCVL
CLR     $CSSV
MOV     SPSAV,SP
MOV     #$CFNMB,R0
MOV     #<$HDSIZ-$CFNMB>/2+1,R1
CLR     (R0)+
DEC     R1
BNE     5$
```

13.3.2.1 Entering SLP Commands Interactively

To alter the example file interactively, first invoke SLP by using any method described in Chapter 1 (except >SLP filespec).

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 defining 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
TST     R1
-4,4
BEQ     10$
-6,,/;JM010/
CLR     R2
/
```

In this example, the edit commands instruct SLP to do the following: -3 inserts a new line after line 3; -4,4 deletes line 4 and replaces it with a new line; -6,,/;JM010/ inserts a line after line 6 with a new audit trail value.

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 another input file and begin another editing session.

The listing file (BASE.LST;1) resulting from SLP processing appears as follows:

```
 1.  MOV     #$SWTCH,R3
 2.  CLR     $ERFLG
 3.  CLR     $CRCVL
 4.  TST     R1                             ;**NEW**
 5.  BEQ     10$                            ;**NEW**
 6.  MOV     SPSAV,SP                       ;**-1
 7.  MOV     #$CFNMB,R0
 8?  CLR     R2                             ;JM010
 9.  MOV     #<$HDSIZ-$CFNMB>/2+1,R1
10.  CLR     (R0)+
11.  DEC     R1
12.  BNE     5$
```

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 (.) after the line number for 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.

13.3.2.2 **Entering SLP Commands Using Indirect Command Files** - To alter the example file by using the SLP edit commands 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 three. This permits indirect command files to reference a text file.

13.3.2.3 **Using SLP Operators** - In addition, you can enter special characters called operators, which perform specific functions. Table 13-1 lists the operators and the function each performs. Enter operators, in edit mode, as the first character of an input line.

Table 13-1
SLP Operators

Operator	Function
-	Identifies the first character of a SLP edit command line.
\	Suppresses audit-trail processing.
%	Reenables audit-trail processing.
@	Invokes an indirect command file for SLP processing.
/	Terminates the SLP edit session, and then returns to SLP command mode.
<	Enables 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. This character can be used with all SLP operators.

13.3.3 Updating Source Files With 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.

13.3.3.1 **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 will produce and infile is the name of the input file whose lines are to be numbered. The slash (/) terminates edit mode. For example, suppose the input file is:

```
MOV     R1,-(SP)
BIC     #177770,@SP
ADD     #60,@SP
MOVB    (SP)+,-(R0)
ASR     R1
ASR     R1
ASR     R1
DEC     R2
BNE     30$
MOV     #MSG,R0
```

SLP processes each line to generate a numbered list file (list file;1):

```
 1.  MOV     R1,-(SP)
 2.  BIC     #177770,@SP
 3.  ADD     #60,@SP
 4.  MOVB    (SP)+,-(R0)
 5.  ASR     R1
 6.  ASR     R1
 7.  ASR     R1
 8.  DEC     R2
 9.  BNE     30$
10.  MOV     #MSG,R0
```

13.3.3.2 **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
    .
    .
    .
```

The following example shows how to add lines to a file. The command input consists of the following lines:

```
MYFILE.MAC;2/AU:48.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1
-3
CMP     (R1)+,B
-4,4
DEC     R2
-6,,,/;JM010/
INC     R3
-9,,/;BJ008/
BEQ     10$
/
```

The next example uses text rather than line numbers to indicate where new lines should be added or deleted:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/BEQ/
CALL    WRITE
/
```

In this example, the edit command /BEQ/ instructs SLP to insert a line after the line with the first occurrence of BEQ.

SLP processing generates the following listing file (MYFILE.LST;1):

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  TST     R2
 5.  BEQ     END
 6.  CALL    WRITE                  ;**NEW**
 7.  CLR     R1
 8.  MOV     R2,NUMC
 9.  CMPB    (R0)+,A
10.  BNE     20$
11.  INC     R1
```

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 BEQ.

The next example uses the same input file and the following new command lines:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/#SIZ/+2
CMP     (R1)+,B
/
```

SLP processing generates the following listing file (MYFILE;1):

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  TST     R2
 5.  CMP     (R1)+,B                ;**NEW**
 6.  BEQ     END
 7.  CLR     R1
 8.  MOV     R2,NUMC
 9.  CMPB    (R0)+,A
10.  BNE     20$
11.  INC     R1
```

Again, SLP has numbered the lines and this time the new input line is inserted so that it is two lines beyond the line containing the first occurrence of the string /#SIZ/.

13.3.3.3 **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 13.3.1; 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. The input file consists of the following lines:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (R0)+,A
BNE     20$
INC     R1
```

The command input consists of the following commands and text lines:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/MOV...R1/,/NUMC/
/
```

SLP processing generates the following listing file (MYFILE;1):

```
1.  MOV     #BUF1,R0
2.  CMPB    (R0)+,A
3.  BNE     20$
4.  INC     R1                      ;**-6
```

In this example, the ellipsis (...) abbreviates the larger string MOV #SIZ,R1. Assuming the two strings bracket a larger string, SLP searches for the first occurrence of the string MOV and then the first occurrence on the same line of the string R1, in this case the string MOV #SIZ,R1. 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 NUMC. 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 previous example, the following example shows how to delete a single line using the period locator. The command input for this example is:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/MOV #SIZ,R1/,.
/
```

SLP processing generates the following listing file (MYFILE;1):

```
1.  MOV     #BUF1,R0
2.  CALL    READ                    ;**-1
3.  TST     R2
4.  BEQ     END
5.  CLR     R1
6.  MOV     R2,NUMC
7.  CMPB    (R0)+,A
8.  BNE     20$
9.  INC     R1
```

SLP moves the current line pointer to the line containing the string MOV #SIZ,R1 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.

13.3.3.4 **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:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
```

The command input is:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-2,.+1
CMP     (R1)+,B
INC     R2
/
```

In this example, the edit command, -2,.+1, instructs SLP to delete lines 2 and 3 and insert two new lines.

SLP processing generates the following listing file (LISTING;1):

```
1.  MOV     #BUF1,R0
2.  CMP     (R1)+,B                 ;**NEW**
3.  INC     R2                      ;**NEW**
4.  TST     R2                      ;**-2
5.  BEQ     END
6.  CLR     R1
7.  MOV     R2,NUMC
```

13.3.4 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. An interactive editor, such as EDI or EDT, is 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). The default device is implied by the output file specification.

primary input device:

Specifies that input for the file being created is coming from this device, for example, a terminal. The default device is your primary input device.

/sw

Specifies any optional SLP switches.

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.

13.4 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.

13.4.1 SLP Switches

SLP output consists of two files -- a listing file and the output file, which is the modified version of the input file. You can use the SLP switches to control the audit trail and print options associated with the two files.

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 13-2 lists the SLP switches and gives a brief description of the functions each performs.

Table 13-2
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. See the following sections for more information about the /AU switch.
/BF /-BF	Positions 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. See Section 13.4.6 for more information about the /CM 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. The procedure SLP uses to calculate the checksum value for the edit commands is described in Section 13.1.2.
/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.

(continued on next page)

Table 13-2 (Cont.)
SLP Switches

Switch	Function
/SQ	Sequences the lines in the output file so that the numbers reflect the line numbers of the original input file. New lines added to the file have the same number as the preceding line. This allows the MACRO Relocatable Assembler to output listing files that contain the original line numbers, thus easing the process of updating correction files. If you specify a listing file, SLP preserves the line numbers of the input file but does not display numbers for the new lines that have been inserted.
/RS	Resequences the lines in the output file so that the line numbers are incremented for each line written to the output file. The /RS switch overrides the /SQ switch.
/NS	Does not sequence the lines in the output file. New lines are indicated by the audit trail (if specified). The /NS switch is the default condition and overrides the /SQ and /RS switches.

13.4.2 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 of the following command lines 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
```

LP.LST will be spooled automatically.

By default, SLP automatically generates an audit trail; that is, you need not explicitly specify the /AU switch in your command line (unless you want to specify the beginning position and length of the audit trail).

13.4.3 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).

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 8(10); no more than 14(10) characters may be specified. (See Section 13.4.2 for more information about the audit trail.)

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 following example shows how to specify the beginning position and length of the audit trail. The input file for this example is:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
```

The command input is:

```
MYFILE.MAC;2/AU:30.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1
-2,.+1,/;CHANGE001/
CMP     (R1)+,B
DEC     R2
/
```

The listing file MYFILE.LST;1 resulting from SLP processing is:

```
1.  MOV     #BUF1,R0
2.  CMP     (R1)+,B         ;CHANGE001
3.  DEC     R2              ;CHANGE001
4.  TST     R2              ;**-2
5.  BEQ     END
```

13.4.4 Changing the Value of the Audit Trail

To change the value of the audit trail, specify:

```
-[locator1],[locator2],/;new value/
```

The following example shows how to change the audit trail values. The input file consists of the following lines:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (R0)+,A
BNE     20$
INC     R1
```

The command input consists of the following commands and text lines:

```
MYFILE.MAC;2/AU:48.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1
-3
CMP      (R1)+,B
-4,4
DEC      R2
-6,,/;JM010/
INC      R3
-9,,/;BJ008/
BEQ      10$
/
```

In this example, the edit commands instruct SLP to insert a line after line 3, to delete and replace line 4, and to insert new lines after lines 6 and 9 with new audit trail values.

The listing file (MYFILE.LST) resulting from SLP processing appears as follows:

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  CMP     (R1)+,B                                   ;**NEW**
 5.  DEC     R2                                        ;**NEW**
 6.  BEQ     END                                       ;**-1
 7.  CLR     R1
 8.  INC     R3                                        ;JM010
 9.  MOV     R2,NUMC
10.  CMPB    (R0)+,A
11.  BNE     20$
12.  BEQ     10$                                       ;BJ008
13.  INC     R1
```

13.4.5 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 the /-AU switch 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-NOV-80
;
;       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.

13.4.6 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:[n]
```

n

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 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 following example, SLP truncates input lines to the default position of the audit trail, column 80(10).

```
SLP>SLPR11.MAC;12=SLPR11.MAC;11/CM
/
```

13.5 SLP MESSAGES

SLP messages are divided into two groups: information and error. The messages and suggested responses are given in the following sections. Section 13.5.1 describes the information message and Section 13.5.2 describes the error messages.

13.5.1 SLP Information Message

SLP -- COMMAND FILE CHECKSUM IS ######

Explanation: By specifying the /CS[:n] switch 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.

13.5.2 SLP Error Messages

This section lists the SLP error messages. Following each message is an explanation of the error and recommended user action to correct the error.

SLP error messages are issued in two formats:

- SLP followed by a dash, 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/CF
  ```

- SLP followed by a dash, 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
  ```

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, specifying the proper syntax.

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, specifying a legal device.

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, specifying a legal directory.

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 (SPR) with the console dialog 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, specifying a legal file name.

SLP -- *FATAL*-ILLEGAL GET COMMAND LINE ERROR

Explanation: The system was unable to read a command line. This error message 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 (SPR) with the console dialog and any other related information.

SLP -- *FATAL*-ILLEGAL SWITCH
command line segment

Explanation: The switch was not a legal SLP switch or a legal switch was used in an illegal manner. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line, specifying the legal switch.

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, specifying the proper syntax.

SLP -- *FATAL*-INDIRECT FILE DEPTH EXCEEDED
command line

Explanation: More than three levels of indirect command files were specified in an indirect command file. Open files were closed and SLP was reinitialized.

User Action: Correct the indirect 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 allowed 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 on-line).
- 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 13.3.1. 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 an indirect command file or from the terminal; for example, -990 was specified for an 800-line file.

User Action: Terminate the current editing session. Restart the editing session, and enter the edit command line, specifying 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 prompts (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 14

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 14-1 shows the processing steps involved in generating an updated task file using PAT.

14.1 SPECIFYING THE PAT COMMAND LINE

Specify the PAT command line 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

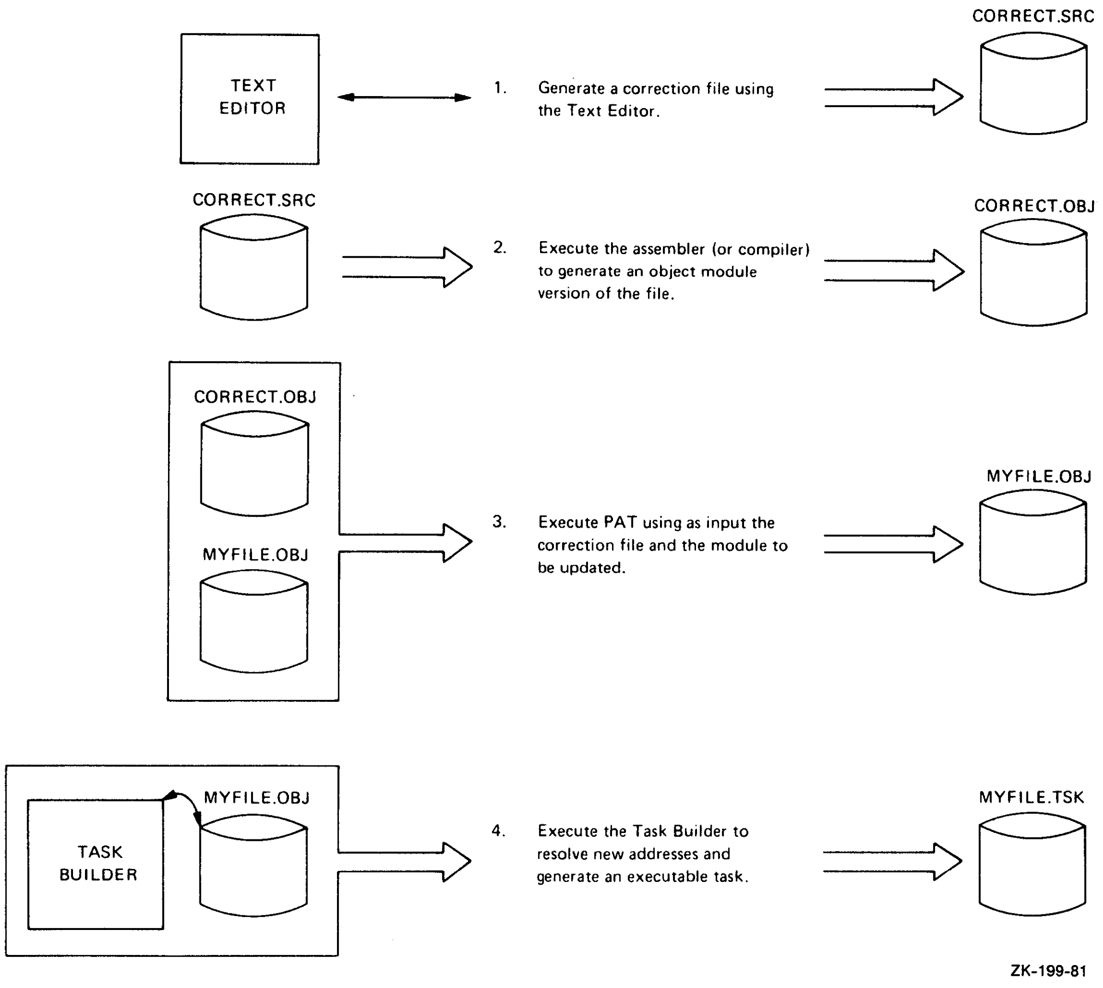
The file specification for the input file. This file can contain one or more concatenated object modules.

correctfile

The file specification for the correction file. This file contains the updates to be 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 14.2.4 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. You must then rerun PAT, specifying the correct checksum.



TEXT EDITOR
1. Generate a correction file using the Text Editor.
CORRECT.SRC
CORRECT.SRC
2. Execute the assembler (or compiler) to generate an object module version of the file.
CORRECT.OBJ
CORRECT.OBJ
MYFILE.OBJ
3. Execute PAT using as input the correction file and the module to be updated.
MYFILE.OBJ
TASK BUILDER
MYFILE.OBJ
4. Execute the Task Builder to resolve new addresses and generate an executable task.
MYFILE.TSK
ZK-199-81

Figure 14-1 Processing Steps Required to Update a Module Using PAT

14.2 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.

14.2.1 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 you execute PAT, 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.

14.2.2 The Correction File

The correction file contains the patches to be applied to the input file. PAT assumes a file type of .OBJ for the correction file. If you use a file type other than .OBJ, you must specify it explicitly in the command line.

As shown in Figure 14-1, 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 the module that you are updating for inputname.

updatenum

Any value acceptable to the MACRO-11 .IDENT assembler directive. Generally, this value reflects the updated version of the file to be processed by PAT (as shown in the examples given in Section 14.2.3).

NOTE

The .IDENT assembler directive is a required part of the correction file. Failure to include an .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.

14.2.3 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.

14.2.3.1 Overlaying Lines in a Module

14.2.3.1 Overlaying Lines in a Module - The following example illustrates a technique using a patch file to overlay lines in a module. 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
```

14.2.3.2 **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 at an appropriate location in the subroutine. The CALL instruction directs the program to branch to the new code, execute that code, and then return to in-line processing.

The input file for this 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.

14.2.4 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 to 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.

14.3 PAT 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 that describe fatal errors (errors that caused PAT to terminate), PAT uses the prefix:

```
PAT -- *FATAL*-error message
```

The following messages are grouped according to message type, as follows:

- Information messages
- Command line errors
- File specification errors
- Input/Output errors
- File content or format errors
- Internal software error
- Storage allocation error

14.3.1 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.

14.3.2 Command Line Errors

The following error messages 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 information.

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 specified an indirect command file 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.

14.3.3 File Specification Errors

The following error 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 using the correct syntax for the device or volume specification.

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 using the correct syntax for the directory specification.

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 using the correct switch or switch value.

INVALID FILE SPECIFIED filename

Explanation: You specified a file that contains one of the following errors:

- 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 specifying the correct filename.

14.3.4 Input/Output Errors

The following error messages 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 deaccessing 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 dialog 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.

14.3.5 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.

14.3.6 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 dialog and any other pertinent information.

14.3.7 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 description of the INSTALL command in the RSX-11M/M-PLUS MCR Operations Manual.)

CHAPTER 15

TASK/FILE PATCH PROGRAM (ZAP)

The Task/File Patch Program (ZAP) allows you to directly examine and modify task image and data files on a Files-11 volume. Using ZAP, you can patch these files interactively without reassembling and rebuilding the task.

ZAP supports four types of task image files:

- Regular task image files, which include those mapped to resident and supervisor mode libraries
- Multiuser task image files
- I- and D-space (instruction and data space) tasks
- Resident libraries

Note that only RSX-11M-PLUS supports tasks mapped to supervisor mode libraries, multiuser tasks, and I- and D-space tasks.

These types of task image files are discussed fully with the /List switch (Section 15.1.1).

ZAP performs many of the functions performed by the RSX-11 on-line debugging utility, ODT. Thus, working knowledge of ODT is helpful in using ZAP. See the _IAS/RSX-11 ODT Reference Manual_ for more information on ODT.

ZAP provides the following features:

- Operating modes that allow you to access specific words and bytes in a file, modify locations in a file, list the disk block and address boundaries for each overlay segment in a task image file on disk, and open a file for reading only
- A set of internal registers that include eight Relocation Registers
- Single-character commands that, with other command line elements, allow you to open and close locations in a file and to display and manipulate the values in those locations

Except in read-only mode, the results of ZAP commands are permanent. Thus, the best way to use ZAP is with a hard-copy terminal so that you have a record of the changes you make.

Although the ZAP program is relatively straightforward to use, patching locations in a task image file requires knowing how to use the map (or memory allocation file) generated by the Task Builder and the listings generated by the MACRO-11 assembler. 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 PDP-11 MACRO-11 Language Reference Manual.

15.1 ZAP OPERATING MODES AND SWITCHES

ZAP provides two addressing modes and two access modes. The addressing modes are task image mode and absolute mode. Task image mode is the default mode. The access modes are read/write mode and read-only mode. Read/write is the default mode. Either addressing mode can be used with either access mode. The modes and their associated switches are as follows:

- Task image mode is the default addressing 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. See Section 15.2 for more information on using task image mode.
- In absolute mode, specified with the /AB switch, ZAP processes the addresses you enter in ZAP command lines as absolute byte addresses within the file. You must use absolute mode for any files that are not task images. See Section 15.2 for more information on using absolute mode.
- Read/write mode is the default access mode for ZAP. In this mode, ZAP opens a file for reading and/or modification.
- In read-only mode, specified with the /RO switch, ZAP opens a file for reading but not modification. That is, you can execute ZAP functions that change the contents of locations, but these changes are not actually made to the file. When ZAP exits, the original values in the file are still there.

15.1.1 The List Switch (/LI)

When using ZAP in task image mode (but not absolute), you can also specify the /List switch (/LI). The /List switch directs ZAP to display the overlay segment table for the task image file on disk that you are working with. The table lists the starting disk block and address boundaries for each overlay segment in the file. The segment table lists are in a different format for each type of task image file. (The formats are discussed later in this section.) You use this table and the Task Builder map to locate the task segments being changed.

The /LI switch displays the overlay segment boundaries in the following format:

```
ssssss:  aaaaaa-bbbbbb [nnnnnn] [dddddd]
```

ssssss:

The starting block in octal.

aaaaaa

The lower address boundary in octal.

bbbbbb

The upper address boundary in octal.

nnnnnn

The segment name that appears for I- and D-space tasks; manually loaded overlays ($LOAD); memory-resident overlays; tasks that link to the library with memory-resident overlays; or for any combination of the previous conditions.

dddddd

The description of the segment type string which appears next to the segment name in the segment table.

The following sections describe the /List switch formats for the different kinds of task image files. Section 15.7 gives examples of the segment table lists.

15.1.1.1 **The /LI Switch and Regular Task Image Files** - For regular task image files (including those mapped to resident and supervisor mode libraries), the /LI switch displays the task's overlay segments in the order of their location in the file. Each segment in these files begins on an even block boundary.

15.1.1.2 **The /LI Switch and Multiuser Task Image Files** - For multiuser tasks, the /LI switch lists the starting disk block number and address boundaries of each segment. In addition, the address boundaries of the shared read-only segment are listed. The block number that is used to reference the multiuser segment is the same as the root segment. The multiuser segment is an extension of the root segment. The segment list disk block numbers have a corresponding entry in the Task Builder map.

See the <u>RSX-11M/M-PLUS Task Builder Manual</u> for more information on multiuser tasks.

15.1.1.3 **The /LI Switch and Resident Libraries** - For resident libraries, the /LI switch displays each of the task's segments as beginning on a new block boundary. However, the segments may not actually begin on even boundaries because of compression by the Task Builder. Resident libraries can be overlaid, but each overlay segment must also be resident in memory.

To avoid the possibility that two or more segments in a single block could have the same virtual address, ZAP treats the resident library in the same way that the Task Builder does. The Task Builder builds the library with each segment beginning on an even block boundary, but then compresses the segments in the task file itself. The Task Builder map is generated before the segments are compressed, so the boundaries given in the map do not necessarily correspond to the actual location of the segments.

The disk block boundaries given in the Task Builder map file are the ones that ZAP uses to address locations in the resident library and that the /LI switch displays in its segment table. They do not use the actual starting blocks of the segments. (You should be aware of this when you are working with resident libraries in absolute mode. However, also remember that you cannot use the /LI switch in absolute mode.)

You should also note that ZAP cannot know the physical starting addresses for the segments of an overlaid resident library because its overlay structure is stored in the symbol definition (.STB) file, not with the task image file itself. For ZAP, each segment's starting address is 000000.

See the <u>RSX-11M/M-PLUS Task Builder Manual</u> for more information on resident libraries.

15.1.1.4 **The /LI Switch and I- and D-Space Tasks** - For I- and D-space tasks, the /LI switch lists the starting block number and the address boundaries of each segment. The I-space or D-space segment may be suppressed in the listing when, for example, a segment with only I-space code does not include a listing for a D-space segment. If the segments are not suppressed, the segment list disk block numbers have a corresponding entry in the Task Builder map.

15.2 ADDRESSING LOCATIONS IN FILES

To address locations in a file, ZAP provides two addressing modes (task image and absolute, described above) and a set of internal registers, which includes eight Relocation Registers. This section first introduces the concept of relocation biases and the use of the Relocation Registers, then explains how to use the addressing modes.

15.2.1 Relocation Biases

When MACRO-11 generates a relocatable object module, the base address of each program section of the module is 000000. In the assembler 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 program sections 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.

15.2.2 ZAP Addressing Modes

As explained in Section 15.1, ZAP's two modes for addressing locations in a file are task image mode and absolute mode. Task image mode is the default mode for ZAP. The next two sections explain how to use these modes.

The following examples show excerpts from a MACRO-11 listing of the module MYFILE and a Task Builder map. These excerpts and the accompanying text 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 lines from a Task Builder map give 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 Task Builder map, 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 program code 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 following command line:

```
     2:120232+574/(RET)
```

ZAP responds by opening the location and displaying its contents:

```
     002:121026/ 032767
```

Section 15.4 describes the ZAP command line formats.

15.2.2.1 **Using the Task Image Addressing Mode** - In task image mode, ZAP allows you to address locations in a task image file by using the addresses the MACRO-11 assembler displays in its listing and the starting block number and byte offset listed in the Task Builder map. Unlike absolute mode, task image mode is useful for working with locations in an overlaid file because the Task Builder and ZAP perform the calculations necessary to relate the file's disk structure to its run-time memory structure.

15.2.2.2 **Using the Absolute Addressing Mode** - In absolute mode, ZAP processes the addresses you enter in the command lines as absolute byte addresses within the file. To use ZAP in absolute mode, invoke ZAP and enter the /AB switch with the file specification.

ZAP interprets the first address in the file you are changing as virtual block 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 data or task image file as well as the label and header blocks of a task image file on disk. However, to modify a disk task image, you must know the disk layout of the task image. Generally, absolute mode is practical only for data files or for task image files that are not overlaid.

15.3 INVOKING AND TERMINATING ZAP

You invoke ZAP using any of the methods described in Chapter 1. However, you cannot enter a file specification on the same line that you use to invoke ZAP unless the file is an indirect command file (see Section 15.3.1).

When ZAP prompts you, enter the file specification for the file you want to change. You enter the file specification in the format:

```
dev:[ufd]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 (explained in Section 15.6.1). This command exits you from ZAP and returns control to your command line interpreter.

15.3.1 Using Indirect Command Files with ZAP

An indirect command file contains the specification for the file you want to work with and the appropriate ZAP commands. You can specify the indirect command file in the same command line in which you invoke ZAP.

The following sample indirect command file (called CHANGE.CMD) contains ZAP commands. The commands will change the default priority of the despooler from 70 to 80 (120 octal). The V command (explained in Section 15.6.5) is used to verify that 70 (106 octal) is what is actually in the location to be changed. The command file has the following ZAP commands:

```
LPP.TSK/AB
0:346/
106V
120
X
```

To use the indirect command file (in this case, from MCR), type the following:

```
>ZAP @CHANGE
```

This command invokes ZAP, which then executes the commands in the file.

The commands being used first open the task image file (LPP.TSK) in absolute mode (/AB). The next two commands open the desired location (byte 346 in block 0) and verify its contents (106). The next command changes the contents to 120, which will be the new default priority for the despooler. The X command exits you from ZAP and returns control to MCR.

15.4 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 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 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 15.5 and 15.6 describe how to combine the command line elements to execute ZAP functions.

15.4.1 ZAP Commands

There are three types of ZAP commands:

- Open/close location commands
- General purpose commands
- RETURN Key

The following sections describe each type of command.

15.4.1.1 **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 15.4.2)
- Close the location after (optionally) modifying it and open another location as specified by the command

Section 15.5 describes the format for specifying open/close location commands.

15.4.1.2 **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 15.6 describes the format for specifying these commands.

15.4.1.3 **RETURN Key** - Unless there is another value or command on the line, the RETURN key closes the current location as modified and opens the next sequential location. ZAP commands take effect only after you press the RETURN key.

15.4.2 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 program section to change the contents of locations in the program section. You load the registers with the base address of the program section that has been relocated by the Task Builder.
- The Constant Register (C). You set this register to contain a 16-bit value, which you can specify as an expression in the ZAP command line.
- 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 byte 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 a register, specify a dollar sign ($) preceding the register name (in this case, C) in the command line. For example:

```
 _$C/ (RET)
 $C/ 000000
 _
```

This command line directs ZAP to display the contents of the Constant Register. The slash (/) is an open command described in Table 15-3.

15.4.3 ZAP Arithmetic Operators

Operators are single-character command line elements that define an arithmetic operation. Generally, ZAP evaluates these expressions as addresses. Table 15-1 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/(RET)
```

This method for calculating such a displacement is faster and more accurate than doing it manually.

Table 15-1
ZAP Arithmetic Operators

Operator	Function
+	The plus sign adds a value to another value. Used in an expression that ZAP then evaluates to be a command line element.
-	The minus sign subtracts a value from another value. Used in an expression that ZAP then evaluates to be a command line element.
*	The asterisk 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 15.4.4 explains the use of the colon and comma; the percent sign is an open command described in Table 15-3.

```
_0,40/ RET
002:0,000040/  000001
_1*33 RET
_/ RET
002:0,000040/  000103
_% RET
002:0,000040%  A$
```

In this example, the first command opens the locations that is 40 bytes offset from the location address contained in Relocation Register 0 and displays in octal format the contents of the new location. The location contains the value 000001. The second command converts 000001 to Radix-50 and adds 33 to the Radix-50 value. The slash (/) command again displays in octal the value contained in the offset location. The value is now 103. The percent sign (%) command displays 103 in Radix-50 format.

15.4.4 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. See Table 15-2.

Table 15-2
ZAP Command Line Element Separators

Separator	Function
,	The comma separates a Relocation Register specification from another command line element.
;	The semicolon separates an address from an internal register specification. Used in expressions that set values for Relocation Registers.
:	The colon separates a block number base value from a byte offset into the block. Used in most of the references to locations in a file.

15.4.5 ZAP Command Line Location-Specifier Formats

ZAP has three formats for specifying locations in a command line. Each provides a means of indexing into a file. The formats are:

- Current location symbol
- Byte offset
- Block number:byte offset
- Relocation register,byte offset

15.4.5.1 The Current Location Symbol

15.4.5.1 The Current Location Symbol - In command line expressions that ZAP evaluates as addresses, a period (.) represents the last open location.

15.4.5.2 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. This format is generally useful only when you are using absolute mode.

The following ZAP command line opens absolute location 664 and displays its contents in octal format:

```
_664/(RET)
```

15.4.5.3 Block Number:Byte Offset Format - This format allows you to specify a byte offset from a specific block in the file. Specify the format as follows:

```
blocknum:byteoffset
```

You can use this format for addressing locations whether or not you enter the /AB switch with the file specification.

In task image mode, ZAP allows you to enter the block number and byte offset displayed in the Task Builder map. The map gives information on the overlay segments in a task image file. See Section 15.2 for more information.

15.4.5.4 Relocation Register,Byte Offset Format - This format allows you to load a Relocation Register with the address of a location. The address 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 address loaded in the Relocation Register. For example:

```
_2:001254;3R RET
_3,64/ RET
002:3,000064/ 037334
```

The first command loads the address 001254 into Relocation Register 3, then the second command opens the location that is 64 bytes offset from block 2, location 001254. The contents of that location are 037334.

15.5 USING ZAP OPEN AND CLOSE COMMANDS

This section gives examples of how to use the ZAP open and close commands. These commands allow you to open locations in a file, modify those locations, and close the locations.

Table 15-3 summarizes the open and close commands.

Table 15-3
ZAP Open and 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 mark	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).

(continued on next page)

Table 15-3 (Cont.)
ZAP Open and Close Commands

Character	Designation	Function
RET	RETURN Key	Closes the current location as modified and opens the next sequential location if no other values or commands are on the command line. ZAP commands take effect only after you press the RETURN key.
^	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 and opens the new 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.

15.5.1 Opening Locations in a File

Use any of the ZAP open commands -- slash (/), quotation mark ("), percent sign (%), backslash (\), or apostrophe (') -- to open a location in a file. The format ZAP uses to display 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 continually press the RETURN key, consecutive locations are displayed in Radix-50 format until you change the format by entering a different special-character open command.

15.5.2 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 pressing the RETURN key. 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.

```
_/(RET)
002:120000/ 000000
_44444(RET)
_/(RET)
002:120000/ 44444
```

The first command (/) displays in octal format the contents (000000) of the current location. The contents are changed by entering the value 44444 and then the location is closed as modified by pressing the RETURN key. The slash (/) and RETURN key display the new contents of the location (last line of example).

15.5.3 Closing Locations in a File

ZAP uses the RETURN key and other special-character commands for closing a location in a file. The close commands perform three functions:

- Close the current location
- Direct ZAP to another location (such as the preceding location or a location referred to by the current location)
- Open the new location

The following sections give examples of how each command works.

15.5.3.1 Closing a Location and Opening 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 following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/(RET)
002:120100/ 000000
_(RET)
002:120102/ 000111
_(RET)
002:120104/ 000222
_(RET)
002:120106/ 000333
_^(RET)
002:120104/ 000222
```

The RETURN key closes the first three open locations and then opens the next location. The circumflex command closes location 120106 and directs ZAP to open the preceding location, 120104.

15.5.3.2 Closing a Location and Opening an Offset Location - 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 the new location. The following sequence of ZAP commands and responses shows how this command works:

```
    _2:120100/(RET)
    002:120100/ 000000
    _(RET)
    002:120102/ 111111
    _(RET)
    002:120104/ 222222
    _(RET)
    002:120106/ 000022
    __(RET)
    002:120132/ 123456
```

The RETURN key closes the first three open locations. The underscore command 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).

15.5.3.3 Closing a Location and Opening an Absolute Location - Use the at sign (@) command to close the current 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/(RET)
    002:120100/ 000000
    _(RET)
    002:120102/ 111111
    _(RET)
    002:120104/ 120114
    _@(RET)
    002:120114/ 114114
```

The RETURN key closes the first three open locations. The at sign command 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.

15.5.3.4 Closing a Location and Opening a Branch Target Location - 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 for the address of the next location, and then to open that location. The following sequence of ZAP commands and responses shows how this command works:

```
    _2:120100/(RET)
    002:120100/ 005000
    _(RET)
    002:120102/ 005301
    _(RET)
    002:120104/ 001020
    _>(RET)
    002:120146/ 052712
```

The RETURN key closes the first three open locations. The right angle bracket command closes location 120104, directs ZAP to use the low-order byte (020) of its contents as the branch offset for the address of the next location (120146), and then opens that location.

15.5.3.5 Closing a Location and Opening a Previous Location

- 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;OR RET
_0,10/ RET
002:0,000010/ 005212
   _RET
002:0,005224/ 001020
   _> RET
002:0,005266/ 000000
   _@ RET
002:0,000000/ 000000
   _< RET
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.

15.6 USING ZAP GENERAL PURPOSE COMMANDS

This section explains the functions of ZAP general purpose commands and shows the formats for specifying them. Table 15-4 describes the commands.

Table 15-4
ZAP General Purpose Commands

Command	Function
X	Exits from ZAP; returns control to your command line interpreter
K	Calculates the offset in bytes between an address and the value contained in a Relocation Register, 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

15.6.1 The X Command

Use the X command to exit from ZAP and then return control to your command line interpreter.

Specify the X command in the format:

```
     _X
```

15.6.2 The K Command

Use the K command to calculate the offset in bytes between an address and the value contained in a Relocation Register, to display the offset value, and to store it in the Quantity Register (Q).

You can enter the K command in the following formats:

K	Calculates the offset in bytes between the address of the currently open location and the value of the Relocation Register whose contents are equal to or closest to (but less than) the value of that address
nK	Calculates the offset in bytes between the currently open location and Relocation Register n
a;nK	Calculates the offset in bytes 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 calculated in the format:

```
     =reg,offset
```

The following example shows how to use the K command:

```
     _2:1172;0R(RET)
     _2:1232;1R(RET)
     _2:1202/(RET)
     002:000020/ 000111
     _K(RET)
     =0,000010
     _0,100;1K(RET)
     =1,000040
```

The first command sets the value of Relocation Register 0 to 001172. The second command sets the value of Relocation Register 1 to 001232. The slash command displays in octal format the contents of location 001202 (000111). The K command calculates the physical distance (offset) between the address of the currently open location (001202) and the value of the Relocation Register whose contents are equal to or closest to (but less than) the value of the address. ZAP then displays the number of the Relocation Register it used (0) and the offset (00010=001202-001172). The last command adds 100 to the address in Relocation Register 0 (001172) and then calculates the offset between the new address 0 (001272) and the contents of Relocation Register 1 (001232). ZAP then displays the number of the specified Relocation Register (1) and the offset (000040=001272-001232).

15.6.3 The O Command

Use the O command to display the jump and branch displacements from the current location to a 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. A branch displacement is the low-order byte of a branch instruction which, when executed, branches to the target location.

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/<RET>
0,4534/ 1234
_45660 <RET>
000030> 000014
_4534;45660 <RET>
000030> 000014
_
```

The first number (000030) is the jump displacement; the second number (000014) is the branch displacement.

15.6.4 The Equal Sign (=) Command

Use the equal sign command (=) to display (in octal) the value of the 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 (note that 177777 equals -1):

```
_2:30/<RET>
002:000030/ 000000
_.+177756=<RET>
000006
```

The first command displays in octal format the contents of location 000030, which are 000000. The next command adds 177756 to the address of the currently open location (000030). ZAP then displays the value of the specified expression (6=30+177756 or 6=30-22).

15.6.5 The V Command

Use the V command to verify that a location contains a specified value.

Specify the V command in the format:

```
contentsV
```

You use the V command 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 issues an error message 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 exits.

15.6.6 The R Command

Use the R command to specify the value for a Relocation Register. As explained in Sections 15.2 and 15.4.2, ZAP uses these registers to index into a program section so that you can change the contents of locations in the program section.

Specify the R command in the format:

```
_contents;nR
```

The variable n is the number of the Relocation Register (0 through 7).

For example:

```
_$3R/ RET
$3R/ 177777
_125670;3R RET
_$3R/ RET
$3R/ 125670
```

The first command accesses the contents of Relocation Register 3, which ZAP displays in octal format as specified by the slash. The contents of the register are 177777. The next command changes the contents of the register to 125670. The last command again displays the contents of the register, which have been changed correctly.

15.7 EXAMPLES

This section gives examples of ZAP usage. The examples show the /LI switch segment table format and how you would use some of the ZAP commands.

All of the ZAP examples in this section are based on information contained in the following excerpts from a sample Task Builder memory allocation map and from the program code for some of the modules in the task. Each example follows the section of program code associated with it.

Excerpts from Task Builder map:

```
MAINMEO.TSK;1   Memory allocation map  TKB M40.10      Page 1
                     14-MAR-83   16:01

Task    name   : ...MEO
Partition name : GEN
Identification : M00
Task  UIC      : [31,102]
Stack    limits: 000300 001277 001000 00512.
PRG xfr address: 020520
Task attributes: ID
Total address windows: 2.
Task  image  size  : 9184. words, I-Space
                     3520. words, D-Space
Task Address limits: 000000 043647 I-Space
                     000000 015507 D-Space
R-W disk blk limits: 000002 000102 000101 00065.

MAINMEO.TSK;1 Overlay description:

Base    Top        Length
----    ---        ------
000000  023135  023136  09822. I    MAINO
000000  014123  014124  06228. D

022140  043645  021506  09030. I    INPUT
014124  015507  001364  00756. D

022140  022307  000150  00104. I    CALC
014124  014167  000044  00036. D

022310  022437  000130  00088. I    AADD
014170  014173  000004  00004. D

022310  022437  000130  00088. I    SUBB
014170  014173  000004  00004. D

022310  022437  000130  00088. I    MULL
014170  014173  000004  00004. D

022310  022441  000132  00090. I    DIVV
014170  014173  000004  00004. D

022140  023725  001566  00886. I    OUTPUT
014124  014251  000126  00086. D

MAINMEO.TSK;1   Memory allocation map  TKB M40.10      Page 2
MAINO                14-MAR-83   16:01

*** Root segment: MAINO

R/W mem  limits: 000000 023135 023136 09822. I-Space
                 000000 014123 014124 06228. D-Space

Disk blk limits: 000002 000024 000023 00019. I-Space
                 000025 000041 000015 00013. D-Space
```

```
Memory allocation synopsis:

Section                                             Title  Ident  File
-------                                             -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)      000300 000216 00142.
                               000300 000216 00142. CBTA   04.3   SYSLIB.OLB;7
                                 .
                                 .
                                 .


MAINMEO.TSK;1   Memory allocation map  TKB M40.10       Page 4
INPUT                   14-MAR-83   16:01

*** Segment: INPUT

R/W mem  limits: 022140 043645 021506 09030. I-Space
                 014124 015507 001364 00756. D-Space

Disk blk limits: 000042 000063 000022 00018. I-Space
                 000064 000065 000002 00002. D-Space
                   .
                   .
                   .
                                                        Page 5
*** Segment: CALC

R/W mem  limits: 022140 022307 000150 00104.  I-Space
                 014124 014167 000044 00036.  D-Space

Disk blk limits: 000066 000066 000001 00001.  I-Space
                 000067 000067 000001 00001.  D-Space
                   .
                   .
                   .
                                                        Page 7
*** Segment: AADD

R/W mem  limits: 022310 022437 000130 00088.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000070 000070 000001 00001.  I-Space
                 000071 000071 000001 00001.  D-Space
                   .
                   .
                   .
                                                        Page 8
*** Segment: SUBB

R/W mem  limits: 022310 022437 000130 00088.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000072 000072 000001 00001.  I-Space
                 000073 000073 000001 00001.  D-Space
                   .
                   .
                   .
```

```
                                                        Page 9
*** Segment: MULL

R/W mem  limits: 022310 022437 000130 00088.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000074 000074 000001 00001.  I-Space
                 000075 000075 000001 00001.  D-Space
                   .
                   .
                   .

MAINMEO.TSK;1   Memory allocation map  TKB M40.10      Page 10
DIVV                 14-MAR-83   16:01

*** Segment: DIVV

R/W mem  limits: 022310 022441 000132 00090.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000076 000076 000001 00001.  I-Space
                 000077 000077 000001 00001.  D-Space
                   .
                   .
                   .
                                                        Page 11
*** Segment: OUTPUT

R/W mem  limits: 022140 023725 001566 00886.  I-Space
                 014124 014251 000126 00086.  D-Space

Disk blk limits: 000100 000101 000002 00002.  I-Space
                 000102 000102 000001 00001.  D-Space

Memory allocation synopsis:

Section                                        Title  Ident  File
-------                                        -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)    022140 000374 00252.
                             022140 000042 00034. SAVAL  00     SYSLIB.OLB;7
                             022202 000074 00060. CATB   03     SYSLIB.OLB;7
                             022276 000126 00086. CDDMG  00     SYSLIB.OLB;7
                             022424 000110 00072. C5TA   02     SYSLIB.OLB;7
```

Example 1:

In this example, the segment table for task MAINMEO is requested. Note that the segment table corresponds exactly to the overlay description list given in the Task Builder map. The sequence of ZAP commands is as follows:

```
>ZAP(RET)
ZAP>MAINMEO/LI(RET)
ZAP Version V02.01 COPYRIGHT (c) DIGITAL EQUIPMENT CORPORATION 1983
Segment Table
000002: 000000-022137 MAINO  I-space root
000025: 000000-014123 MAINO  D-space root
000042: 022140-043647 INPUT    I- and
000064: 014124-015507 INPUT    D-space
000066: 022140-022307 CALC     I- and
000067: 014124-014167 CALC     D-space
000070: 022310-022347 AADD       I- and
000071: 014170-014173 AADD      D-space
000072: 022310-022437 SUBB      I- and
000073: 014170-014173 SUBB      D-space
000074: 022310-022437 MULL      I- and
000075: 014170-014173 MULL      D-space
000076: 022310-022443 DIVV      I- and
000077: 014170-014173 DIVV      D-space
000100: 022140-023727 OUTPUT  I- and
000102: 014124-014253 OUTPUT  D-space
```

In Example 1, the first command line invokes ZAP and the second command line requests the segment table for the task MAINMEO. The /List switch directs ZAP to give the starting disk block for the root segment of the task (in this example, MAINO) and for each segment overlaid on the root of the task. The /List switch also lists the base and top addresses, plus the segment text string for each segment.

Because this is an I- and D-space overlaid task, there is an I-space root segment and a D-space root segment and each is a part of the root segment of the task MAINO. In this example, the I-space root segment begins at disk block 2. The addresses for the I-space root segment range from 000000 to 022137. The next line of numbers is for the D-space root segment which begins at disk block 25. The addresses for the D-space root segment range from 000000 to 014123. The next line of numbers is for the segment INPUT. That I-space segment begins at disk block 42 and the D-space segment begins at disk block 64. The I-space addresses range from 022140 to 043647 and the D-space addresses range from 014124 to 015507. The table continues for the remaining overlaid segments in the task MAINMEO.

Example 2:

In this example, the contents of a location in another task module (TEST.MAC) are being changed. The following excerpt from the module shows the associated code.

```
TEST - TEST MACRO FILE     MACRO M1113  18-MAR-81 07:48  PAGE 8-1

1269                                  ; THIS IS A PART OF THE MODULE
1270                                  ;           TEST.MAC
1271                                  ; WHICH IS IN THE ROOT SEGMENT:
1272                                  ;            TEST
1273                                  ;
1274 010132  010146                            MOV    R1,-(SP)
1275 010134  012704  041114                    MOV    #"LB,R4
1276 010140  005003                            CLR    R3
1277 010142                                    CALL   $FNDUB
1278 010146  010146                            MOV    R1,-(SP)
1279 010150  012704  050123                    MOV    #"SP,R4
```

The first command line invokes ZAP and the second command line requests the segment table for TST. The /List switch gives the starting disk block for the root task and for each task overlaid to the root task. The switch also lists the base and top addresses for each of the tasks.

In this example, TEST (the root task) begins at disk block 42 and its addresses range from 120000 to 151513. The next line of numbers is for the task module TSTC1.MAC. That task begins at disk block 74 and its addresses range from 151514 to 153133. The table continues for the remaining tasks.

Example 2:

In this example, the contents of a location in another task module (TEST.MAC) are being changed. The following excerpt from the module shows the associated code.

```
TEST - TEST MACRO FILE     MACRO M1113  18-MAR-81 07:48  PAGE 8-1

                                   ; THIS IS A PART OF THE MODULE
                                   ;          TEST.MAC
                                   ; WHICH IS IN THE ROOT SEGMENT:
                                   ;           TEST
                                   ;
     010132  010146                         MOV    R1,-(SP)
     010134  012704  041114                 MOV    #"LB,R4
     010140  005003                         CLR    R3
     010142                                 CALL   $FNDUB
     010146  010146                         MOV    R1,-(SP)
     010150  012704  050123                 MOV    #"SP,R4
```

The sequence of ZAP commands is:

```
_42:121244;OR (RET)
_0,10136" (RET)
042:131402" LB
_' (RET)
042:131402' L
_' (RET)
042:131403' B
_120 (RET)
_0,10136" (RET)
042:131402" LP
```

The first command line loads the starting address of TEST.MAC (121244 in disk block 42) into Relocation Register 0. The second command line displays as an ASCII word the contents of location 10136 of the module. The contents are LB. The first apostrophe command (') displays the first byte of the word (L) and the second command displays the second byte (B). The following command line changes the contents of the second byte to 120, which is the ASCII code for the letter P. The last command displays the new contents of location 10136, which are now LP.

Example 3:

In this example, the contents of a location are also being changed. This time, the location is in the module TSTVB1.MAC. The following excerpt is the associated code.

```
TSTVB1 - TSTVB1 MACRO FILE  MACRO M1113  18-MAR-81 07:52  PAGE 4-2

153                                  ;  PART OF MODULE TSTVB1.MAC
154                                  ; WHICH IS ALSO IN THE ROOT SEGMENT:
155                                  ;           TEST
156                                  ;
157  000334  005702                            TST    R2
158  000336  001404                            BEQ    70$
159  000340  052767  000060  172516            BIS    #60,SR3
160  000346  000403                            BR     75$
```

The sequence of ZAP commands is:

```
    _42:133470;1R<RET>
    ¯1,342/<RET>
    ¯042:134032/ 000060
    _100<RET>
    ¯1,342/<RET>
    ¯042:134032/ 000100
```

The first command line loads the starting address of TSTVB1.MAC (133470 in disk block 42) into Relocation Register 1. The second command line displays in octal the contents of location 342 in the module. The third command line changes the contents of this location from 60 to 100. The last command line displays the new contents (again in octal).

Example 4:

In this example, the operation code (op code) for one of the instructions in another module is being changed. The module is TSTCM.MAC, and the following excerpt is the associated code.

```
    TSTCM - TSTCM MACRO FILE    MACRO M1113  18-MAR-81 07:47  PAGE 3-7

    402                               ; PART OF THE MODULE TSTCM.MAC
    403                               ; WHICH IS IN THE SEGMENT: TSTCM
    404                               ;
    405  001272  073127  177766             ASHC     #-10.,R1
    406  001276  010037  00000G             MOV      R0,@#KISAR6
    407  001302  062701  140002             ADD      #140000+2,R1
```

The sequence of ZAP commands is:

```
    _113:154530;R2 (RET)
    ¯2,1302/ (RET)
    ¯113:156032/ 062701
    _162701 (RET)
    ¯2,1302/ (RET)
    ¯113:156032/ 162701
    _X
```

The first command line loads the starting address of TSTCM.MAC (154530 in disk block 113) into Relocation Register 2. The second command line displays in octal the current instruction contained in location 1302. The instruction includes the op code 06 for the ADD operation. The third command line changes the op code to 16, which signifies the SUBTRACT operation. The fourth command line displays the new contents of the location and the X command ends the ZAP session.

15.8 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 line, specifying 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: Reenter the command line, using the correct file specification.

ZAP -- ERROR ON COMMAND INPUT

Explanation: An I/O error occurred while a command line was being read. This could be a hardware error.

User Action: Ensure that the hardware is functioning properly. If it is, reenter the command line. If not, call your DIGITAL 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, reenter the command line. If not, call your DIGITAL Field Service representative.

ZAP -- NO OPEN LOCATION

Explanation: You attempted to modify the contents of 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 line, specifying the correct value.

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 line, specifying a valid Relocation Register number.

ZAP -- NO SUCH SEGMENT

Explanation: The starting disk block was not the start of any segment in the task image file on disk.

User Action: Reenter the command line, specifying the correct disk block address.

ZAP-- NOT A TASK IMAGE OR NO TASK HEADER

Explanation: An error occurred while the segment tables were being constructed. Possibly, the file is not a task image, the /AB switch was not specified, or the task image is defective.

User Action: Terminate the ZAP session, then try invoking ZAP with the /AB switch 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 specification for errors; ensure that the volume is properly mounted; or use PIP to check your file access privileges (see Chapter 3).

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 with a larger address space. (See the description of the INSTALL command in the RSX-11M/M-PLUS MCR Operations Manual or the RSX-11M/M-PLUS Command Language Manual.)

ZAP -- TOO MANY ARGUMENTS

Explanation: You entered more arguments on the command line than are allowed.

User Action: Reenter the command line, specifying 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, then reenter the command line, specifying the correct syntax.

ZAP -- VERIFY FAILURE

Explanation: The V command 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 reenter the command line. Ensure that you are correcting the right file or file version.

APPENDIX A

COMMANDS AND SWITCHES

A.1 INTRODUCTION

This appendix presents a summary of the commands and switches used by the RSX-11M/M-PLUS utilities described in this manual. Each section number of the appendix corresponds to the chapter discussing that utility. For example, Chapter 3 and Section A.3 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 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 to the line preceding the top line 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 the block buffer or input file.

[n]Change /string1/string2[/]

Replaces string1 with string2 n times in the current line

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. If filespec is given, the output file is renamed.

CLOSES

Closes the secondary 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

Closes files and terminates the editing session.

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. [-n] is valid in block mode only.

DP [n] or [-n]

Same as DELETE, except that the new current line is printed.

End

Same as the BOTTOM command.

ERASE [n]

Erases the current line or the block buffer and the next n-1 blocks.

ESC

Prints the previous line, makes it the new current line, and exits from input mode (block mode only). <ESCAPE> is the same as NP-1.

EXit [filespec]

Closes files, renames the output file, and terminates the editing session.

EDx [filespec]

Transfers remaining lines in block buffer and input file to output file, and closes the file. If filespec is given, the output file is renamed. The command then deletes the input file and terminates the editing session.

FF

Inserts a form feed into the block buffer (used to delimit a page).

FIle filespec

Transfers lines from the input file to both the output file and 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 not specified, EDI enters input mode.

KILL

Closes the input and output files, and deletes the output file.

[n]LC /string1/string2[/]

Changes all occurrences of string1 in current line and n-1 lines to string2.

LIst

Prints on the user terminal all lines remaining in the block buffer or input file, starting with the 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 for string or, if n is specified, the nth occurrence of string. In block mode, the search stops at end of the current block.

MACRO x definition

Defines macro number x to be definition. The value x may be 1, 2, or 3.

MCall

Retrieves macro definitions stored in the file MCALL;n.

[n]Mx [a]

Executes macro x for n times, passing it the numeric argument a.

[n]<definition> (MACRO IMMEDIATE)

Allows you to define and execute a macro n times in one step. The macro is stored as macro number 1.

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 specified secondary input file.

OUtput [ON] or [OFF]

Continues or discontinues writing to output file (line-by-line mode).

Overlay [n]

Deletes the current line and the next n-1 lines, enters input mode, and inserts new lines as typed in place of original lines.

PAGe [n]

Enters block mode, if not already in block mode, and reads page n into the block buffer.

[n]PFind (string)

Searches successive block buffers until the nth occurrence of string is found. The string must begin in column 1.

[n]PLocate (string)

Searches successive block buffers for the nth occurrence of string. The string can occur anywhere in the line.

PAste /string1/string2[/]

Same as the LINE CHANGE (LC) 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. The last line printed becomes the current line.

REAd [n]

Reads the next n blocks of text into the block buffer.

RENew [n]

Writes the current block to the output file and reads new block n from the input file (block mode only).

RET

Prints the next line, makes it the current line, and exits from input mode. Pressing the RETURN key is the same as NP+1.

Retype string

Replaces the current line with the text of string. If string is not specified, the line is deleted.

SAve [n] [filespec]

Saves the current line and the next n-1 lines in the file specified by filespec. If filespec is not given, the lines are saved in SAVE.TMP.

SC /string1/string2[/]

Locates string1 and replaces it with string2.

SP

Reestablishes the primary file as the input file.

SS

Selects the secondary file that will be the input file.

SIZE n

Specifies the maximum number of lines to be read into the block buffer.

TAb [ON] or OFF

Turns automatic tabbing on or off.

Top

Same as BEGIN command.

TOF

In block mode, returns to the top of the input file and saves all pages edited. TOF reads in a new block after writing the previous block to the output file.

TYpe [n]

Prints next n lines. Same as PRINT command in line-by-line mode. In block mode, the current line pointer does not change unless EOB is reached.

UNSave [filespec]

Inserts all lines from the specified file following the current line. If filespec is not given, SAVE.TMP is used.

UC [ON] or OFF

Turns uppercase conversion of lowercase characters on or off.

Verify [ON] or OFF

Specifies whether locator and change commands are verified.

Write

Writes the contents of the current block buffer to the output file and then erases the block buffer.

A.3 PIP COMMAND SUMMARY

outfile=infile1[,infile2...,infilen]/AP[/FO]

Opens an existing file (outfile) and appends the input file(s) to the end of it. (/FO is the File Owner subswitch.)

outfile/BS:n=infile1[...,infilen]

Defines the block size for magnetic tapes.

outfile=infile1[,infile2...,infilen][[/ME][/subswitch(es)]]

Creates a copy of a file on the same or another device. The /ME (Merge) switch creates a new file from two or more existing files. Subswitches are one or more of the following:

/BL:n[.]	Block allocated
/CO	Contiguous output
/-CO	Output can be noncontiguous
/FO	File Owner
/NV	New Version
/SU	Supersede

outfile/CD=infile1[,infile2...,infilen]

outfile=infile1/CD[,infile2...,infilen]

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) and/or with an output magnetic tape device.

/DD:startdate:enddate

Restricts file searches to files created during a specified period of time.

infile1[,infile2...,infilen]/DE[/LD]

Deletes files from a UFD. (/LD is the List Deleted files subswitch.) See the sections on Delete (3.2.2.5) and Purge (3.2.2.17) for a complete description of the List Deleted files subswitch.

ddnn:[ufd]/DF or ddnn:/DF or [ufd]/DF or /DF

Changes the default device and/or UFD for the current PIP task.

outfile=infile1[,infile2...,infilen]/EN[/NV]

Enters a synonym for a file in a directory or directories on the same device 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 subswitch.)

infile1/EOF[:block:byte][...,infilen/EOF[:block:byte]]

Specifies the end-of-file pointers for a file.

filespec/EX

Excludes all the files specified with the file specification during a file search.

```
outfile=/FI:filenum:seqnum
```

Accesses an existing file by its file identification number.

```
[ddnn:]/FR
```

Displays the amount of available space on a specified volume, the largest contiguous space on that volume, the number of available headers, and the number of headers used. If ddnn: is not specified, it defaults to SY0:.

```
/ID
```

Identifies the version of PIP being used and if PIP is linked to ANSFCS.

```
[listfile=]infile1[...,infilen]/LI[subswitch]
```

Lists one or more files contained in a UFD 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 subswitches:

```
/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, or /UN (Unlock files).

```
infile/PR[/SY[:RWED]][/OW[:RWED]][/GR[:RWED]][/WO[:RWED]][/FO]
```

Sets the protection status for a file. The switches are:

```
/SY -- system access rights
/OW -- owner access rights
/GR -- group access rights
/WO -- world access rights
/FO -- File Owner subswitch
```

The privileges are:

```
R -- Read
W -- Write
E -- Extend
D -- Delete
```

You can specify any of the letters in any order.

```
infile1[,infile2...,infilen]/PU[:n][/LD]
```

Deletes a specified range of obsolete versions of a file. (The switch never deletes the latest version of a file.) (/LD is the List Deleted files subswitch.)

outfile=infile1[,infile2...,infilen]/RE[/NV]

Changes the name of a file with an option to force the version number of outfile to be one greater than the latest version for the file. (/NV is the New Version subswitch.)

infile1[,infile2...,infilen]/RM

Removes an entry from a UFD, but does not delete the file associated with that entry.

outfile/RW=infile or outfile=infile/RW

Rewinds a magnetic tape. When you specify /RW with the output file specification, PIP begins writing the file at the beginning of the tape. When you specify /RW with the input file specification, PIP rewinds the tape before searching for the input file.

outdsk:outfile/SB=inmag:infile

The file copied to disk from magnetic tape may have records crossing block boundaries. (/SB is the default.) If you specify /-SB, the records will not cross block boundaries.

infile1[,infile2...,infilen]/SD

Prompts for user response before deleting files.

infile1[,infile2...,infilen]/SP[:n]

Specifies a list of files to be printed on a line printer (n is the number of copies).

outfile=infile/SR

Allows shared reading of a file that has already been opened for writing.

/TD

Restricts file searches to files created on the current day.

infile1[,infile2...,infilen]/TR

Truncates files to their logical end-of-file point.

outfile/UF[/FO]=infile1[,infile2...,infilen]

Creates a User File Directory entry in the Master File Directory of the volume specified with outfile.

infile1[,infile2...,infilen]/UN

Unlocks a file that was locked as a result of being improperly closed.

outfile=infile1[,infile2...,infilen]/UP[/FO]

Opens an existing file (outfile) and writes new data (infile) into it from the beginning. (/FO is the File Owner subswitch.)

A.4 FLX COMMAND SUMMARY

The FLX commands generally have the form:

```
outspec/sw=infile1[,infile2...,infilen]/switch
```

/switch can be one or more of the following:

/BL:n[.]

Specifies the number of contiguous blocks (n) in octal or decimal to be allocated to the output file.

/BS:n

Specifies the block size (n) for cassette tape output.

/CO

Specifies that the output file is to be contiguous.

/DE

Deletes files from a DOS-11 or RT-11 (used with the /RT switch) volume.

/DI

Causes a directory listing of a cassette or DOS-11 volume or, when used with the /RT switch, of an RT-11 volume. The directory is placed in the specified output file.

/DNS:n

Specifies a density of 800, 1600, or 6250 bpi for a magnetic tape volume.

/DO

Identifies the volume as a DOS-11 formatted volume.

/FA:n

Specifies formatted ASCII transfer mode file format.

/FB:n

Specifies formatted binary transfer mode file format.

/FC

Specifies that FORTRAN carriage control conventions are to be used.

/ID

Displays 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 (n) in octal or decimal to allocate when initializing an RT-11 disk or DECtape.

/RS

Identifies the volume as a Files-11 formatted volume.

/RT

Identifies the volume as an RT-11 formatted volume.

/RW and /-RW

Specifies whether the magnetic tape will rewind before FLX begins the file transfer.

/SP

Specifies that the converted file is to be spooled by the print spooler or the Queue Manager.

/UI

Specifies that the output file is to have the same UFD as the input file.

/VE

Verifies each record written to a cassette.

/ZE[:n.]

Initializes cassettes or DOS-11 volumes or, when used with the /RT switch, RT-11 volumes. Initializing erases any files already on the volume.

A.5 FMT COMMAND SUMMARY

The general format of an FMT command line is:

```
FMT ddnn:[/switch1.../switchn]
```

/switch can be one or more of the following:

/BAD

Runs the Bad BLock Locator Utility if it is installed. Note that this switch can only be used with operating systems that allow spawning of tasks. RSX-11M provides spawned tasks as a system generation option. RSX-11M-PLUS always includes support for spawned tasks.

/DENSITY or /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.

/MANUAL or /MAN

Enters manual operating mode and formats the sector or track you specify.

/NOVERIFY or /NOVE or /-VERIFY or /-VE

Inhibits the default verification of a successful FMT operation.

/OVR

Overrides or ignores the manufacturer's bad block sector file (MDBSF).

/VERIFY or /VE

Verifies that an FMT operation was successfully completed. This switch is the default.

/WLT

Rewrites the MDBSF (on the last track of the device) to add bad sectors found during an FMT operation.

/@Y

Informs FMT that it is receiving input from an indirect command file. User intervention is not allowed during the operation.

A.6 BAD COMMAND SUMMARY

The format for executing BAD is:

```
ddnn:[/swl.../swn]
```

/sw can be one or more of the following:

/ALO:volume label

Prompts you for blocks to be allocated to BADBLK.SYS and to be entered in the bad block descriptor file.

/CSR=nnnnnn

Specifies the CSR address of a device that is not in a standard location (stand-alone version of BAD only).

/LI

Lists bad blocks as they are located.

/MAN

Allows you to enter bad blocks, which are then included in the bad block descriptor file.

/NOWCHK

Negates the effect of /WCHK (see below).

/OVR

Creates the bad block descriptor file on a last-track device.

/PAT=m:n

Specifies the double-word data pattern used to locate bad blocks.

/RETRY

Recovers soft errors.

/UPD

Reads the bad block descriptor file and prompts for user entries.

/VEC=nnn

Specifies the interrupt vector address of a device that is not in a standard location (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). The switch is not valid for DT:-, DX:-, or DY:-type devices.

A.7 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 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.

```
/BAD[:MANUAL   ]
    [:AUTOMATIC]
    [:OVERRIDE ]
```

Enters the locations of bad blocks on volumes. The default is /BAD:AUTOMATIC.

/BUFFERS:number

Specifies the default 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:{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 before or after a specified date and/or time.

/DENSITY:number

Specifies the data density at which BRU writes to tape. The default is /DENSITY:6250.

/DIRECTORY

Displays information (such as backup set names, file names, or volume number of a tape) for 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 during a restore operation 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 a file is extended 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 the output disk before proceeding with the operation.

/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 as determined by the number of file headers in the volume's index file.

/MOUNTED

Allows you to back up files from a disk that is mounted (with the MCR or DCL MOUNT commands) as a Files-11 volume.

/NEW_VERSION

Directs BRU to resolve conflicts resulting from files with identical file specifications by creating a new version of the file.

/NOINITIALIZE

Specifies that you do not want to initialize the output disk because it is already in Files-11 format.

/NOPRESERVE

Specifies that you do not want 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 not transferring the file on the input volume. Therefore, the file on the output volume is not superseded. (The default is /NOSUPERSEDE.)

/OUTVOLUME:name

Specifies the volume label of the output disk. The label can be up to 12(10) characters long.

```
/POSITION:(BEGINNING    )
          (MIDDLE       )
          (END          )
          (BLOCK:number )
```

Specifies the location of the index file on the output disk volume.

```
/PROTECTION:(SYSTEM:value)
            (OWNER:value )
            (GROUP:value )
            (WORLD:value )
```

Specifies the default protection status for all files created on the output volume being initialized.

```
/REVISED:(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 before or after a specified date and/or time.

/REWIND

Rewinds the first tape of a tape set before performing the operation.

/SUPERSEDE

Resolves file specification conflicts by deleting the old file on the output volume and replacing it with the file from the input volume. (The default is /NOSUPERSEDE.)

/TAPE_LABEL:label

Specifies a 6-character ANSI volume identifier for identifying the tape volume.

/UFD

Directs BRU to create UFDs (if they do not already exist) on a mounted output volume, then copy into them the files from the same UFDs on the input volume.

/VERIFY

Copies data from the input volume to the output volume, compares the volumes, and reports any differences.

/WINDOWS:number

Specifies for the output disk the default number of mapping pointers allocated for file windows. The default number is the same as for the input disk.

A.8 DSC COMMAND SUMMARY

The format for executing DSC is:

```
outdev[s]:[label][/switch]=indev[s]:[label][/switch]
```

/switch can be one or more of the following:

/AP

Appends a DSC file to the first volume of a magnetic tape set that already contains a DSC file.

$$/BAD=\left\{\begin{array}{l}\text{MAN}\\ \text{NOAUTO}\\ \text{MAN:NOAUTO}\\ \text{OVR}\\ \text{MAN:OVR}\end{array}\right\}$$

Allows manual entry of bad block locations; can supplement, override, or ignore the disk's own bad block file.

/BL=n or /BL:n

Sets the number of 256-word blocks DSC can include in each of its two buffers.

/CMP

Compares input and output volumes for differences.

/CSR=xxxx

Specifies control status addresses for a specific Status Control Block (SCB). /CSR is valid only with the stand-alone version of DSC.

/DENS=1600 or /DENS:1600
/DENS=800:1600 or /DENS:800:1600
/DENS=6250

Overrides the DSC default storage density for magnetic tapes of 800 bpi. The first form of the switch creates magnetic tapes at 1600 bpi density. The second form (the split density switch) creates magnetic tapes with volume header information at 800 bpi and the rest of the tape at 1600 bpi. The third form creates magnetic tapes at 6250 bpi. The /DENS=6250 option is valid with TU78 tapes only.

/RW

Rewinds all volumes in a magnetic tape set before execution of the current command line.

/TM02=x

Specifies the physical unit number of the formatter on the RH11/RH70 controller (stand-alone version of DSC only).

/UNIT=x

Specifies the physical unit that will be referenced by the indicated Unit Control Block (UCB). The /UNIT switch is valid only with the stand-alone version of DSC.

/VE

Copies data from the input volume and compares it with the output volume following the data transfer.

/VEC=xxx

Specifies the vector address for a specific Status Control Block (SCB). The /VEC switch is valid only with the stand-alone version of DSC.

A.9 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,scratchdev=indev/DV
    or
indev/DV
```

Validates directories against the files they list.

```
listfile=indev/FR
    or
indev/FR
```

Displays the amount of available space on a volume.

```
listfile,scratchdev=indev/HD
    or
indev/HD
```

Recognizes all bad file headers on a volume. The /AL subswitch allows all bad headers to be automatically deleted with no user intervention.

```
listfile,scratchdev=indev/LI
    or
indev/LI
```

Lists the index file by file identification.

```
listfile,scratchdev=indev/LO
    or
indev/LO
```

Scans the file structure looking for files that are not in any directory and cannot be referenced by file name.

```
listfile=indev/RC[:n]
     or
indev/RC[:n]
```

Ensures that every block of every file on the specified volume can be read. The optional parameter [:n] indicates how many blocks are to be read at a time.

```
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.10 LBR COMMAND SUMMARY

```
outfile/CO:size:ept:mnt:=infile
```

Compresses a library file by physically deleting logically deleted records, putting the free space at the end of the file, and making the free space available for new library module inserts.

```
outfile/CR:size:ept:mnt:libtype=infiletype
```

Allocates a contiguous library file on a direct access device (for example, a disk).

```
outfile/DE:module1[:module2...:modulen]
```

Logically deletes library modules and their associated entry points from a file.

```
outfile/DF:type...
     or
/DF:type
```

Specifies the default library file type.

```
outfile/DG:global1[:global2:...:globaln]
```

Deletes the specified library module entry points from the entry point table.

```
outfile[/EP]=infile[...,infilen]
    or
outfile=infile[/EP][...,infilen[/EP]]
```

Includes or excludes entries in the entry point table.

```
outfile=infile/EX[:modulename1:...:modulenamen]
```

Reads (extracts) one or more modules from a library and writes them into the specified output file.

```
outfile/IN=infile1[,infile2...,infilen]
    or
outfile=infile/IN:name:op:op:op:op   (universal)
```

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 their corresponding entry points.
/FU	Lists all modules in the library file and provides a full module description that includes the size, date of insertion, and module-dependent information.

```
outfile/MH:module:op:op:op:op
```

Modifies the optional user-specified information in the module header of a universal library.

```
outfile/RP=infile1[,infile2...,infilen] (global format)
    or
outfile=infile1/RP[,infile2[/RP]...,infilen[/RP]] (local format)
    or
outfile/RP:name:op:op:op:op=infile1[,infile2...,infilen](universal/global format)
    or
outfile=infile1/RP:name:op:op:op:op[,infile2...,infilen](universal/local format)
```

Replaces or, in certain cases, inserts library modules in a library file.

```
outfile,listfile/SP
```

Spools the listing file for printing. This is the default setting; use /-SP to prevent the file from being printed.

```
outfile=infile1/SS[,infile2[/SS]...,infilen[/SS]]
```

Sets the selective search attribute bit in the object module header.

```
outfile/SZ=infile1[,infile2...,infilen] (global format)
     or
outfile=infile1/SZ[,infile2[/SZ]...,infilen[/SZ]] (local format)
```

Reduces the size of macro definitions by removing comments, blank lines, and trailing blanks and tabs from the macro text.

A.11 DMP COMMAND SUMMARY

The format for executing DMP is:

```
[outfile][/switch(es)][=inspec][/switch(es)]
```

/switch can be one of the following:

/AS

Specifies that data be dumped one byte at a time in ASCII mode.

/BA:n:m

Specifies a two-word base block address.

/BL:n:m

Specifies the first and last logical blocks to be dumped.

/BY

Specifies that data be dumped in octal byte format.

/DC

Specifies that data be dumped in decimal word format.

/DENS:n

Specifies density of an input magnetic tape when DMP is in device mode only. Values for n can be 800, 1600, or 6250.

/FI:file-number:sequence-number

Specifies the input file with its file-ID instead of its name (File Mode only).

/HD[:F or :U]

Includes the file header in the data dumped. "F", the default, specifies a formatted Files-11 dump for the header. "U" specifies an unformatted octal dump.

/HF

Specifies the format for data blocks that have the Files-11 header structure. Other blocks are dumped as unformatted octal.

/HX

Specifies that data be dumped in hexadecimal byte format.

/ID

Causes the current version of DMP to be displayed or printed.

/LB

Requests the starting (logical) block number and a contiguous or noncontiguous indication for the file to be displayed.

/LC

Specifies that the data should be dumped in lowercase characters. This switch is valid only if the output device supports lowercase characters.

/LW

Specifies that data be dumped in hexadecimal double-word format.

/MD[:n]

Controls line number sequencing during a memory image dump.

/OCT

Specifies that the data should be dumped in octal format. If no DMP format switches are included, the default is octal format.

/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. The /RW switch can be used at any time to reposition a tape at beginning-of-tape (BOT).

/R5

Dumps in Radix-50 word format.

/SB:n or /SB:-n

Specifies the number of blocks DMP spaces forward (n) or backwards (-n) on a tape.

/SF:n or /SF:-n

Specifies the number of end-of-file (EOF) marks DMP spaces forward (n) or backward (-n) on a tape.

/SP

Spools the dump file (the output file) to the line printer.

/WD

Specifies that data be dumped in hexadecimal word format.

A.12 CMP COMMAND SUMMARY

The CMP command line takes the following form:

```
[outfile [/sw...]=]infile1,infile2
```

/sw can be any one of the following:

/BL or /-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 switch.

/CB or /-CB

Specifies that CMP list infile2 with change bars, in the form of exclamation marks (!), to denote 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 after the deleted lines is marked.

/CO or /-CO

Specifies that CMP include comments (that is, text preceded by a semicolon) in compare processing. /CO is the default switch.

/DI or /-DI

Specifies that CMP list the differences between the two files (rather than marking the lines in infile2). /DI is the default switch.

/CB and /DI are mutually exclusive switches. If both are specified, /CB overrides /DI.

/FF or /-FF

Specifies that CMP include records consisting of a single form-feed character in compare processing. /-FF is the default 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 (/LI:3).

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 or /-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 or /-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] or /-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 switch.

This switch applies only if you have the print spooler task (RSX-11M) or the Queue Manager (RSX-11M/M-PLUS) installed.

/TB or /-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 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). /VB:041 (for the exclamation mark) is the default switch.

A.13 SLP COMMAND SUMMARY

The format of a SLP command line is:

```
outfile[/switch,listfile/[-]SP]=infile[/switch]
```

/switch can be any one of the following:

/AU or /-AU

Enables or disables the audit trail, which indicates the changes made during the most recent editing session.

/BF or /-BF

Enables or disables blank fill (right-justification) for an audit trail.

/CM[:n]

Deletes the audit trail and any trailing spaces or tabs, and truncates the text at the specified horizontal position.

/CS[:n]

Calculates the checksum value for the edit commands.

/DB or /-DB

Enables or disables double-spaced listings. /-DB is the default switch.

/NS

Does not sequence lines in the output file. New lines are indicated by the audit trail (if specified). This switch overrides the /RS and /SQ switches.

/RS

Resequences the lines in the output file so that the line numbers are incremented for each line written to the output file.

/SP or /-SP

Enables or disables the spooling of listing files to a line printer. This switch applies only if the print spooler task (RSX-11M) or the Queue Manager (RSX-11M/M-PLUS) is installed.

/SQ

Sequences the lines in the output file so that the numbers reflect the line numbers of the original input file.

/TR

Specifies that a diagnostic error message occur when lines are truncated by the audit trail.

A.14 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 calculate the checksum for all the binary data of a specified module.

A.15 ZAP COMMAND AND SWITCH SUMMARY

The ZAP command line is in the form:

```
ddnn:[ufd]filename.filetype;version[/sw...]
```

A.15.1 ZAP Open/Close Commands

/ (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 mark)

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).

RET (RETURN key)

Closes the current location as modified and opens the next sequential location if no other values or commands are on the command line. ZAP commands take effect only after you press the RETURN key.

^ 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, and opens the new 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.

A.15.2 ZAP General-Purpose Commands

X

Exits from ZAP, returns control to the CLI.

K

Calculates the offset in bytes between an address and the value contained in a Relocation Register, 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.

A.15.3 ZAP Switches

/AB

Specifies absolute mode.

/LI

Displays the overlay segment table for an overlaid task image file.

/RO

Specifies read-only mode.

APPENDIX B

THE CROSS-REFERENCE PROCESSOR (CRF)

The Cross-Reference Processor (CRF) is an independent task that produces cross-reference listings for the MACRO-11 and Task Builder tasks. CRF is invoked by the /CR switch in 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 in the MACRO-11 command line, refer to the PDP-11 MACRO-11 Language Reference Manual. For information on how to invoke CRF with the Task Builder command line, refer to the RSX-11M/M-PLUS Task Builder Manual.

B.1 HOW CRF PROCESSES DATA

When the /CR switch is specified in 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 B-1 is an overview showing the processing steps performed in each phase, along with input and output files used.

B.1.1 MACRO-11 or Task Builder Processing

The first steps to produce cross-reference listings are performed by the MACRO-11 Assembler or the Task Builder and the /CR switch. These tasks generate three files during execution: an object file (filename.OBJ) or task image file (filename.TSK), a listing file (filename.LST or filename.MAP), and a CRF symbol table file (filename.CRF).

The object file or task image file is directed to an appropriate device and does not affect CRF processing.

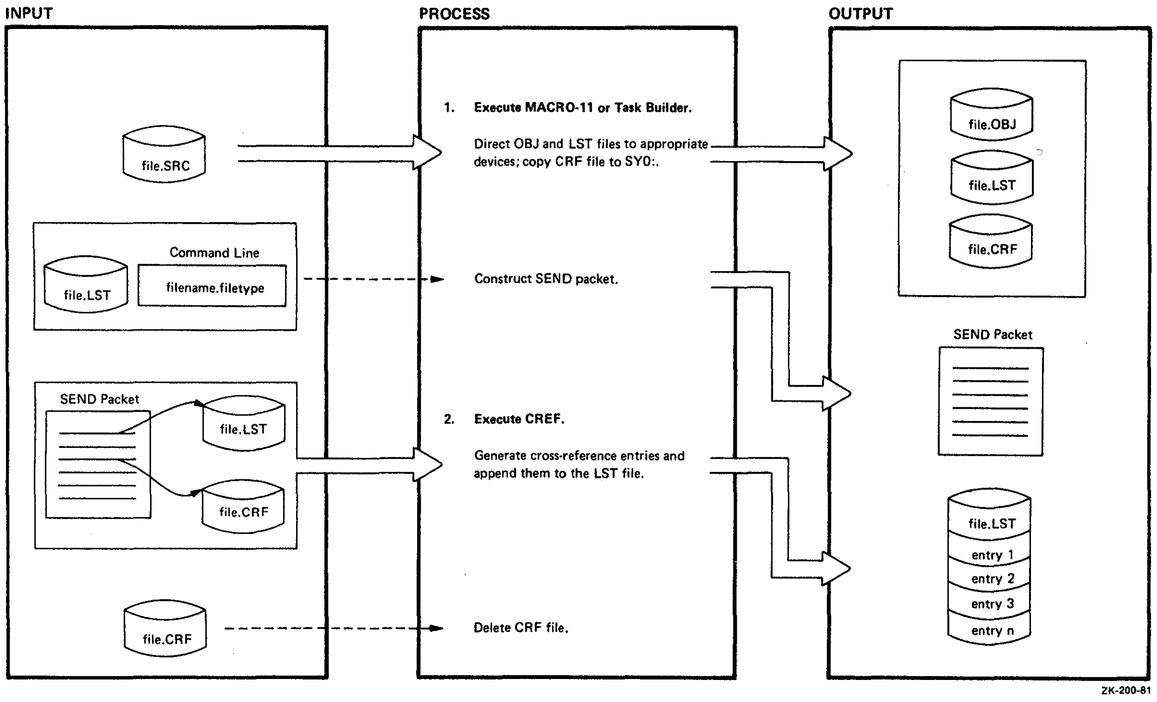


Figure B-1 How MACRO-11, the Task Builder, and CRF Generate Cross-Reference Listings

The listing file consists of text information generated during the execution of the tasks, for example, the listing of the source code in a MACRO-11 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 sends CRF a SEND packet, using information in the listing file and the file name and file type 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 the Task Builder.

The formats of the symbol table file and the SEND packet are shown in Figures B-2 and B-3.

If spooling is requested with the /SP switch 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.

B.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 specify the symbol table file for processing (filename.CRF):

```
Symbol table filename - Text file name
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 filename.CRF.

B.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 B-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 2-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 flags 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 that 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.

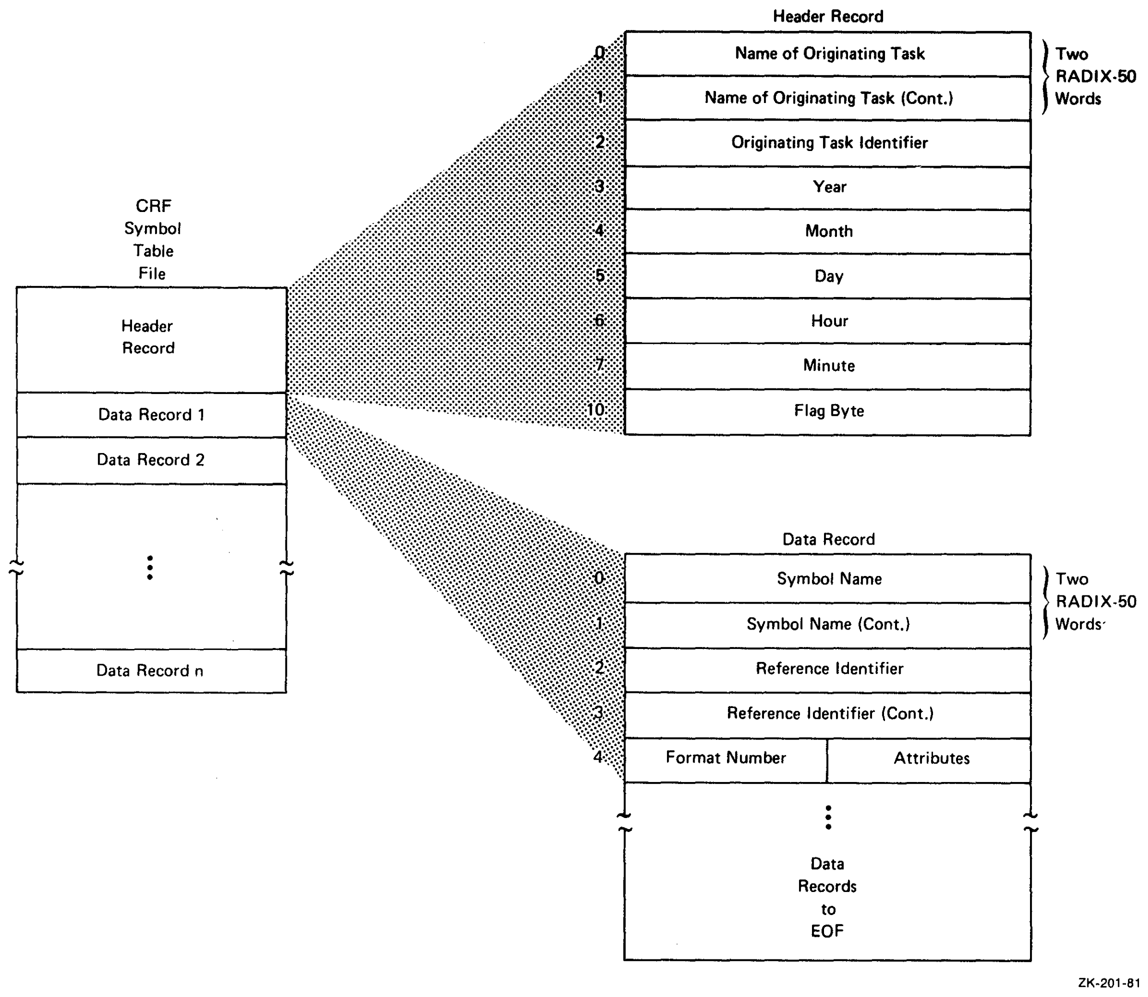


Figure B-2 Format of the CRF Symbol Table File

B.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 17 words, as described in Figure B-3.

Words 0 and 1 of the SEND packet contain the name of the sending task. Words 2 through 4 contain the file name of the output listing file in Radix-50 format. Word 5 contains the file type 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 off line 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.

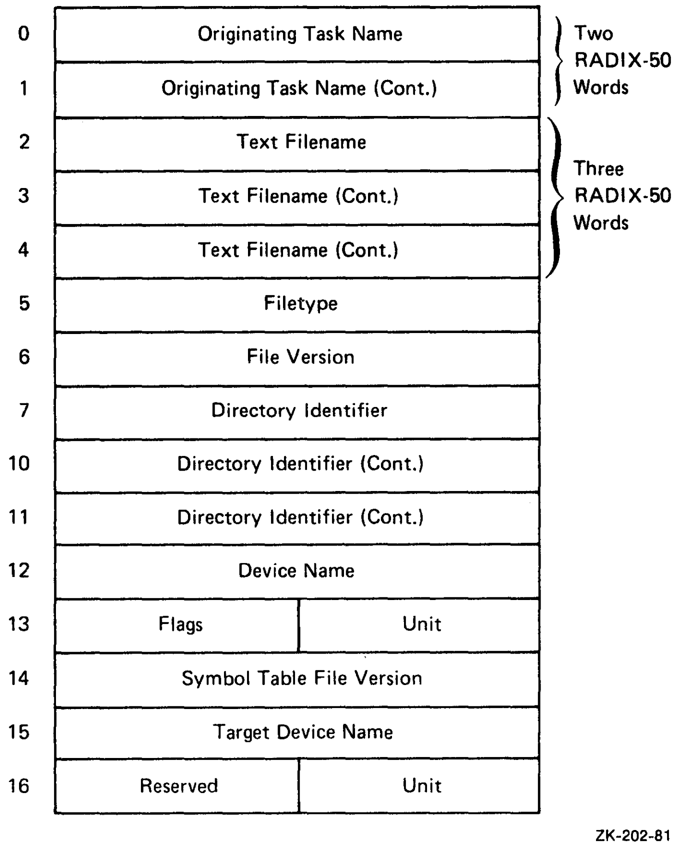


Figure B-3 Format of the CRF SEND Packet

B.4 CRF ERROR MESSAGES

The following error messages are output by CRF. 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 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: The file 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 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.

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 CRF was 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-11 or the Task Builder to produce the cross-reference output.

INVALID OUTPUT FORMAT SPECIFIED

Explanation: This error indicates an inconsistency in the data file submitted for CRF processing.

User Action: Submit a Software Performance Report with the related console dialogue and any other pertinent information.

NO DYNAMIC STORAGE AVAILABLE

Explanation: The CRF 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 CRF 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 indicates a CRF malfunction.

User Action: Submit a Software Performance Report with the related console dialogue and any other pertinent information.

UNABLE TO OPEN WORKFILE

Explanation: Possible causes are:

- The workfile device is not mounted.
- The workfile device is 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: CRF encountered an I/O error while reading or writing data to its workfile. Possible causes are:

- Device is 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.

INDEX

RSX-11M/M-PLUS Utilities Manual
AA-FD13A-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ______

Name ______ Date ______

Organization ______

Street ______

City ______ State ______ Zip Code or Country ______

- - — Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

— — Do Not Tear - Fold Here —