RSX-11M-PLUS
Utilities Manual

Order No. AA-JS15A-TC

RSX-11M-PLUS Version 4.0

Digital Equipment Corporation Maynard, Massachusetts

First printing, December 1979
Revised, November 1981
Updated, April 1983
Revised, July 1985
Revised, August 1987

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

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

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

Copyright ©1979, 1981, 1983, 1985, 1987 by Digital Equipment Corporation

All Rights Reserved.
Printed in U.S.A.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

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

ZK3087

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*	CANADA	INTERNATIONAL
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital's local subsidiary or approved distributor

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

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

In Canada call 800-267-6215.

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

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a trademark of the American Mathematical Society.

Contents

Chapter 2 Bad Block Locator Utility (BAD)

Chapter 3 Backup and Restore Utility (BRU)

Chapter 4 File Compare Utility (CMP)

Chapter 5 File Dump Utility (DMP)

Chapter 6 Disk Save and Compress Utility (DSC)

Chapter 7 Line Text Editor (EDI)

Chapter 8 File Transfer Program (FLX)

Chapter 9 Disk Volume Formatter (FMT)

Chapter 10 Librarian Utility Program (LBR)

Chapter 11 Object Module Patch Utility (PAT)

Chapter 12 Peripheral Interchange Program (PIP)

Chapter 13 Source Language Input Program (SLP)

Chapter 14 File Structure Verification Utility (VFY)

Chapter 15 Task/File Patch Program (ZAP)

Appendix A Command Formats

Appendix B The Cross-Reference Processor (CRF)

Appendix C Data Terminal Emulator (DTE) and the Micro/RSX File Transfer Utility (MFT)

Index

Examples

Figures

Tables

Preface

Manual Objectives

The *RSX-11M-PLUS Utilities Manual* is a reference manual describing the use of 15 utilities supported on the RSX-11M-PLUS operating system.

Intended Audience

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

Structure of This Document

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

Chapter 2 describes the Bad Block Locator Utility (BAD).

Chapter 3 describes the Backup and Restore Utility (BRU).

Chapter 4 describes the File Compare Utility (CMP).

Chapter 5 describes the File Dump Utility (DMP).

Chapter 6 describes the Disk Save and Compress Utility Program (DSC).

Chapter 7 describes the Line Text Editor (EDI).

Chapter 8 describes the File Transfer Utility Program (FLX).

Chapter 9 describes the Disk Volume Formatter Utility (FMT).

Chapter 10 describes the Librarian Utility Program (LBR).

Chapter 11 describes the Object Module Patch Utility (PAT).

Chapter 12 describes the Peripheral Interchange Program (PIP).

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

Chapter 14 describes the File Structure Verification Utility (VFY).

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).

Appendix C describes the Data Terminal Emulator (DTE) and the Micro/RSX File Transfer Utility (MFT).

Associated Manuals

The *RSX-11M-PLUS MCR Operations Manual* describes the Monitor Console Routine (MCR) and its commands. The utilities are usually invoked from MCR. This manual uses MCR to specify the formats and examples of the utilities.

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

Conventions Used in This Manual

The following conventions are used in this manual:

Convention	Meaning
>	A right angle bracket is the default prompt for the Monitor Console Routine (MCR), which is one of the command interfaces used on RSX-11M-PLUS systems. All systems include MCR.
$	A dollar sign followed by a space is the default prompt of the DIGITAL Command Language (DCL), which is one of the command interfaces used on RSX-11M-PLUS and Micro/RSX systems. Many systems include DCL.
MCR>	This is the explicit prompt of the Monitor Console Routine (MCR).
DCL>	This is the explicit prompt of the DIGITAL Command Language (DCL).
xxx>	Three characters followed by a right angle bracket indicate the explicit prompt for a task, utility, or program on the system.
UPPERCASE	Uppercase letters in a command line indicate letters that must be entered as they are shown. For example, utility switches must always be entered as they are shown in format specifications.
command abbreviations	Where short forms of commands are allowed, the shortest form acceptable is represented by uppercase letters. The following example shows the minimum abbreviation allowed for the DCL command DIRECTORY: `$ DIR`

Convention	Meaning
lowercase	Any command in lowercase must be substituted for. Usually the lowercase word identifies the kind of substitution expected, such as a filespec, which indicates that you should fill in a file specification. For example: `filename.type;version` This command indicates the values that comprise a file specification; values are substituted for each of these variables as appropriate.
/keyword, /qualifier, or /switch	A command element preceded by a slash (/) is an MCR keyword; a DCL qualifier; or a task, utility, or program switch. Keywords, qualifiers, and switches alter the action of the command they follow.
parameter	Required command fields are generally called parameters. The most common parameters are file specifications.
[option]	Square brackets indicate optional entries in a command line or a file specification. If the brackets include syntactical elements, such as periods (.) or slashes (/), those elements are required for the field. If the field appears in lowercase, you are to substitute a valid command element if you include the field. Note that when an option is entered, the brackets are not included in the command line.
[,...]	Square brackets around a comma and an ellipsis mark indicate that you can use a series of optional elements separated by commas. For example, (argument[,...]) means that you can specify a series of optional arguments by enclosing the arguments in parentheses and by separating them with commas.
{ }	Braces indicate a choice of required options. You are to choose from one of the options listed.
:argument	Some parameters and qualifiers can be altered by the inclusion of arguments preceded by a colon. An argument can be either numerical (COPIES:3) or alphabetical (NAME:QIX).
( )	Parentheses are used to enclose more than one argument in a command line. For example: `SET PROT = (S:RWED,O:RWED)`
,	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.

Convention	Meaning
[g,m] [directory]	The convention [g,m] signifies a User Identification Code (UIC). The g is a group number and the m is a member number. The UIC identifies a user and is used mainly for controlling access to files and privileged system functions. This may also signify a User File Directory (UFD), commonly called a directory. A directory is the location of files. Other notations for directories are: [ggg,mmm], [gggmmm], [ufd], [name], and [directory]. The convention [directory] signifies a directory. Most directories have 1- to 9-character names, but some are in the same [g,m] form as the UIC. Where a UIC, UFD, or directory is required, only one set of brackets is shown (for example, [g,m]). Where the UIC, UFD, or directory is optional, two sets of brackets are shown (for example, [[g,m]]).
filespec	A full file specification includes device, directory, file name, file type, and version number, as shown in the following example: `DL2:[46,63]INDIRECT.TXT;3` Full file specifications are rarely needed. If you do not provide a version number, the highest numbered version is used. If you do not provide a directory, the default directory is used. Some system functions default to particular file types. Many commands accept a wildcard character (*) in place of the file name, file type, or version number. Some commands accept a filespec with a DECnet node name.
.	A period in a file specification separates the file name and file type. When the file type is not specified, the period may be omitted from the file specification.
;	A semicolon in a file specification separates the file type from the file version. If the version is not specified, the semicolon may be omitted from the file specification.
@	The at sign invokes an indirect command file. The at sign immediately precedes the file specification for the indirect command file, as follows: `@filename[.type;version]`

Convention	Meaning
...	A horizontal ellipsis indicates the following: • Additional, optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis shows where elements of command input or statements in an example or figure have been omitted because they are irrelevant to the point being discussed.
KEYNAME	This typeface denotes one of the keys on the terminal keyboard. For example, the RETURN key.
"print" and "type"	The term "print" refers to any output sent to a terminal by the system. The term "type" refers to any user input from a terminal.
black ink	In examples, what the system prints or displays is printed in black.
red ink	In interactive examples, what the user types is printed in red. System responses appear in black.
[xxx]	A symbol with a 1- to 3-character abbreviation, such as [x] or [RET], indicates that you press a key on the terminal. For example, [RET] indicates the RETURN key, [LF] indicates the LINE FEED key, and [DEL] indicates the DELETE key.
[CTRL/a]	The symbol [CTRL/a] means that you are to press the key marked CTRL while pressing another key. Thus, [CTRL/Z] indicates that you are to press the CTRL key and the Z key together in this fashion. [CTRL/Z] is echoed on some terminals as ^Z. However, not all control characters echo.

Summary of Technical Changes

The following sections list features, commands, qualifiers, error messages, and restrictions that are new to the utilities or have been modified for the RSX–11M–PLUS Version 4.0 operating systems. These new or modified features are documented in this revision of the *RSX–11M–PLUS Utilities Manual*

Also, major changes to the organization of the manual are included at the end of this summary.

New Hardware Support

RSX–11M–PLUS Version 4.0 supports the following new hardware:

- MicroPDP–11/53 processor
- DHQ11 16-line multiplexer
- CXA16/CXB16 16-line multiplexer
- CXY08 8-line multiplexer
- DELUA UNIBUS synchronous communications controller
- RX33 diskette drive
- RA82, RD31, RD32, RD53, and RD54 disk drives
- TU81 and TU81E 9-track magnetic tape drives
- TK70 and TK50 cartridge tape drives
- LA75 printer
- DFA01 modem

For FLX, the new RT–11 devices are as follows:

- RX33 and RX50 diskettes
- RD50 and RD51 disks
- RC25 and RCF25 fixed/removable disks

New or Modified Features

BRU has the following new or modified features:

- BRU detects and automatically skips over a bootable system image when appending to or restoring from a magnetic tape.
- Access to a device allocated and mounted by another user is no longer allowed.
- A method of executing indirect command files containing BRU commands is described.
- The defaults for wildcard (*) file specifications have been changed.

For FLX, file type CDA has been added to the list of default file types for the Image Mode switch (/IM).

For FMT, DL-type devices can be used with the /WLT and /VE switches.

LBR has the following new and modified features:

- A description of the user file attributes is included.
- LBR now recognizes lowercase and uppercase characters in macro directives.

New and Modified Qualifiers and Switches

BRU has the following new and modified qualifiers:

- The /IDENTIFICATION qualifier displays the version of BRU in use.
- The /LENGTH qualifier should not be used with cartridge tape devices such as the TK25 and TK50.
- The description of the /NOSUPERSEDE qualifier has been expanded.
- The description of the /VERIFY qualifier has been changed.

The new DMP switch /LIM:n:m specifies the range of bytes n to m of each record or block to be dumped.

New or Modified Error Messages

BRU produces the following new and modified error messages:

- The following new error messages are displayed when the mount status of the input and/or output devices is inconsistent with the qualifiers specified:

```
BRU -- *FATAL* -- Device not mounted foreign on ddnn

BRU -- *FATAL* -- Device not mounted files 11 on ddnn
```

- The following new message is displayed if an append operation is attempted on a tape other than the first:

```
BRU -- *FATAL* -- Cannot append on continuation volume
```

Restrictions

BRU has the following restrictions:

- The /NEW_VERSION and /VERIFY qualifiers are mutually exclusive. They cannot be specified together in the same command line.
- You must specify /REWIND/APPEND when placing the first backup set on a magnetic tape that contains a bootable system image
- The /TAPE_LABEL qualifier is ignored during a restore operation if there is a bootable system image at the beginning of the tape
- Do not run BRU and BAD simultaneously when using standalone BRU (BRUSYS).

Changes to the Document

The following changes in organization and additions to the document have occurred:

- The chapters on the utilities have been alphabetized according to 3-character utility names. Chapter 1 is an introductory chapter.
- Appendix C, which describes the Data Terminal Emulator (DTE) and the Micro/RSX File Transfer Utility (MFT), has been added.

Chapter 1
Introduction

The RSX–11M–PLUS operating system provides 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 they also allow you to work with different kinds of media (such as disks, magnetic tapes, and cassettes). The RSX–11M–PLUS utility programs are listed and described briefly in the next section; reference information for each utility is presented as a separate chapter of this manual. Three appendixes are also included to provide you with a summary of commands and switches for the utilities; to describe the Cross-Reference Processor (CRF), which is used with the MACRO–11 assembler (MAC) and the Task Builder (TKB); and to describe the Data Terminal Emulator (DTE) and the Micro/RSX File Transfer Utility (MFT).

In addition to summarizing the RSX–11M–PLUS utilities, this introduction provides the following information:

- Describes how to enter RSX–11M–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–PLUS Utility Programs

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

- Bad Block Locator Utility (BAD)
- Backup and Restore Utility (BRU)
- File Compare Utility (CMP)
- File Dump Utility (DMP)
- Disk Save and Compress Utility (DSC)
- Line Text Editor (EDI)
- File Transfer Utility Program (FLX)
- Disk Volume Formatter Utility (FMT)

- Librarian Utility Program (LBR)
- Object Module Patch Utility (PAT)
- Peripheral Interchange Program (PIP)
- Source Language Input Program (SLP)
- File Structure Verification Utility (VFY)
- 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-PLUS operating systems that are used as or considered to be utilities. The Task Builder (TKB), Crash Dump Analyzer (CDA), and MACRO-11 assembler (MAC) are examples of other utility-like programs. Some programs, such as the DIGITAL standard editor (EDT), are common across different operating systems. These other programs are documented elsewhere in the RSX-11M-PLUS documentation set. Refer to the *RSX-11M-PLUS Information Directory and Master Index* for information on what programs are available and where they are described.

1.1.1 Bad Block Locator Utility

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.

1.1.2 Backup and Restore Utility

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 in most areas. Also, BRU compresses data, the volumes do not have to be initialized, and incremental backups are possible. BRU can be run either on line or stand alone.

1.1.3 File Compare Utility

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

1.1.4 File Dump Utility

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.

1.1.5 Disk Save and Compress Utility

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.

1.1.6 Line Text Editor

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

1.1.7 File Transfer Utility Program

FLX is a file transfer and format conversion program that transfers files between DOS–11, RT–11, and Files–11 volumes, with some restrictions.

1.1.8 Disk Volume Formatter Utility

FMT formats and verifies several types of disks. FMT writes and verifies sector headers, sets the density for flexible disks, and allows spawning of BAD.

1.1.9 Librarian Utility Program

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

1.1.10 Object Module Patch Utility

PAT updates, or patches, relocatable binary object modules.

1.1.11 Peripheral Interchange Program

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

1.1.12 Source Language Input Program

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

1.1.13 File Structure Verification Utility

VFY is a disk verification program that verifies the consistency and validity of the file structure on a Files–11 volume.

1.1.14 Task/File Patch Program

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–PLUS utilities is shown next.

Format

outfile[,...]=infile[,...]

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.)

Note

This general format varies from utility to utility. Some utilities 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 shown next.

Format

ddnn:[directory]filename.type;version[/switch[...]/subswitch[...]]

Parameters

ddnn

Specifies the physical or logical device unit containing the desired volume. The name consists of two American Standard Code for Information Interchange (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, SY0.

directory

Specifies the directory containing the desired file or files. The directory can have two possible formats. The named directory format has one to nine of the following characters: the 26 letters A to Z and the numbers zero to nine (for example, [USER1]). The numbered directory (sometimes referred to as the User Identification Code (UIC) or the User File Directory (UFD)) format has two numbers separated by a comma (for example, [g,m]). 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 default is the current directory to which your terminal is set.

See the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual* for more information on directories.

filename

Specifies the name of the file. File names can be from one to nine characters in length. There is no default.

type

Specifies 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.

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

version

Specifies a decimal (or octal if this option was chosen during system generation) 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 revised 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 to 32767_{10}. However, some utilities allow the use of 0 to specify the highest numbered version and -1 to specify the lowest numbered version (0, or the latest version, is the default). If a file has a version number of 32767_{10}, 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

Specifies 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 the following three forms:

Switch Format	Meaning
/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 number 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 number sign (#).

The following are examples of valid switch specifications:

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

```
/-SW
```

```
/NOSW:-#50:SWITCH
```

subswitch

Specifies 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
```

In the previous command line, /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-PLUS MCR Operations Manual.* For more information on DCL, see the *RSX-11M-PLUS Command Language Manual.*

To determine whether you are using MCR or DCL or another CLI, press 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. Section 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 prompt is as follows:

```
>
```

If you press CTRL/C first, the MCR prompt is as follows:

```
MCR>
```

The DCL prompt is shown next.

```
$
```

If you press CTRL/C first, the DCL prompt is as follows:

```
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, you should use the general format for specifying files shown next.

Format

command[/qualifier[...]] infile outfile

For more information on DCL commands and their formats, refer to the *RSX-11M-PLUS Command Language Manual.*

DCL users can also use all MCR command forms by using the DCL command MCR (or MC) as follows:

```
$ MCR utilityname RET
utilityname>commandline RET
utilityname>
```

or

```
$ MCR utilityname commandline RET
$
```

1.4.1 Invoking Installed Utilities

The RSX-11M-PLUS operating system provided in distribution kits does not have any utilities installed. Once the system has been generated, the system manager usually installs any commonly used utilities. Use the MCR command TAS or DCL command SHOW TASKS /INSTALL 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 command INSTALL. 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 Your CLI

You can use the following command line to invoke a utility, to execute a function, and to return control directly to MCR:

```
>utilityname commandline RET
>
```

You can use the following command line to invoke a utility, to execute a function, and to return control directly to DCL:

```
$ MCR utilityname commandline RET
$
```

Using this method to invoke the utility allows you to enter a single command for execution. The command is executed and control returns to your CLI. (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 CLI.)

Two exceptions to this command format are the SLP and ZAP utilities. Unless you are using indirect command files, you must first invoke these utilities and then enter the command lines as described in Section 1.4.1.2. See Section 1.5 for more information on indirect command files.

1.4.1.2 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 [RET]
utilityname>commandline  [RET]
utilityname>
```

For DCL:

```
$ MCR utilityname [RET]
utilityname>commandline [RET]
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.

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

Example

```
>PIP [RET]
PIP> test.dat;5/DE [RET]
PIP>
```

1.4.2 Invoking Uninstalled Utilities

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

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

```
>RUN $utilityname
```

or

```
$ RUN $utilityname
```

The dollar sign ($) directs MCR or DCL to search the system directory for the utility and to bring it into storage. 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 following 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, press 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, as follows:

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 press 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 (MAC), or the Task Builder (TKB)). 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]
```

For DCL:

```
>RUN $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.

The RSX–11M–PLUS operating system also allows you to use indirect command files that contain MCR or DCL commands. The command lines must be legal for the CLI. You invoke the indirect command file by entering only the file specification preceded by the at sign (@) character in response to the prompt (in this case, the MCR prompt) as follows:

```
>@indirectcommandfile
```

The default values for elements of an indirect command file specification are as follows:

Element	Default Value
ddnn	SY0
directory	The current directory
filename	No default; the file name must be specified
type	CMD
version	The latest version of the file

For complete information on how to use indirect command files, see the *RSX-11M-PLUS Indirect Command Processor Manual.*

Chapter 2
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 Monitor Console Routine (MCR) command INI, which allocates the bad blocks to the bad block 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. You can use BAD in its task version, which runs at the same time as other tasks, or in its standalone version included in [6,54]BRUSYS.SYS, which runs by itself on the computer. The standalone version is required if you have a system with a single disk drive.

BAD can also be invoked through the Digital Command Language (DCL) command ANALYZE /MEDIA. See the *RSX-11M-PLUS Command Language Manual* for more information.

2.1 BAD Command Line

You can invoke BAD by any of the methods described in Chapter 1.

The command line for BAD is shown next.

Format

ddnn:[/switch[...]]

Parameters

ddnn

Specifies a physical device.

switch

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.

Table 2–1 contains a reference list of BAD switches along with a brief description of each. For a detailed description of BAD switches see Section 2.2.

Table 2–1: BAD Switches

Switch	Format	Function
Allocate	/ALO:volumelabel	Prompts you for blocks to be allocated to BADBLK.SYS and entered in the bad block descriptor file.
List	/LI	Lists bad blocks as they are located.
Manual	/MAN	Prompts you for additional bad blocks, which are entered in the bad block descriptor file.
Override	/OVR	Creates the bad block descriptor file on a last-track device.
Pattern	/PAT=m:n	Specifies the doubleword data pattern used to locate bad blocks.
Retry	/RETRY	Recovers soft errors.
Update	/UPD	Reads the bad block descriptor file and prompts for input.

2.2 BAD Switch Descriptions

The following sections describe the switches you can use with BAD commands.

2.2.1 Allocate

The Allocate switch (/ALO:volumelabel) prompts you for additional bad blocks that are added to the bad block descriptor file and allocated to the bad block file [0,0] BADBLK.SYS.

Format

/ALO:volumelabel

Parameter

volumelabel

Specifies the volume.

The /ALO:volumelabel switch eliminates the need to reinitialize the disk after updating the bad block descriptor file. This switch does not cause BAD to perform pattern checks.

Note

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

If you enter bad blocks by using the /ALO:volumelabel switch, BAD will prompt you as follows:

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

You may then enter bad blocks in the following 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 number sign (#), which indicates an octal value.

Examples

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

Enters the block numbers 70, 71, and 72 in the bad block descriptor file. The blocks are allocated to the bad block file [0,0]BADBLK.SYS.

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

Specifies a single bad block and enters block 3 in the bad block descriptor file. Block 3 is allocated to the bad block file [0,0]BADBLK.SYS.

```
BAD>LBN(S)= 100:2,3,  200:100  45:1 [RET]
```

Enters blocks 100, 101, 3, 200 to 299, and 45 in the bad block descriptor file. These blocks are allocated to the bad block file [0,0] BADBLK.SYS. You can separate bad block sequences with a space, tab, or comma.

```
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) =
```

Lists all the Logical Block Numbers (LBNs) allocated to BADBLK.SYS before listing the LBNs in the bad block descriptor file when a carriage return is entered in response to the prompt.

LBNs 100, 101, 003, and 200 are allocated to the bad block file [0,0] BADBLK.SYS.

The previous command line also allocates 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) that is contained in the bad block file but is not yet allocated to the bad block file [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 it lists entries at your terminal as part of existing bad block sequences.

When you have finished supplying information for the /ALO:volumelabel switch, press the ESCAPE key, the ALTMODE key, or CTRL/Z in response to the prompt. The bad block descriptor file will then be rewritten with the new bad block information. Blocks that you enter manually and that BAD decides are reliable are included in the bad block descriptor file.

2.2.2 List

The List switch (/LI) prints all bad blocks 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. The List switch (/LI) is valid for all devices.

2.2.3 Manual

The Manual switch (/MAN) first prompts you for bad block information and then performs data pattern checking. Any block that you enter by using the Manual switch (/MAN) is included in the bad block descriptor file or the Software-Detected Bad Sector File (SDBSF).

If you enter bad blocks by using the /MAN switch, BAD will prompt you as follows:

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

You may then enter bad blocks in the following 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 number sign (#), which indicates an octal value.

Examples

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

Enters the block numbers 70, 71, and 72 in the bad block descriptor file.

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

Specifies a single bad block and enters block 3 in the bad block descriptor file.

```
BAD>LBN(S)= 100:2,3,  200:100  45:1 [RET]
```

Uses both of the previous forms on the same command line.

This command enters blocks 100, 101, 3, 200 to 299, and 45 in the bad block descriptor file. You can separate bad block sequences with a space, tab, or comma.

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 it lists entries at your terminal as part of existing bad block sequences.

When you have finished supplying information for the /MAN switch, press the ESCAPE key, the ALTMODE key, or CTRL/Z in response to the prompt. Pattern checking will then start. Blocks that you enter manually and that BAD decides are reliable are included in the bad block descriptor file.

2.2.4 Override

The Override switch (/OVR) ignores last track information and writes a bad block descriptor file on the last good block before the last track. In other words, the /OVR 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 /OVR switch before using the MCR command INI. The /OVR 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-PLUS MCR Operations Manual* for a description of the MCR command INI.

2.2.5 Pattern

The Pattern switch (/PAT=m:n) locates bad blocks by means of a user-specified doubleword data pattern.

Format

/PAT=m:n

Parameter

m:n

Represents the two 16-bit octal numbers used as the doubleword data pattern. Decimal numbers may be specified by placing a period (.) after each number.

2.2.6 Retry

The Retry switch (/RETRY) attempts 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.

The /RETRY switch should be specified when using BAD with DU-type devices.

2.2.7 Update

The Update switch (/UPD) reads the bad block descriptor file immediately and prompts 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 bad block file [0,0]BADBLK.SYS to be updated.

Example

```
BAD>LBN(s)= RET
000100:002
000003:001
000200:100
000045:001
BAD>LBN(S)=
```

Lists all the sequences in the bad block descriptor file when a carriage return is entered in response to the prompt.

The first number in the display represents the beginning block of the sequence. The second number represents the number of bad blocks. Bad block numbers are listed in decimal.

When a bad block sequence is entered, BAD determines if these bad blocks are adjacent to an already existing sequence. If you are using a non-last-track device, BAD appends your bad block entry to the existing sequence. If you are using a last-track device, BAD records individual bad blocks in core memory, but it lists entries at your terminal as part of existing bad block sequences.

When you have finished supplying information for the /UPD switch, press the ESCAPE key, ALTMODE key, or CTRL/Z in response to the prompt. The bad block descriptor file will then be rewritten with the new bad block information. Blocks that you enter manually and that BAD decides are reliable are included in the bad block descriptor file.

2.3 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 RET
```

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 invoked by entering only the file specification preceded by the at sign (@). For example:

```
>@INDIRECT.CMD RET
```

The format for entering file specifications is as follows:

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

The default values for indirect command file specifications are as follows:

Element	Default Value
ddnn	SY0
directory	The current directory
filename	No default
type	CMD
version	The latest version of the file

For complete information on how to use indirect command files, see the *RSX-11M-PLUS Indirect Command Processor Manual.*

2.4 Processing Bad Block Data

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

2.4.1 Verifying Devices

BAD verifies disks and DECtapes by writing a test pattern onto each of the blocks on the device, by reading the pattern into a buffer in memory, and by 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 (/PAT=m:n) may be used to specify the doubleword 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.

2.4.1.1 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 block information.

2.4.1.2 BAD and Last-Track Devices

BAD records bad block information differently on last-track devices from non-last-track devices. Last-track devices include the RK06/07, RL01/02, and the RM02/03/05/80 devices. 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 the bad block file [0,0]BADBLK.SYS when you issue the MCR command INI.

2.4.2 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 doubleword entry in the bad block descriptor file. The first word of this doubleword contains two entries: the high-order byte contains the number of bad blocks minus 1 and the low-order byte contains bits 16 to 23 of the logical block number of a bad block or a range of bad blocks. The second word of the doubleword contains bits 0 to 15 of that block number.

For last-track devices, bad block descriptor entries are recorded as a doubleword in the SDBSF. The first word of the doubleword 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.

2.4.3 INI Command and BAD

Use BAD followed by the MCR command INI to produce a Files–11 volume. The INI command uses the bad block information to create the bad block file [0,0]BADBLK.SYS. The bad block file has allocated to it those blocks found to be bad and the last good block for non-last-track devices, thus ensuring that the file system does not allocate a known bad block to a file.

For information on how to use the MCR command INI, see the *RSX–11M–PLUS Command Language Manual.*

2.5 Using BAD

Before BAD can validate a device, that device must be formatted by the manufacturer or by the Disk Volume Formatter (FMT) (see Chapter 9).

The volume to be formatted must be mounted foreign and allocated by the user.

You may execute BAD while other tasks are executing.

Note that if the Allocate switch (/ALO:volumelabel) (see Section 2.2.1) is used with BAD, the volume must be mounted as a Files–11 device and the user must have a privileged account.

Example

```
ALL DM1: [RET]
MOU DM1:/FOR [RET]
FMT DM1:[/switch] [RET]
BAD DM1:[/switch] [RET]
INI DM1:[label][/switch] [RET]
DMO DM1: [RET]
MOU DM1:[label][/switch] [RET]
```

Illustrates a typical sequence of steps for introducing a disk to an RSX-11M-PLUS operating system.

2.5.1 Programming Considerations

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

2.5.1.1 Use of Block 0

On bootable disks, block 0 contains the bootstrap block. If block 0 is bad, BAD prints a message warning the operator not to use the disk for a bootable system image.

2.5.1.2 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.

2.6 BAD Messages

The following sections list the BAD messages, give a brief description of the condition that causes each message, and suggest a response to the condition. BAD messages are either informational or error messages. The messages are arranged alphabetically beginning with the text following the device symbol, ddnn.

Messages are issued to your terminal in the following format:

```
BAD -- ddnn: message
```

Error messages describe a condition that caused BAD to terminate the processing of a command. When this occurs, BAD returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, BAD will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>BAD commandline
BAD -- ddnn: message
>
```

If the command is entered in response to the BAD prompt, BAD will issue an error message and prompt you with the BAD prompt, as follows:

```
BAD>commandline
BAD -- ddnn: message
BAD>
```

BAD—ddnn: 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—ddnn: Fatal hardware error

Explanation: There is a hardware problem with the specified disk.

User Action: Contact your DIGITAL Field Service representative.

BAD—ddnn: Handler/Driver missing

Explanation: The disk driver is not loaded.

User Action: Load the disk driver and reenter the command line.

BAD—ddnn: 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 /ALO:volumelabel switch.

User Action: The disk must be initialized by using the MCR command INI.

BAD—ddnn: 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 - n

Explanation: You entered an invalid block number sequence. The value n is the invalid sequence.

User Action: Specify another value and reenter the command line. This message applies to the /MAN, /ALO:volumelabel, or /UPD switches only.

BAD—Invalid switch

Explanation: The specified switch is not a legal one for the BAD utility.

User Action: Specify an appropriate switch and reenter the command line.

BAD—ddnn: 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—ddnn: 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: Specify the /OVR switch and reenter the command line.

BAD—ddnn: 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 that you specified the right device.

BAD—ddnn: 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—ddnn: 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—ddnn: 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 *RSX-11M-PLUS and Micro/RSX 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—ddnn: Volume label incorrect

Explanation: The volume label entered with the /ALO:volumelabel switch did not match the label on the disk.

User Action: Use the correct volume label and reenter the command line.

BAD—ddnn: Write locked

Explanation: The unit is write-locked.

User Action: Write-enable the unit and reenter the command line.

Chapter 3
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 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. BRUSYS is the standalone version. Refer to Section 3.5 for more information on standalone BRU.

Backup and restore operations take place on disk and magnetic tape volumes as follows:

- 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 allow you to do the following:

- Initialize disks
- Perform selective backup and restore operations
- Perform volume and data checking
- Display information such as backup set names and file names

Section 3.9 contains examples of various BRU operations.

BRU can also be invoked through the DIGITAL Command Language (DCL) command BACKUP. For more information, see the *RSX-11M-PLUS Command Language Manual*.

3.1 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 mounted foreign and mounted 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 directories and files against the User Identification Code (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 disks mounted foreign, no qualifier is necessary.

BRU also restores to volumes mounted foreign or mounted Files–11. Specify the /INITIALIZE qualifier to restore to a volume mounted foreign. This qualifier initializes the volume to Files–11 format. To restore to a mounted Files–11 volume, 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 mounted foreign. No qualifier is necessary.

Table 3–1 summarizes the correct combinations of mount status and qualifiers. BRU returns an error message for any wrong combination of conditions.

Table 3–1: Mounting and Initializing Volumes

Volume	Mount Status	Mandatory Qualifier
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 *RSX–11M–PLUS and Micro/RSX I/O Operations Reference Manual.*

With BRU, you can also specify that a disk volume contain up to 65,500 files. The default is the value assigned to the input disk. See the descriptions of the /HEADERS and /MAXIMUM qualifiers in Section 3.3 for more information.

3.1.1 Backup Sets

A backup set consists of all the data directed to a tape or disk volume (or volumes) during a single backup 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.

Files in the backup set cannot be accessed directly; therefore, the backup set must first be restored before the individual files in the backup set can be accessed.

A backup set created on a tape volume or volumes is called a tape set. Tape sets are created by a conventional backup operation.

A backup set created on a disk volume or volumes is called a disk set. Disk sets are created by an image backup operation.

3.1.1.1 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 the first volume. 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.

3.1.1.2 Image Backup 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 similar 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 3.3 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.

3.1.2 Full and Selective Backup Operations

You can classify a backup operation by how much of the original tape or disk you are backing up.

The following backup operations are available to you:

- Full
- Selective

A full backup or a selective backup refers to the files you are backing up.

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 is backed up. You select and identify the files to be backed up by directory, 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 too many system resources.

3.1.3 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 disks in image mode. You cannot mix disks.

3.1.4 Supported Devices

Table 3–2 lists all the devices that On-Line BRU supports. The disks from DB to EM are all block-structured devices.

Table 3–2: 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/RA82/RC25/RD51/RD52/RD53/RD54/RX33/RX50 disk
DX	RX11/RX01 diskette
DY	RX211/RX02 diskette
EM	ML11 electronic memory
MF	TM78/TU78 9-track magnetic tape (on VMS systems with VAX–11 RSX only)
MM	RH11/TM02-03/TE16/TU16/TU45/TU77 RH70/TM02-03/TE16/TU16/TU45/TU77 9-track magnetic tape
MS	TS11/TSV05/TU80 9-track magnetic tape
MT	TM11/TE10/TU10 7- or 9-track magnetic tape TS03 9-track magnetic tape
MU	TK50 cartridge tape drive TU81E 9-track magnetic tape

3.2 BRU Command Line

This section describes the rules for entering command lines for BRU. This section defines the command line format and describes prompts, command line parameters, and command qualifiers. BRU command lines have a maximum length of 256_{10} characters except in one case that uses continuation lines (see Section 3.2.2).

You can execute an indirect command file containing BRU commands from BRU. Type an at sign (@) followed by the file specification for the indirect command file.

You can invoke BRU by any of the methods described in Chapter 1. The command line for BRU is shown next.

Format

/qualifier[...] indevice[,...][filespec[,...]] outdevice[,...]

Parameters

qualifier

Specifies any of the command qualifiers described in Section 3.3. 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.

When a qualifier has options, you must separate the qualifier from the option with a colon (:).

You can use a shorter form of a qualifier as long as it is unique. All BRU qualifiers are unique to three characters. Table 3–3 lists the BRU qualifiers.

indevice

Specifies the input device you want to transfer files from. In a backup operation, the input device contains the files you want to safeguard. In a restore operation, the input device contains the backup set you are restoring.

Devices are specified in the following form:

```
ddnn:
```

Refer to Chapter 1 for a detailed decription of this parameter.

There is no default for the input device. It must be specified.

For a restore operation, up to eight input devices may be specified, separated by commas. BRU restores the backup set beginning on the first device and continuing on the other devices in the order specified. If the number of volumes in the backup set you are restoring exceeds the number of input devices specified, BRU prompts you to place additional volumes on the specified drives, one at a time.

filespec

Specifies the file specification used to select particular files or categories of files to back up or restore. A file specification takes the following form:

```
[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

You may specify as many as 16 file specifications per command, separating the file specifications by a comma. The file specification always follows the input device. If more than one input device is specified, the file specification must follow the first input device. Note that file specifications may not be placed after the output device.

When you enter a command without a file specification, all the files on the input volume are copied to the output volume.

Files can also be backed up or restored selectively by directory, file name, file type, or version number. If only the directory is specified, the remainder of the file specification will be treated as a wildcard; for example, [directory] is equivalent to [directory]*.*;*. If the version number is omitted, a wildcard is the default; for example, filename.type is equivalent to filename.type;*. Note, however, that BRU does not accept 0 or -1 as version numbers.

outdevice

Specifies the output device you want to transfer the files to. In a backup operation, the output device contains the backup set you want to create. In a restore operation, the output device is the disk that receives the files you are restoring.

There is no default for the output device. It must be specified.

In a backup operation, up to eight output devices may be specified, separated by commas. BRU creates the backup set beginning on the first device and continuing on the other devices in the order specified. If the number of volumes required for the backup set exceeds the number of output devices specified, BRU prompts you to place additional volumes on the specified drives, one at a time.

The format of outdevice is the same as for indevice (described previously). A file specification may *not* be placed after the output device.

If you enter a RETURN key as the only character on the command line, the following prompts will appear:

```
BRU>
FROM:
TO:
INITIALIZE [Y/N]:
```

The BRU> prompt requests that you enter the qualifiers needed to perform the desired operation. Refer to Table 3–3 for a summary of the BRU qualifiers.

The FROM prompt 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 previous description of the command line parameters.

The TO prompt requests that you enter the name (or names) of the output devices. The names should be in the form specified in the previous description of the command line parameters.

The INITIALIZE [Y/N] prompt requests that you enter Y (for Yes) if you want to initialize the output volume, or 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.

Table 3–3: Summary of BRU Command Qualifiers

Command Qualifiers	Options	Default
/APPEND		None
/BACKUP_SET:name		Volume name of the disk being backed up
/BAD[:option]	AUTOMATIC MANUAL OVERRIDE	BAD:AUTOMATIC
/BUFFERS:n		Number of File Control Blocks (FCBs) from the input disk
/COMPARE		None
/CREATED:option	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:n		Default density of drive
/DIRECTORY		None
/DISPLAY		None
/ERRORS:n		25_{10} errors
/EXCLUDE		None
/EXTEND:n		Number of blocks from the input disk
/HEADERS:n		Number of headers allocated to the input volume
/IDENTIFICATION		None
/IMAGE:option	SAVE RESTORE	None
/INITIALIZE		None
/INVOLUME:name		None
/LENGTH:n		The length of the output tape
/MAXIMUM:n		Maximum number of files allowed on the input volume
/MOUNTED		None
/NEW_VERSION		/NOSUPERSEDE

Table 3–3 (Cont.): Summary of BRU Command Qualifiers

Command Qualifiers	Options	Default
/NOINITIALIZE		None
/NOPRESERVE		None
/NOSUPERSEDE		/NOSUPERSEDE
/OUTVOLUME:name		Input disk volume name
/POSITION:option	BEGINNING MIDDLE END BLOCK:n	Index file position on the input disk
/PROTECTION:option	SYSTEM:value OWNER:value GROUP:value WORLD:value	Protection of the input disk
/REVISED:option	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

3.2.1 Wildcards in Input Specifications

The following wildcard (*) features are provided for input file directory specifications:

[*] Specifies all directories.

[*,*] Specifies all directories.

[g,*] Specifies all member numbers for group g.

[*,m] Specifies all group numbers for member m.

BRU also supports the wildcard in the file name, file type, and version number elements of the file specification. BRU follows the rules for use of wildcards (see the *RSX-11M-PLUS MCR Operations Manual*) except in the following two instances:

- If the version number is omitted, it will be treated as a wildcard. For example, NAME.EXT is the equivalent of NAME.EXT;*.
- If only the directory is specified, the entire file specification will be treated as a wildcard. For example, [directory] is the equivalent of [directory]*.*;*.

3.2.2 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 the RSX-11M-PLUS operating system, BRU supports continuation lines under all circumstances. However, when you use BRU from the Monitor Console Routine (MCR) command level, the total number of characters for the initial command line and any subsequent continuation lines cannot exceed 80.

Section 3.9 gives examples of continuation lines on the RSX-11M-PLUS operating system.

3.3 BRU Qualifier Descriptions

The following sections describe the BRU command qualifiers in detail.

3.3.1 Append

The Append qualifier (/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 is 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, the /APPEND qualifier 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 the /APPEND qualifier to rewind the tape and then space forward until the logical end-of-tape.

If the tape is a continuation volume (that is, not the first tape or disk 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, the /APPEND qualifier 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.

Note

BRU detects and automatically skips over a bootable system image (for example, BRUSYS) when appending to or restoring from a magnetic tape. This feature allows you to place a BRU backup set onto a magnetic tape that contains a bootable system image at its beginning and eliminates the need for a separate magnetic tape containing the bootable system image.

When BRU is used to append to or restore from a magnetic tape which contains a bootable system image, there will be a small delay for rewinding the tape and for skipping over the bootable system image.

When placing the first backup set on a magnetic tape that contains a bootable system image, you must specify /REW/APPEND. You cannot leave the magnetic tape positioned at the end of the bootable system image and then specify the /APPEND qualifier. Also, the /TAPE_LABEL qualifier is ignored during a restore operation if there is a bootable system image at the beginning of the magnetic tape.

3.3.2 Backup Set

The Backup Set qualifier (/BACKUP_SET) names the backup set (refer to Section 3.1.1) to be placed on tape or disk.

Format

/BACKUP_SET:name

Parameter

name

Specifies the name of the backup set.

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.

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 characters long.

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.

3.3.3 Bad

The Bad Qualifier (/BAD) creates the bad block file BADBLK.SYS on the output disk. The /BAD qualifier is used with the /INITIALIZE qualifier during tape-to-disk or disk-to-disk operations.

Format

/BAD[:option]

Parameter

option

Specifies one of the following three options:

AUTOMATIC — For last-track devices, the AUTOMATIC option causes BRU to use the manufacturer-written bad block information and the Software-Detected Bad Sector File (SDBSF) to create the bad block file BADBLK.SYS. For non-last-track devices, it uses the software bad block descriptor block to create BADBLK.SYS. AUTOMATIC is the default option.

MANUAL — 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 SDBSF (for last-track devices) or the bad block descriptor block (for non-last-track devices) to create the bad block file BADBLK.SYS.

OVERRIDE — The OVERRIDE option applies only to last-track devices, and it causes the last-track device to appear to be a non-last-track device. When OVERRIDE is specified, BRU uses the software bad block descriptor block to create the bad block file BADBLK.SYS and ignores the manufacturer-written information.

For complete information on how to use the options for the /BAD qualifier, see Section 3.6.

3.3.4 Buffers

The Buffers qualifier (/BUFFERS) 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 files contained in heavily used directories are found. The default number of buffers is the same as for the input disk.

Format

/BUFFERS:n

Parameter

n

Specifies the number of FCBs.

The /BUFFERS qualifier is used with the /INITIALIZE qualifier during tape-to-disk or disk-to-disk operations.

3.3.5 Compare

The /COMPARE qualifier (/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.

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.

3.3.6 Created

The Created qualifier (/CREATED) backs up or restores files created before or after the specified date and/or time.

Format

/CREATED:option

Parameter

option

Specifies one of the two options in three possible forms, as follows:

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

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.

3.3.7 Density

The Density qualifier (/DENSITY) specifies the density, in bits per inch (bpi), at which BRU writes to tape.

Format

/DENSITY:n

Parameter

n

Specifies the density. The following table 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
TU81E	6250	1600

If you specify the /DENSITY qualifier with the /APPEND qualifier, 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 bpi. 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.

3.3.8 Directory

The Directory qualifier (/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.

3.3.8.1 Displaying Backup Set Names

When specified with no backup set name, the /DIRECTORY qualifier lists all the backup sets on the volume.

Example

```
BRU>/DIRECTORY MM0:

VOL1     BACKUP1 LABEL1  2-Jan-87
VOL1     BACKUP2 LABEL1  3-Jan-87

BRU>/DIRECTORY DU0:

VOL1     BACKUP1 LABEL1  13-Jun-87
VOL2     BACKUP1 LABEL2  14-Jun-87
```

3.3.8.2 Displaying File Names

To display the names of files in a backup set, enter the backup set name by using the /BACKUP_SET qualifier with the /DIRECTORY qualifier.

If the backup set is not on the tape or disk, BRU halts execution and displays a message at your terminal.

Example

```
>RUN BRU
BRU>/BACKUP_SET:23MAY87A/DIRECTORY MM1:
VOL1.    23MAY87A         HWHDOC  13-JUN-87 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
>
```

Displays the names of the files in the backup set.

3.3.9 Display

The Display qualifier (/DISPLAY) prints, at your terminal, the file name and directory of each file as the header for that file is being transferred by BRU.

3.3.10 Errors

The Errors qualifier (/ERRORS) terminates a restore operation after the specified number of nonfatal tape-read errors is reached.

Format

/ERRORS:n

Parameter

n

Specifies the number of errors. The range of numbers is 0 to 65535. The default number of errors before termination is 25_{10}.

3.3.11 Exclude

The Exclude qualifier (/EXCLUDE) backs up or restores all of the files on the tape or disk except the files specified on the command line.

3.3.12 Extend

The Extend qualifier (/EXTEND) specifies the default number of blocks by which a file is extended when that file has exhausted its allocated space.

Format

/EXTEND:n

Parameter

n

Specifies the number of blocks. This value is used by the 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.

3.3.13 Headers

The Headers qualifier (/HEADERS) specifies the number of file headers to allocate initially to the index file.

Format

/HEADERS:n

Parameter

n

Specifies the number of file headers.

The five system files (INDEXF.SYS, BITMAP.SYS, BADBLK.SYS, CORIMG.SYS, and 000000.DIR) are not included in n. The primary reason for preallocating file headers is to locate them near the storage bit map file. (The storage bit map file is generally located in the middle of the disk.) Proper placement of file headers can help reduce head motion during I/O operations. The 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 the /HEADERS qualifier 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 MCR command INI in the *RSX-11M-PLUS MCR Operations Manual* and the DCL command INITIALIZE in the *RSX-11M-PLUS Command Language Manual* for a table of maximum and default values.

3.3.14 Identification

The Identification qualifier (/IDENTIFICATION) directs BRU to identify itself by displaying its version. This qualifier may be specified on a command line alone or in combination with other qualifiers.

3.3.15 Image

The Image qualifier (/IMAGE) specifies that you want to do a multiple disk-to-disk backup or restore operation.

Format

/IMAGE:option

Parameter

option

Specifies one of the following two options:

SAVE	If you are doing a backup operation, you must specify the SAVE option on the command line.
RESTORE	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.

3.3.16 Initialize

The Initialize qualifier (/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, BITMAP.SYS, CORIMG.SYS, and 000000.DIR.

The volume must be mounted foreign. BRU returns a privilege violation if the conditions are not met satisfactorily (see Table 3–1).

Along with the /INITIALIZE qualifier, you can specify the following qualifiers when you initialize a disk:

- /BAD
- /BUFFERS
- /EXTEND
- /HEADERS
- /MAXIMUM
- /NOPRESERVE
- /OUTVOLUME
- /POSITION
- /PROTECTION
- /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.

3.3.17 Involume

The Involume qualifier (/INVOLUME) specifies the volume label of the input disk.

Format

/INVOLUME:name

Parameter

name

Specifies the name of the volume label. 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, the /INVOLUME qualifier 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.

3.3.18 Length

The Length qualifier (/LENGTH) specifies the length of the output tape.

Format

/LENGTH:n

Parameter

n

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.

Note

This qualifier should not be used with cartridge tape devices such as the TK25 and TK50.

3.3.19 Maximum

The Maximum qualifier (/MAXIMUM) 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.)

Format

/MAXIMUM:n

Parameter

n

Specifies the number of files. The default is the maximum number of files on the input disk. The value includes the five system files.

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 the /MAXIMUM qualifier 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 MCR command INI in the *RSX-11M-PLUS MCR Operations Manual* and the DCL command INITIALIZE in the *RSX-11M-PLUS Command Language Manual* for a table of maximum and default values.

3.3.20 Mounted

The Mounted qualifier (/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 command MOUNT).

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 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 deletion of files) while BRU is running provides the following results:

- If the file is being changed while BRU is backing up the disk, BRU copies only the data that comprise 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 comprise the file may be corrupted.

  If the file ID from the deleted file is reused in a directory 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 logging 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 logging file to a different disk or exclude it with the /EXCLUDE qualifier.

3.3.21 New Version

The New Version qualifier (/NEW_VERSION) resolves file specification conflicts that occur during restore operations and during backups to a mounted disk.

When a file already exists on the output disk volume, the /NEW_VERSION qualifier creates a new version of the file. You cannot use the /VERIFY qualifier when you specify the /NEW_VERSION qualifier. If these qualifiers are specified together, BRU will issue a "Conflicting qualifiers" error message.

3.3.22 Noinitialize

The Noinitialize qualifier (/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 the /NOINITIALIZE qualifier. If you enter any of these qualifiers, BRU issues an error message.

3.3.23 Nopreserve

The Nopreserve qualifier (/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.

The /NOPRESERVE qualifier is used only with the /INITIALIZE qualifier.

When file IDs are not preserved, BRU assigns new file IDs and increments them sequentially.

3.3.24 Nosupersede

The Nosupersede qualifier (/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.

When an output file and an input file have identical file specifications but different version numbers, the /NOSUPERSEDE qualifier causes the input file to be copied, but it does not delete the output file. The /NOSUPERSEDE qualifier is the default.

3.3.25 Outvolume

The Outvolume qualifier (/OUTVOLUME) specifies the volume label of the output disk.

Format

/OUTVOLUME:name

Parameter

name

Specifies the name of the volume. 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 the /OUTVOLUME qualifier, 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.

3.3.26 Position

The Position qualifier (/POSITION) specifies the location of the index file on the output disk volume being initialized, usually to minimize access time.

Format

/POSITION:option

Parameter

option

Specifies one of the four possible location options, as follows:

BEGINNING
MIDDLE
END
BLOCK:n

The BEGINNING, MIDDLE, and END options specify the beginning, middle, and end of a volume. The BLOCK:n option specifies a block, n, where the index file is to be placed. Generally, the BEGINNING or END option is used only when a disk contains large contiguous files. MIDDLE is recommended to minimize access time.

When you use the BLOCK:n option, the number is decimal by default (the period (.) is optional). To specify an octal number, place a number 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.

3.3.27 Protection

The Protection qualifier (/PROTECTION) 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 the Ancillary Control Processor (ACP) when the volume is mounted. If you do not specify any values, they default to the protection values of the input disk.

Format

/PROTECTION:option

Parameter

option

Specifies one of the four possible options, as follows:

SYSTEM:value
OWNER:value
GROUP:value
WORLD:value

The protection value can be R (read), W (write), E (extend), or D (delete), or any combination of the four.

See the *RSX-11M-PLUS MCR Operations Manual* for an explanation of file protection.

3.3.28 Revised

The Revised qualifier (/REVISED) backs up or restores files revised or created on, before, or after the specified date and time.

Format

/REVISED:option

Parameter

option

Specifies one of the two options in three possible forms, as follows:

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

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.

3.3.29 Rewind

The Rewind qualifier (/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.

3.3.30 Supersede

The Supersede qualifier (/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 a multiple disk-to-disk backup or restore operation (using the /IMAGE qualifier), if you create a backup set file on a mounted volume, and a file with the same name exists, the /SUPERSEDE qualifier replaces this file.

The /NOSUPERSEDE qualifier is the default.

3.3.31 Tape Label

The Tape Label qualifier (/TAPE_LABEL) specifies the volume identifier 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.

Format

/TAPE_LABEL:label

Parameter

label

Specifies the 6-character identifier. This label allows you to check that you are using the correct tape.

3.3.32 UFD

The UFD qualifier (/UFD) directs BRU to create directories (if they do not already exist) on a mounted output volume, and it then directs BRU to copy into them the files from the same directory on the input volume. If you do not specify the /UFD qualifier, BRU does not copy the files.

The /UFD qualifier is used only with the /NOINITIALIZE qualifier.

3.3.33 Verify

The Verify qualifier (/VERIFY) verifies that the output volume was written correctly by comparing the input volume to the output volume and reporting any differences.

During a backup operation, each tape or disk is verified before starting the next volume in the backup set. During a restore operation, however, the entire backup set is restored before beginning the verify operation.

3.3.34 Windows

The Windows qualifier (/WINDOWS) specifies the default number of mapping pointers to be allocated for file windows when initializing an output disk.

Format

/WINDOWS:n

Parameter

n

Specifies the number of pointers. This value is used by the 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). If pool space is more critical than file access time, choose a smaller number of pointers.

Refer to the *RSX–11M–PLUS and Micro/RSX I/O Operations Reference Manual* for more information.

3.4 Command Qualifier Functions

When you initialize a disk by using the BRU qualifier /INITIALIZE, use the following qualifiers to specify various characteristics for the output disk:

- /BAD
- /BUFFERS
- /EXTEND
- /HEADERS
- /MAXIMUM
- /NOPRESERVE
- /OUTVOLUME
- /POSITION
- /PROTECTION
- /WINDOWS

The /MOUNTED 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 the following qualifiers are not available in standalone BRU (see Section 3.5):

- /NEW_VERSION
- /NOINITIALIZE
- /NOSUPERSEDE
- /SUPERSEDE
- /UFD

The following command qualifiers allow you to backup or restore data:

- /CREATED
- /EXCLUDE
- /REVISED

These qualifiers backup and restore data according to the following:

- File specification
- Date and time of creation
- Date and time of revision

The following qualifiers allow you to control backup and restore tape processing:

- /APPEND
- /BACKUP_SET
- /DENSITY
- /ERRORS
- /LENGTH
- /REWIND
- /TAPE_LABEL

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 following command qualifiers display information about the files being transferred:

- /DIRECTORY
- /DISPLAY

The /IDENTIFICATION qualifier directs BRU to identify itself by displaying its version number.

The /IMAGE qualifier is for both disk image backups and restore operations.

3.5 Standalone BRU

You can also run BRU stand alone. On the RSX-11M-PLUS operating system, the standalone system is called BRUSYS. The difference between the BRU task contained in this standalone system and the On-Line BRU described in the preceding sections is that standalone 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 standalone BRU systems are the /NEW_VERSION, /NOINITIALIZE, /NOSUPERSEDE, /SUPERSEDE, and /UFD qualifiers.

On an RSX-11M-PLUS operating system, BRUSYS contains four other tasks besides the BRU task. These other tasks are BAD, CNF, DSC, and FMT. The Bad Block Locator Utility (BAD) is described in Chapter 2, the Standalone Configuration and Disk Sizing Program (CNF) is described in this chapter, the Disk Save and Compress Utility Program (DSC) is described in Chapter 6, and the Disk Volume Formatter Utility (FMT) is described in Chapter 9. You invoke BAD, BRU, DSC, and FMT with the MCR command RUN. CNF is invoked automatically when you boot the system. (You can use CNF again by invoking it with the RUN command.)

If you need to use BAD or FMT, you must run them before running BRU. When you are finished with these other tasks, issue the RUN BRU command to begin using BRU.

BRU and BAD should not be used simultaneously because they use common buffer space. Running both tasks at the same time yields unpredictable results.

After you boot the standalone 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, as follows:

```
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	330	Present

Device	CSR	Vector	CSR Status
MS	172522	224	Not Present
MT	160000	320	Not Present
MU	174500	260	Present

```
Enter first device:
```

You can also use CNF and its switches to set the control and status register (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, as follows:

```
Enter first device: DM2:/CSR=177450/VEC=274
Enter second device: DL:
```

The CNF switches are listed in Table 3–4.

Table 3–4: CNF Switches

Switch	Format	Function
CSR	/CSR=n	Changes the default CSR number for the device.
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.
Formatter	/FOR=n	Changes the default formatter number for some of the magnetic tape devices. The qualifier is valid for only the MF- and MM-type devices. The initial default for n is 0.
Vector	/VEC=n	Changes the default vector number for the device.

3.5.1 Locating and Booting Standalone BRU

The BRUSYS system on the distribution kit requires 124K words of memory to run the five tasks (BAD, BRU, CNF, DSC, and FMT). The BRUSYS system image and symbol table are located in directory [6,54] on the distribution disk.

On the RSX–11M–PLUS operating system, you can bootstrap the standalone BRU system in one of two ways, as follows:

- Software boot standalone BRU by using the privileged MCR command BOOT as follows:

```
>INS $BOO
>BOO [6,54]BRUSYS
```

- Hardware boot standalone BRU following the hardware bootstrap procedure for your processor.

To create a hardware-bootable, standalone BRU tape from the distribution disk, use the Virtual Monitor Console Routine (VMR) to save the system image to tape as follows:

```
>MOU MT:/FOR
>SET /UIC=[6,54]
>ASN SY:=LB:
>RUN VMRM42
ENTER FILENAME: BRUSYS
VMR>SAVE MT:BRUSYS
VMR> CTRL/Z
```

This tape contains a hardware-bootable image of the standalone BRU system. (See the *RSX-11M-PLUS and Micro/RSX System Management Guide* for information on VMR.)

3.6 On-Line BRU Bad Block Processing

After you have formatted a disk with the Disk Volume Formatter Utility (FMT; see Chapter 9) and marked any bad blocks on it with the Bad Block Locator Utility (BAD; see Chapter 2), you can initialize it with BRU.

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), MANUAL, and OVERRIDE. The following sections describe how to use these options.

3.6.1 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-Detected Bad Sector File (SDBSF). For non-last-track devices, BRU uses the bad block descriptor block.

3.6.2 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 SDBSF (for last-track devices) or the bad block (for non-last-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 (LBN), followed by a count of how many consecutive blocks are bad.

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 "Duplicate blocks found" error message.

To get a list of the LBNs you have typed so far, type a slash (/) or press the RETURN key.

When you have finished entering bad blocks, type two slashes (//). BRU will then allocate the bad blocks that you have entered to BADBLK.SYS and continue processing.

3.6.3 Using the OVERRIDE Option

The OVERRIDE option applies only to last-track devices. It makes the disk appear to be a non-last-track device.

When you use OVERRIDE with BRU, ensure that the disk you are processing has been processed previously by the BAD utility with the /OVR switch specified. Using the BAD switch /OVR makes last-track devices look like non-last-track devices by using the last good block before the last track as the bad block descriptor block. The OVERRIDE 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.

3.7 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.

3.7.1 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 Virtual Monitor Console Routine (VMR) to remove and reinstall any tasks. (VMR is described in the *RSX-11M-PLUS and Micro/RSX System Management Guide*.)

3.7.2 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, see Sections 3.7.2.1 and 3.7.2.2.

3.7.2.1 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 3.3) 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).

3.7.2.2 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 command BOOT to boot the saved system on the output disk. Then you use the MCR command SAV /WB to write the correct boot block on the output disk.

You can copy a DB-, DM-, DU-, EM- or DR-type disk to any other of these controller types without having to SAVE the system again because the boot block for these devices is common on the RSX–11M–PLUS operating system. Use the following command to write the correct boot block on the output disk:

```
>SAV /WB RET
```

3.8 BRU File Treatment

The following sections describe how BRU treats file dates, file headers, file synonyms, and lost files.

3.8.1 Creation and Revision Dates of Files

BRU always preserves the creation and revision dates of files that it transfers. However, since BRU creates directories during a restore operation to a disk being initialized, and also when the /UFD qualifier is specified, the creation date of the directory is the date on which BRU created it.

3.8.2 File Headers

BRU preserves all characteristics of a file, if possible. There are three exceptions, as follows:

- If there is insufficient room on the output volume to restore the file contiguously, it is restored noncontiguously.
- The file name is updated in the file's header to match the directory entry.
- The physical end-of-file (EOF) in the user attribute area is updated to reflect the file's size correctly.

3.8.3 File Synonyms

File synonyms are files that have different names but share the same file ID and data. They can be created with the Peripheral Interchange Program (PIP) but also occur when a file ID from a deleted file is reused in a directory 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.

3.8.4 Lost Files

A file that is not contained in any directory is known as a lost file. BRU does not find lost files. To find lost files, use the File Structure Verification Utility (VFY) with the Lost switch (/LO) before using BRU to back up the disk. (The VFY utility is described in Chapter 14.)

3.9 BRU 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 to 6 and Examples 11 and 12 also show informational messages that BRU returns during some operations. The other 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 operations
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
14. Continuation command lines (from MCR)

Examples

1. This example 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.

   Use the following command lines to back up DM2 onto the magnetic tapes on MM0 and MM1:

```
>MOU DM2:/FOR [RET]
>MOU MM0:/FOR [RET]
>MOU MM1:/FOR [RET]
>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

>
```

Use the following command lines to restore the entire disk and rewind the first input tape. (The /INI qualifier specifies that DM2 will be initialized before the restore operation begins.)

```
>MOU DM2:/FOR [RET]
>MOU MM0:/FOR [RET]
>MOU MM1:/FOR [RET]
>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

>
```

2. This example 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.

   The following command lines cause BRU to initialize the output disk (DM1) and then back up all of the files on the input disk (DM0) onto it:

```
>MOU DM:/FOR [RET]
>MOU DM1:/FOR [RET]
>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]
>
```

3. This example 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.

   This command initializes the output disk (DM1) and tells BRU that the maximum number of files allowed on the disk will be $10{,}000_{10}$ 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.

```
>MOU DM:/FOR [RET]
>MOU DM1:/FOR [RET]
BRU>/INI/MAX:10000/HEA:5000/POS:BEG DM: DM1: [RET]
BRU - Completed

>
```

4. This example 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.

```
>MOU DL:label [RET]
>MOU DY:/FOR [RET]
>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

>
```

5. This example shows a image disk-to-disk restore operation. You must specify the RESTORE option on the command line with the /IMAGE qualifier.

```
>MOU DY:/FOR [RET]
>MOU DL:/FOR [RET]
>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 if 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

>
```

6. This example shows how to append backup sets on a multivolume disk, and it also shows how to 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 by using the /APPEND qualifier.

```
>MOU DB:/FOR [RET]
>MOU DK:/FOR [RET]
>MOU DK1:/FOR [RET]
BRU>/APP/IMA:SAVE/BACKUP:SECOND/INI [RET]
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
>
```

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:  DBO: [RET]
BRU - Mount disk 1 on DKO:. Press "RETURN" when done.
BRU - Starting disk 1 on DKO:
BRU - End of disk 1 on DKO:
BRU - Mount disk 2 on DKO:. Press "RETURN" when done.
BRU - Starting disk 2 on DK1:
BRU - End of disk 2 on DK1:
BRU - Completed
>
```

7. This example shows how to do an incremental backup operation by date and with tape verification.

   The following command lines back up all files on the disk that were revised after 5:00 P.M. on 14 February 1987. 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.

```
>MOU DM:/FOR [RET]
>MOU MT:/FOR [RET]
>BRU/REV:AFT:(14-FEB-87 17:00)/VER [RET]
FROM: DM: [RET]
TO:   MT: [RET]
```

8. This example shows how to do a selective backup operation by file specification.

   In this case, BRU backups up all the files in directory [7,10] and all the files with type MAC and CMD in directory [301,304] on the input disk to a magnetic tape.

```
>MOU DB:/FOR [RET]
>MOU MM:/FOR [RET]
>BRU DB:[7,10],[301,304]*.MAC,*.CMD [RET]
TO:   MM: [RET]
```

9. This example shows how to back up files from a mounted disk and then two ways to restore files to a mounted disk.

The following command line informs BRU that the input disk is mounted as a Files–11 device:

```
BRU>/MOU DL:[304,303],[7,326] MM: RET
```

The following command line restores the files in directory [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/NOINIT MM:[304,303] DB: RET
```

The following command line restores the files in directory [7,326] on the magnetic tape to the mounted disk volume without first initializing it. The /NEW_VERSION qualifier tells BRU to create a new version of any duplicate files.

```
>BRU/NOINIT/NEW MM:[7,326] DB: RET
```

10. This example 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.

    The command line appends the files in directory [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, as follows:

    ```
    >BRU/APP/VER/BAC:TODAY RET
    FROM: DB:[7,326] RET
    TO:   MM: RET
    ```

    The following 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 the /NOSUPERSEDE qualifier to resolve the conflict.

    ```
    >BRU/REW/BAC:TODAY/NOI MM:  DB: RET
    ```

11. The following example backs up all of the files on the input disk, except for those in directory [1,6], onto the magnetic tape:

    ```
    >BRU/MOU/EXC DM:[1,6] MM1: RET
    BRU - Starting tape 1 on MM1:

    BRU - End of tape 1 on MM1:

    BRU - Completed

    >
    ```

12. This example shows how to enter bad blocks manually and how to display them.

The 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 ends the entry of bad blocks and BRU continues the restore operation.

In this example, two locations of bad blocks for the disk were entered: there are two bad blocks starting at LBN 10500 and one 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, if specified, must be in decimal. The default number of bad blocks is 1.

```
>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

>
```

13. This example shows BRU continuation command lines (see Section 3.2.2 for more information).

    The RSX-11M-PLUS operating system supports continuation lines when you invoke BRU and then responds to the BRU> prompt. The command line can be 256_{10} characters long.

```
>BRU [RET]
BRU>/REWIND- [RET]
BRU>/INVOLUME:BACKUP- [RET]
BRU>/BACKUP_ SET:25MAY87- [RET]
BRU>/TAPE_ LABEL:BRU123 [RET]
FROM:   DMO: [RET]
TO:     MMO: [RET]
```

14. The RSX-11M-PLUS operating system 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).

```
>BRU/REWIND- [RET]
->/INVOLUME:BACKUP- [RET]
->BACKUP_SET:25MAY87- [RET]
->/TAPE_LABEL:BRU123 DMO: MMO: [RET]
```

3.10 BRU Messages

This section lists BRU messages, describes the meaning of each message, and suggests actions to correct the errors. Error messages from the BRU utility are preceded by the mnemonic BRU.

BRU has three kinds of messages: informational, warning, and fatal error.

- Informational messages do not require a response. An informational message provides information on the current state of your backup operation.
- Warning messages are marked *WARNING*. They inform you of conditions that require your consideration. Depending on the message and the operation, you decide what further action is called for, and the operation continues.
- Fatal error messages are marked *FATAL*. They inform you of conditions that caused BRU to terminate execution of your command. The operation does not complete.

The subsections that follow describe each type of message. The messages are listed in alphabetical order.

3.10.1 Informational Messages

BRU issues the following informational messages:

BRU—BRU version xx.xx

Explanation: This message identifies the version of the Backup and Restore Utility (BRU) being used.

User Action: No user action is required.

BRU—Completed

Explanation: The backup operation is complete.

User Action: Enter another BRU command line or exit by pressing CTRL/Z.

BRU—End of disk n on ddnn

Explanation: BRU finished transferring data or verifying the disk n on device ddnn.

User Action: No user action is required.

BRU—End of tape n on ddnn

Explanation: BRU finished transferring data or verifying the tape n on device ddnn.

User Action: No user action is required.

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—Mount disk n on ddnn:. Press "RETURN" when done

Explanation: BRU is requesting a disk for a backup or restore operation.

User Action: Mount the disk n on the drive ddnn.

BRU—Mount tape n on ddnn:

Explanation: BRU is requesting a tape for a backup or restore operation. This message prints every 2 minutes until the tape is loaded.

User Action: Mount the tape n on the drive ddnn.

BRU—Please answer yes or no

Explanation: BRU requires a YES or NO response.

User Action: Enter YES or NO at your terminal.

BRU—Starting disk n on ddnn:

Explanation: The disk n is being copied to (or from) the drive ddnn.

User Action: No user action is required.

BRU—Starting tape n on ddnn:

Explanation: The tape n is being copied to (or from) the drive ddnn.

User Action: No user action is required.

BRU—Starting verify pass

Explanation: The verify pass of a disk-to-disk operation is beginning.

User Action: No user action is required.

BRU—Starting verify pass tape n on ddnn:

Explanation: The tape n is being verified during a backup or restore operation.

User Action: No user action is required.

3.10.2 Warning Messages

BRU issues the following warning messages:

BRU—•WARNING•—Allocation failure [directory]filename.type;version

Explanation: During a copy to a mounted volume, there was not enough free space to copy the specified file.

User Action: Delete unnecessary files to create free space on the volume. Then, reenter the command line.

BRU—•WARNING•—Appending at default bpi on ddnn:

or

BRU—•WARNING•—Appending at 1600 bpi on ddnn:

Explanation: Either of these messages indicate that you specified the wrong tape density with the BRU qualifier /APPEND. 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: No user action is required. BRU continues at the density at which the tape was previously written.

BRU—•WARNING•—Bad block data error

Explanation: You entered an incorrect bad block location, count, or syntax.

User Action: Enter the correct bad block information by using the BRU qualifier /BAD:MANUAL.

BRU—•WARNING•—Block exceeds volume size

Explanation: You specified 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: The boot block is marked as bad on the output disk, so the disk is not hardware bootable.

User Action: If you need a hardware-bootable disk, wait until the operation completes. Then, repeat the procedure with a disk that has a usable boot block (logical block number (LBN) 0).

BRU—•WARNING•—Boot block is corrupt

Explanation: The input disk does not contain a valid boot block, so the output disk will not be hardware bootable.

User Action: If you need hardware-bootable output, use an input disk with a valid boot block (LBN 0).

BRU—•WARNING•—Boot block read error

Explanation: An error occurred while BRU was reading the boot block. Although the input disk may contain a valid boot block, BRU was unable to read it. The output disk will not be hardware bootable.

User Action: If you need a hardware-bootable disk, wait until the operation completes. Then, repeat the procedure.

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: If you need a hardware-bootable disk, wait until the operation completes. Then, repeat the procedure.

BRU—•WARNING•—Cannot restore contiguously [directory]filename.type;version

Explanation: The output device does not contain enough contiguous blocks to restore the indicated file contiguously. The file will be restored noncontiguously.

User Action: Use the Peripheral Interchange Program (PIP) (see Chapter 12) 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 [directory]filename.type;version
IO Error code number

Explanation: During a copy to a mounted volume, BRU encountered an error while attempting to close the specified file.

User Action: Determine the cause of the error from the I/O code (see the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual*). If it is correctable, delete the portion of the file that BRU has copied. Then, 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. (A data-ID record is a control record used by BRU.)

User Action: No user action is required. BRU continues the operation.

BRU—•WARNING•—Data record verify error [directory]filename.type;version
File ID number LBN number

Explanation: There is a difference in a data block on the input and output devices. The file ID of the file with the error and the LBN of the block follow the message.

If a directory is printed with a file name, it is the owner directory from the file's header, not the directory in which the file is contained.

User Action: Repeat the backup operation. If it fails again, repeat the operation with a different disk or tape.

BRU—•WARNING•—Data was lost due to IO errors
[directory]filename.type;version

Explanation: A read error resulted in missing data. The files may contain invalid data, and some of the files may not be restored.

User Action: Repeat the backup operation.

BRU—•WARNING•—Directory verify error

Explanation: A directory record on the input device did not match a directory record on the output device.

This error can occur when files are created, deleted, or renamed during the backup operation; BRU is unable to verify the data on the output device. It can also indicate a hardware error.

User Action: Be sure the contents of the input disk do not change during the data transfer or verify operations. Before repeating the operation, be sure there is no activity on the disk. If you are backing up or restoring the system disk, be sure there is no activity on the system (such as Resource Accounting and error logging). Then, reenter the command line.

BRU—•WARNING•—Disk out of sequence. Please mount correct disk.

Explanation: You mounted the wrong volume on the disk drive during a restore operation from a multidisk backup set.

User Action: Mount the correct disk on the drive.

BRU—•WARNING•—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 on the output disk. 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•—Duplicate blocks found

Explanation: You entered a bad block that is already listed in the bad block file.

User Action: No user action is required.

BRU—•WARNING•—EOT marker error

Explanation: During a backup operation, an error occurred while BRU was writing or verifying the end-of-tape (EOT) 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: Your response depends on the type of error that has occurred.

- On a write error, BRU rewinds the current tape and places it off line. BRU then asks you to mount a new tape. After you mount the new tape, BRU rewrites the data.
- 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 the beginning-of-tape (BOT).

BRU—•WARNING•—Error accessing file
IO error code number
File ID number

Explanation: During a copy to a mounted volume, 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 continues with the next file.

User Action: For a definition of the error code number, see the I/O error codes in the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual.*

After BRU finishes, delete the file. Then, reenter the command line and specify the file on which the error occurred.

BRU—•WARNING•—Error accessing UFD Skipping [directory]
IO Error code number

Explanation: During a copy to a mounted volume, an error occurred when BRU attempted to access a directory.

User Action: For a definition of the error code number, see the I/O error codes in the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual.* Determine the cause of the error from the I/O error code. If the error is correctable, try the copy operation again.

BRU—•WARNING•—Error reading data blocks
IO Error code number
File ID number LBN number
RECOVERED

Explanation: An I/O error occurred while BRU was reading a data block from the input disk. The file identification (ID) of the file that contains the block and the logical block number (LBN) of the block are displayed, as well as the I/O error code.

If RECOVERED is printed after the message, BRU recovered the block by reading the disk again.

BRU continues the operation.

User Action: For a definition of the error code number, see the I/O error codes in the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual.*

BRU—•WARNING•—Error reading UFD [directory]

Explanation: An I/O error occurred while BRU was reading a block from the specified directory. Any files contained in this block of the directory are not backed up.

User Action: After the BRU operation completes, reenter the command line. Specify the directory indicated in the error message.

BRU—•WARNING•—Error reading UFD header [directory]

Explanation: An error occurred while BRU was reading the header of the specified directory. Files in this directory are not backed up.

User Action: Reenter the command line and specify the directory indicated in the error message. If the error still occurs, use the /LO switch with the File Structure Verification Utility (VFY) to find the lost files (refer to Chapter 14).

BRU—•WARNING•—Extending index file

Explanation: The initial number of file headers is too small. If the number of blocks on the output disk is greater than or equal to 5120_{10}, BRU allocates 256_{10} additional headers. If the number of blocks on the output disk is less than 5120_{10}, BRU allocates 16_{10} additional headers.

User Action: No user action is required. BRU continues the operation with the extended file index.

BRU—•WARNING•—File header read error [directory]filename.type;version IO error code number

Explanation: An I/O error occurred while BRU was reading a file header, so the corresponding file was skipped. BRU continues with the next file.

User Action: For a definition of the error code number, see the I/O error codes in the *RSX–11M–PLUS MCR Operations Manual* or the *RSX–11M–PLUS Command Language Manual.*

BRU—•WARNING•—File header file verify error [directory]filename.type;version

Explanation: A file header is not the same on the input and output devices.

This error can occur when files are created, deleted, or renamed during the backup operation; BRU is unable to verify the data on the output device. It can also indicate a hardware error.

User Action: Be sure the contents of the input disk do not change during the data transfer or verify operations. Before repeating the operation, be sure there is no activity on the disk. If you are backing up or restoring the system disk, be sure there is no activity on the system (such as Resource Accounting and error logging). Then, reenter the command line.

BRU—•WARNING•—File ID area verify error

Explanation: The BRU-generated file ID area of a data record is not the same on the input and output devices.

User Action: Refer to the previous message, "File header file verify error."

BRU—•WARNING•—File ID sequence number error [directory]filename.type;version

Explanation: The two possible sources of this error are as follows:

1. The sequence number in the file ID of a file does not match the sequence number of the file's entry in the directory.
2. The sequence number of a directory does not match the sequence number of the directory's entry in the Master File Directory (MFD).

Therefore, the file or directory is not valid and is not copied.

User Action: Repeat the backup operation.

BRU—•WARNING•—File IDs will not be preserved

Explanation: File IDs cannot be preserved because the index file bit map on the output device is too small. The value you specified with the /MAXIMUM qualifier is too small.

User Action: No user action is required. BRU continues the operation without preserving file IDs. If you want the disk to be hardware bootable, perform the backup operation again, and specify a larger value with the /MAXIMUM qualifier.

BRU—•WARNING•—File marked for delete [directory]filename.type;version

Explanation: The marked-for-deletion bit (SC.MDL) of the system-controlled characteristics in the file header is set, indicating that the file is partially deleted. So, BRU does not copy the file.

User Action: No user action is required. BRU continues the operation with the next file.

BRU—•WARNING•—File not found [directory]filename.type;version

Explanation: During a backup operation, BRU cannot find a header in the index file (INDEXF.SYS) for a file or directory. BRU does not copy the file.

During the verify or compare pass of a restore operation, BRU cannot find the file on the output device.

User Action: No user action is required. BRU continues the operation with the next file.

BRU—•WARNING•—File not superseded [directory]filename.type;version

Explanation: You attempted to restore a file to a mounted volume by using the /NOSUPERSEDE qualifier (the default). The specified file was not restored because it already existed on the output disk.

User Action: If you want the file to be restored, reenter the command line. Specify the file and either the /SUPERSEDE or the /NEW_VERSION qualifier.

BRU—•WARNING•—Header ID record verify error

Explanation: The header-ID record generated by BRU on the output device is incorrect.

This error can occur when files are created, deleted, or renamed during the backup operation; BRU is unable to verify the data on the output device. It can also indicate a hardware error.

User Action: Be sure the contents of the input disk do not change during the data transfer or verify operations. Before repeating the operation, be sure there is no activity on the disk. If you are backing up or restoring the system disk, be sure there is no activity on the system (such as Resource Accounting and error logging). Then, reenter the command line.

BRU—•WARNING•—Header read error [directory]filename.type;version IO error code number

Explanation: During a backup operation, an I/O error occurred while BRU was reading a file header in the index file.

During a restore operation, this error is fatal; the operation terminates.

User Action: For a definition of the I/O error code number, see the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual.*

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 you should repeat the operation.

BRU—•WARNING•—Input volume structure level differs from output volume

Explanation: You initialized the output volume with a value for the /MAXIMUM qualifier that is greater than 25,593. As a result, the index file (INDEXF.SYS) on the output volume has more than one file header.

User Action: To make the volume structure the same on the input and output devices, initialize the output volume with a value for the /MAXIMUM qualifier that is less than 25,593.

BRU—•WARNING•—Internal error

Explanation: BRU has detected an error within itself.

User Action: Reenter the command line. If the problem recurs, there is a problem with your software.

BRU—•WARNING•—Invalid date or time [directory]filename.type;version

Explanation: BRU encountered an invalid date or time in a file header during an incremental backup.

User Action: No user action is required. BRU continues the operation and the file is copied.

BRU—•WARNING•—Invalid disk format

Explanation: If you were performing an /IMAGE:RESTORE operation, you mounted a disk that is not a multivolume backup disk. If you were performing an /IMAGE:SAVE operation, you attempted to append to a disk that is not a multivolume backup disk. The disk may be corrupted.

User Action: Mount the correct disk and reenter the BRU command line.

BRU—•WARNING•—MFD read error

Explanation: An I/O error occurred while BRU was reading a block of the Master File Directory (MFD). BRU cannot copy the directories or files in that block of the MFD.

User Action: Reenter the command line. If the header still cannot be read, the files on the disk are "lost." Use the /LO switch with the File Verification Utility (VFY) to recover the lost files (refer to Chapter 14).

BRU—•WARNING•—No bad block data found

Explanation: You did not use the BAD utility to produce a file of the output disk's bad blocks. BRU continues the operation.

User Action: For more information on bad block processing by BRU, see Section 3.6.

BRU—•WARNING•—No files found

Explanation: During a backup or restore operation, BRU did not find any files to transfer. You may have entered the wrong file or directory specification.

User Action: If you are sure you entered the correct file and directory specifications, no further action is necessary. If not, use the correct file specification and reenter the command line.

BRU—•WARNING•—Nonfatal qualifier conflicts being ignored

Explanation: You entered a qualifier that conflicts with the rest of the command line. You used the /REWIND, /DENSITY, or /LENGTH qualifier during a disk-to-disk copy operation; or you used the /NEW_VERSION, /SUPERSEDE, or /NOSUPERSEDE qualifier during a disk to tape backup operation.

User Action: No user action is required. BRU ignores all the conflicting qualifiers for the operation.

BRU—•WARNING•—No such UFD exists. Skipping [directory]

Explanation: During a copy to a mounted volume, BRU encountered one or more files in the specified directory on the input volume. However, there is no corresponding directory on the output volume.

User Action: Specify the /UFD qualifier to create the directory and reenter the command line.

BRU—•WARNING•—Open error [directory]filename.type;version
IO error code number
File I/D number

Explanation: During a copy operation to a mounted volume, an error occurred while BRU was attempting to open the specified file.

User Action: For a definition of the error code number, see the I/O error codes in the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual.* If the error is correctable, delete any portion of the file already copied by BRU. Then, reenter the command line.

BRU—•WARNING•—Privilege violation [directory]filename.type;version

Explanation: During a backup operation, you attempted to copy a file that you did not have read access to.

User Action: No user action is required. BRU does not copy the file.

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: No user action is required.

BRU—•WARNING•—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—•WARNING•—Tape label error on ddnn
IO Error code number

Explanation: An I/O error occurred while BRU was reading or writing a tape label. A read error is not fatal as long as BRU can continue reading the tape. If a write error has occurred, BRU will display the message "BRU—Mount another tape."

User Action: Specify a different tape and reenter the command line.

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.

This error can occur when files are created, deleted, or renamed during the backup operation; BRU is unable to verify the data on the output device. It can also indicate a hardware error.

User Action: Be sure the contents of the input disk do not change during the data transfer or verify operations. Before repeating the operation, be sure there is no activity on the disk. If you are backing up or restoring the system disk, be sure there is no activity on the system (such as Resource Accounting and error logging). Then, reenter the command line.

BRU—•WARNING•—Tape out of sequence. Please mount the correct tape.

Explanation: You mounted the wrong tape volume on the tape drive during a restore operation from a tape backup set.

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 by using /REWIND before attempting the append operation.

BRU—•WARNING•—Tape read error

Explanation: A nonrecoverable I/O error occurred while BRU was reading a tape. BRU ignores the error.

User Action: No user action is required.

BRU—•WARNING•—Tape write error
IO 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.

User Action: If the error is related to the tape drive, terminate the BRU operation and begin again from another tape drive.

BRU—•WARNING•—This disk will not contain a hardware bootable system

Explanation: The output disk will not be hardware bootable. This can be caused by one of the following:

- The input disk is not bootable.
- The input disk is not the same size or type as the output disk.
- The system image (RSX11M.SYS) was not copied.

This message is not issued when BRU is restoring to a mounted volume.

User Action: If you want the output disk to be hardware bootable, use a bootable input disk and an output disk that is the same size as the input disk. Also, be sure the system image is included in the operation.

BRU—•WARNING•—UFD record verify error

Explanation: A directory record is not the same on the input and output devices.

This error can occur when files are created, deleted, or renamed during the backup operation; BRU is unable to verify the data on the output device. It can also indicate a hardware error.

User Action: Be sure the contents of the input disk do not change during the data transfer or verify operations. Before repeating the operation, be sure there is no activity on the disk. If you are backing up or restoring the system disk, be sure there is no activity on the system (such as Resource Accounting and error logging). Then, reenter the command line.

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 directory entry was corrupted on the input disk.

User Action: No user action is required.

BRU—•WARNING•—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—•WARNING•—Wrong backup set

Explanation: During a restore operation from a multivolume backup set, BRU found that one of the tapes or disks does not contain the correct backup set.

User Action: Specify the correct tapes or disks and reenter the command line.

BRU—•WARNING•—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: Specify the correct input volume label and reenter the command line.

3.10.3 Fatal Error Messages

BRU issues the following fatal error messages:

BRU—•FATAL•—Allocation for system file exceeds volume limit

Explanation: One of the five system files (file type SYS) that are created when a volume is initialized does not fit on the output disk. This message usually occurs when 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: You specified a BRU option that is not unique. For example, the B in /POSITION:B could mean either BEGINNING or BLOCK.

User Action: Use at least three characters to specify a unique form of a BRU option.

BRU—•FATAL•—Ambiguous qualifier

Explanation: You specified a BRU qualifier that is not unique. For example, /RE could mean either /REVISED or /REWIND.

User Action: Use at least three characters to specify a unique form of a BRU qualifier.

BRU—•FATAL•—Attach failed on ddnn:

Explanation: You specified a device that BRU cannot access.

User Action: Use the DCL command SHOW DEVICE or the MCR command DEVICES to see if the device is in use (allocated or mounted) by another user. Allocate an available device and repeat the operation.

BRU—•FATAL•—Bad block data error

Explanation: An error occurred while reading the bad block information on the output disk.

User Action: Use the BAD utility (see Chapter 2) to re-create the bad block information or use another disk.

BRU—•FATAL•—Bad block file full

Explanation: You entered more than 204_{10} sets of contiguous bad blocks. You cannot enter more bad blocks than the bad block file can (BADBLK.SYS) hold.

User Action: If the number of contiguous bad blocks on your disk exceeds 204_{10}, you can no longer use it effectively. Use another disk.

BAC- •FATAL•—Cannot append on continuation volume

Explanation: The /APPEND qualifier was specified with a continuation volume. A backup set can be appended to the first volume only.

User Action: Mount a blank volume and reenter the command line without the /APPEND qualifier.

BRU—•FATAL•—Close or write attributes error [directory]filename.type;version IO Error code number

Explanation: During a copy to a mounted volume, BRU encountered an error while attempting to close the specified file.

User Action: Determine the cause of the error from the I/O code (see the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual*). If it is correctable, delete the portion of the file that BRU has copied. Then, reenter the command line.

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 without one of the mutually exclusive qualifiers.

BRU—•FATAL•—Device conflict

Explanation: You specified both tape and disk drives as part of either the input device specification (for a restore operation) or the output device specification (for a backup operation).

User Action: Specify a single type of device (that is, *either* a tape *or* a disk drive) and reenter the command line.

BRU—•FATAL•—Device not in system

Explanation: A device was specified that is not known to the sy: em.

User Action: To obtain a list of the devices currently available, use the SHOW DEVICE command. Specify a device that is known to the system and reenter the command line.

BRU—•FATAL•—Device not mounted files 11 on ddnn:

Explanation: The BRU qualifiers specified on the command line cannot be used unless the device is mounted with the Files–11 format.

User Action: Mount the device Files–11 and reenter the command line, or use a different command line. (For a list of the correct combinations of mounted devices and qualifiers, see Table 3–1.)

BRU—•FATAL•—Device not mounted foreign on ddnn:

Explanation: The BRU qualifiers specified on the command line cannot be used unless the device is mounted foreign.

User Action: Mount the device foreign and reenter the command line, or use a different command line. (For a list of the correct combinations of mounted devices and qualifiers, see Table 3–1.)

BRU—•FATAL•—Device not supported

Explanation: You specified a device that is not supported by the operating system.

User Action: Check the command line for valid device names. Then, correct any errors and reenter the command line.

BRU—•FATAL•—Directive error

Explanation: An internal error occurred in BRU. This may indicate that pool is low.

User Action: Reenter the command line. If the problem recurs, there may be a problem with your software.

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 on the output disk. Then, reenter the BRU command line.

For information on the BAD utility, refer to Chapter 2.

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 on the output disk. Then, reenter the BRU command line.

For information on the BAD utility, refer to Chapter 2.

BRU—•FATAL•—Doubly defined qualifier

Explanation: You specified the same qualifier more than once on the command line.

User Action: Specify the qualifier only once and reenter the command line.

BRU—•FATAL•—End of volume encountered. Backup set not found.

Explanation: You specified a backup set that is not on the tape or disk volume.

User Action: Mount the correct tape or disk volume. If you have the correct volume mounted, specify the correct backup set name and reenter the command line.

BRU—•FATAL•—Error limit exceeded

Explanation: BRU reached the maximum number of read errors allowed and terminated execution.

User Action: Using a different tape drive, reenter the command line. If you do not have another drive, clean the tape drive heads on the original drive. Then, reenter the command line.

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—•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: Use the BAD utility (see Chapter 2) with the /OVR switch to locate all bad blocks. Then, specify the /BAD:OVERRIDE qualifier and reenter the command line.

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: Specify a larger value with the /MAXIMUM qualifier and reenter the command line.

BRU—•FATAL•—File not found

Explanation: You specified an indirect command file or backup set file that BRU could not find.

User Action: Specify the correct file name and reenter the command line.

BRU—•FATAL•—Handler not resident

Explanation: You specified a device that does not have a driver loaded in memory.

User Action: Load the driver for the specified device. If the driver you intended to use is currently loaded, then specify the correct device name and reenter the command line.

BRU—•FATAL•—Header read error [directory]filename.type;version
IO error code number

Explanation: During a restore operation, an I/O error occurred while BRU was reading a file header in the index file. The operation terminates.

User Action: See the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual* for a definition of the error code number.

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: Before initializing the disk, use the BAD utility (see Chapter 2) to find all the bad blocks. Then, reenter the BRU command line.

BRU—•FATAL•—Illegal use of directory qualifier

Explanation: You specified one of the following:

- The /DIRECTORY qualifier with a disk or tape that is not part of a backup set
- The /INI qualifier (or one of its related qualifiers) with the /DIRECTORY qualifier

User Action: If you are initializing a device, do not use the /DIRECTORY qualifier.

BRU—•FATAL•—Inconsistent /INITIALIZE qualifiers

Explanation: You specified the /NOINITIALIZE qualifier with one or more of the initialization qualifiers for the output disk.

User Action: Reenter the /NOINITIALIZE command line without using any of the initialization qualifiers.

BRU—•FATAL•—Index file header read error
IO error code number

Explanation: An I/O error occurred while BRU was reading the header of the index file on the input disk.

User Action: For a definition of the I/O error code number, see the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual.* 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 (refer to Chapter 2) to identify the bad blocks on the output disk; then, reenter the command line.

BRU—•FATAL•—INDEXF.SYS is full

Explanation: The index file (INDEXF.SYS) cannot map any more file headers.

User Action: Specify larger values for the /MAXIMUM and /HEADER qualifiers and reenter the command line.

BRU—•FATAL•—Input device equals output device

Explanation: The input and output devices must be different.

User Action: Specify different input and output devices and reenter the command line.

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—•FATAL•—Internal error

Explanation: BRU has detected an error within itself.

User Action: Reenter the command line. If the problem recurs, there is a problem with your software.

BRU—•FATAL•—Invalid date or time

Explanation: You specified an incorrect date or time in the command line.

User Action: Specify the correct date or time and reenter the command line.

BRU—•FATAL•—Invalid density

Explanation: This error occurs for one of two reasons:

- You specified a density tape that was not 1600 (the default), 800, or 6250 bpi.
- You attempted to use both 7-track and 9-track tapes in a multivolume tape set.

User Action: If you specified the wrong density, reenter the command line with the correct density. For a multivolume tape set, use one type of tape (7 track or 9 track).

BRU—•FATAL•—Invalid disk format

Explanation: If you were performing an /IMAGE:RESTORE operation, you mounted a disk that is not a multivolume backup disk. If you were performing an /IMAGE:SAVE operation, you attempted to append to a disk that is not a multivolume backup disk.

User Action: Mount the correct disk and reenter the BRU command line.

BRU—•FATAL•—Invalid tape format

Explanation: An invalid tape record was read during a restore operation.

User Action: No user action is required. The invalid record is not restored.

BRU—•FATAL•—Invalid filename

Explanation: The name of the indirect command file is not syntactically correct.

User Action: Reenter the command line with the correct file name.

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 the description of the qualifier to determine the correct values for it.

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: Using the BAD utility with the /OVR switch, check the disk for bad blocks. Then, specify the /BAD:OVERRIDE qualifier and reenter the command line.

BRU—•FATAL•—MFD read error

Explanation: An I/O error occurred while BRU was reading a block of the Master File Directory (MFD). BRU cannot copy the directories or files in that block of the MFD.

User Action: Reenter the command line. If the header still cannot be read, the files on the disk are "lost." Use the File Verification Utility (VFY) with the /LO switch to recover the lost files (refer to Chapter 14).

BRU—•FATAL•—Missing colon

Explanation: A qualifier option that accepts a value was not followed by a colon.

User Action: Reenter the command line and include the missing colon and value.

BRU—•FATAL•—More than 1 level of indirection

Explanation: BRU does not support more than one level of indirect command files. You cannot nest indirect command files when you are using BRU.

User Action: Specify an indirect command file that does not use other command files and reenter the command line. That is, use only one level of indirect command files.

BRU—•FATAL•—Name exceeds maximum allowed length

Explanation: You specified a name (such as a backup set name) that is longer than 12 characters.

User Action: Specify a name that is less than 12 characters.

BRU—•FATAL•—Number of headers inconsistent with maximum files

Explanation: During an attempt to initialize an output volume, you specified a maximum number of files (/MAXIMUM) that is inconsistent with the number of headers specified with the /HEADERS qualifier.

User Action: For the valid ranges of values for the /MAXIMUM and /HEADERS qualifiers, see the description of the INITIALIZE command in the *RSX–11M–PLUS MCR Operations Manual* or the *RSX–11M–PLUS Command Language Manual.*

BRU—•FATAL•—Open error [directory]filename.type;version
IO error code number
File ID number

Explanation: During a copy operation to a mounted volume, an error occurred while BRU was attempting to open the specified file.

User Action: See the I/O error codes in the *RSX–11M–PLUS MCR Operations Manual* or the *RSX–11M–PLUS Command Language Manual* for a definition of the error code number. If the error is 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 because of 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 or disk that is already full.

User Action: Use another output disk or tape with more free space, or reenter the command line with a subset of the files specified.

BRU—•FATAL•—Override invalid with non-last-track device

Explanation: The /BAD:OVERRIDE qualifier is valid only when the output disk is a last-track device.

User Action: For information on the /BAD qualifier, see Section 3.3.3.

BRU—•FATAL—Privilege violation

Explanation: The mount status of one of the devices is inconsistent with the qualifiers specified in the command line. For example, the device may be mounted FOREIGN, but the qualifiers you used apply to Files–11 devices only.

User Action: Check the mount status of the device. Then, reenter the command line with the appropriate qualifiers (see Table 3–1).

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 two reasons:

- The output disk is too fragmented because of bad blocks.
- 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, with the /NOPRESERVE qualifier specified.

If you want to preserve file IDs, do one of the following:

- If the output disk is too fragmented, use the BAD utility (refer to Chapter 2) to display the number of bad blocks. If it contains a large number of bad blocks, use a different disk.
- Use the PIP switch /FR (see Chapter 12) to display the number of free file headers on the input disk. If there are fewer than four free headers, delete some of the files and reenter the command line. If you still do not have enough file headers, specify the /NOPRESERVE qualifier in the command line.

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—•FATAL•—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 user 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—•FATAL•—Stack overflow in sort routine

Explanation: BRU has detected an error within itself.

User Action: Reenter the command line. If the problem recurs, there is a problem with your software.

BRU—•FATAL•—Syntax error

Explanation: The command line is invalid.

User Action: Use valid syntax and reenter the command line.

BRU—•FATAL•—Tape label error on ddnn
IO Error code number

Explanation: An I/O error occurred while BRU was writing a tape label on the first tape of an append operation.

User Action: Specify a different tape and reenter the command line.

BRU—•FATAL•—Tape not at BOT. No rewind or append specified

Explanation: For a backup operation to tape, unless you specify the BRU qualifier /APPEND, BRU does not process a tape that is not at the beginning of the tape (BOT).

User Action: If you want to start writing at the beginning of the tape, use the /REWIND qualifier.

You can append to tape only at the end of the last backup set. If the tape is already positioned there, specify the /APPEND qualifier in the command line. If it is not positioned at the end of the last backup set, specify both the /REWIND and /APPEND qualifiers in the command line.

BRU—•FATAL•—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 by using /REWIND before attempting the append operation.

BRU—•FATAL•—Tape positioning error. No EOV encountered.
I/O Error code number.

Explanation: The tape spacing operation failed to find the end-of-tape (EOT) volume for an append operation.

User Action: Reenter the command line.

BRU—•FATAL•—Tape read error

Explanation: A nonrecoverable I/O error occurred while BRU was reading a tape. BRU terminates the operation.

User Action: No user action is required.

BRU—•FATAL•—Tape to tape not supported

Explanation: BRU does not back up a tape to another tape.

User Action: Use a fixed disk to perform a backup operation.

BRU—•FATAL•—Too many devices

Explanation: You may specify only one disk drive as an input device for a backup operation or as an output device for a restore operation.

Up to eight tape drives may be specified as output devices for a backup operation or as input devices for a restore operation. If the /IMAGE qualifier is specified, up to eight disk drives may be specified as output devices for a backup operation (/IMAGE:SAVE) or as input devices for a restore operation (/IMAGE:RESTORE).

For a disk-to-disk operation, only one disk drive may be specified as an input device and only one disk may be specified as an output device.

User Action: Specify the correct number of drives and reenter the command line.

BRU—•FATAL•—Too many file specifications

Explanation: You specified more than 16 file specifications on the command line.

User Action: Reenter the command line. To reduce the number of file specifications on the command line, use wildcards.

BRU—•FATAL•—UFD or MFD requires unsupported extension headers

Explanation: BRU does not support extension headers for Master File or User File directories (MFDs or UFDs, respectively).

User Action: Reenter the command line. If the problem recurs, there is a problem with your software.

BRU—•FATAL•—Unknown option

Explanation: You specified an option that BRU does not recognize.

User Action: Check the qualifiers and options, and then reenter the command line.

BRU—•FATAL•—Unknown qualifier

Explanation: You specified a qualifier that BRU does not recognize.

User Action: Check the qualifiers and reenter the command line.

BRU—•FATAL•—Unsupported structure level

Explanation: The file structure level on the input disk is not supported by BRU.

User Action: Check to be sure you have placed the correct disk in the drive.

For more information on file structures, see the descriptions of the /HEADERS and /MAXIMUM qualifiers.

BRU—•FATAL•—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 directory entry was corrupted on the input disk.

User Action: No user action is required.

BRU—•FATAL•—Verify lost

Explanation: During the verify pass, BRU lost synchronization between the input and the output. Either the tape position was lost, or you backed up from a disk that was changed during the backup operation.

This error can occur when files are created, deleted, or renamed during the backup operation; BRU is unable to verify the data on the output device. The message can also indicate a hardware error.

User Action: Be sure the contents of the input disk do not change during the data transfer or verify operations. Before repeating the operation, be sure there is no activity on the disk. If you are backing up or restoring the system disk, be sure there is no activity on the system (such as Resource Accounting and error logging). Then, 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. Then, reenter the command line.

If you have the correct tape, reenter the command line with 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 multivolume backup set, BRU found that one of the tapes or disks did not contain the correct backup set.

User Action: Specify the correct tapes or disks and reenter the command line.

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: Specify the correct input volume label and reenter the command line.

Chapter 4
File Compare Utility (CMP)

The File Compare Utility (CMP) compares two ASCII text files. The maximum record length of the files is 132_{10}. The files are compared line by line to determine whether parallel records are identical. CMP allows you to 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, and 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 (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. Use of these switches allows you to 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.

4.1 CMP Command Line

You can invoke CMP by any of the methods described in Chapter 1. The command line for DMP is shown next.

Format

[outfile[/switch[...]]=] infile1,infile2

Parameters

outfile

Specifies the file specification for the output file.

The format for entering file specifications is as follows:

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

This file can be in one of the following three formats, depending on the switch you specify in the command line:

- Differences
- Change bar
- SLP command input

The defaults for outfile are as follows:

SY0	Specifies the user's default system device.
directory	Specifies the directory that CMP is running under.
FILCOM	Specifies the default file name.
LST	Specifies the default file type.

If you do not specify an output file, the output defaults to your terminal.

If you do not specify a file version number, the default is the most recent version of the file plus one.

switch

Specifies switches that you apply to the output file specification. Some of the switches can be negated and some are mutually exclusive. Refer to Section 4.1.1 for details on switches.

infile1

Specifies 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

Specifies the file specification for the input file to be compared to infile1. You do not need a complete file specification. The specifications for infile1 are used as defaults for any unspecified portions of infile2.

If you type an equal sign (=) but give no output file specification, only the total number of differences is output to your terminal.

Examples

```
CMP>FILE1.MAC,FILE2.MAC RET
```

Lists the differences between FILE1.MAC and FILE2.MAC on your terminal.

```
CMP>DB2:[42,10]EXEC.;2 RET
```

Interprets the second input file as DB2:[42,10]EXEC.MAC;2.

```
CMP>=FILE1.MAC;1,FILE2.MAC;1 RET
10 differences found
```

Lists the total number of differences.

4.1.1 CMP Switches

Table 4–1 lists the CMP switches, and describes the function of each one. Table 4–2 gives the default setting for each one. You specify switches after the output file in the command line.

Table 4–1: CMP Switches

Switch Format	Function
/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. The /-BL switch is the default.
/CB [/-CB]	Specifies that CMP list infile2 with change bars, in the form of exclamation points (!), 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. The /-CB switch is the default. You can change the change bar character from the exclamation point to any character you wish by means of the /VB switch.
[/CO] /-CO	Specifies that CMP include comments (that is, text preceded by a semicolon (;)) in compare processing. The /CO switch is the default.
[/DI] /-DI	Specifies that CMP list the differences between the two files (rather than marking the lines in infile2). The /DI switch is the default. The /CB and /DI switches are mutually exclusive. If you specify both, the /CB switch overrides the /DI switch.
/FF [/-FF]	Specifies that CMP include records consisting of a single form-feed character in compare processing. The /-FF switch is the default.
/LI:n [/LI:3]	Specifies that n lines must be identical before CMP recognizes a match. The /LI:3 switch is the default. 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. The /LN switch is the default. If you specify the /SL switch, the /LN switch is unnecessary.
[/MB] /-MB	Specifies that CMP include all blank and tab characters in a line in compare processing. If you specify the /-MB switch, 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. The /MB switch is the default.
/SL[:au] [/-SL]	Directs CMP to generate an output file suitable for use as SLP command input. When you specify the /SL switch, CMP generates the SLP command input necessary to make infile1 identical to infile2. If a 1- to 8-character alphanumeric symbol is included in the /SL switch (:au), an audit trail is specified for SLP input. Section 4.1.2.3 gives an example of how CMP generates SLP command input. (For information on how SLP processes source files, refer to Chapter 13.) The /-SL switch is the default.

Table 4–1 (Cont.): CMP Switches

Switch Format	Function
/SP[:n] [/-SP]	Specifies that the output file be spooled on the line printer. You can optionally specify the number, n, (in octal or decimal) of files to be spooled. The /-SP switch is the default. This switch applies only if you have the Queue Manager system installed.
[/TB] /-TB	Specifies that CMP include all trailing blanks on a line in compare processing. If you specify the /-TB switch, CMP ignores all blanks following the last nonblank character on a line. When you specify the /-CO and the /-TB switches together, blanks that precede a semicolon (;) are considered trailing blanks and are ignored. The /TB switch is the default.
/VB:nnn [/VB:041]	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). The default switch is /VB:041 for an exclamation point.

CMP default switch settings are listed in Table 4–2.

Table 4–2: Summary of CMP Default Switch Settings

Switch	Function
/-BL	Does not compare blanks.
/-CB	Does not generate change bars.
/CO	Compares comments.
/DI	Lists only the differences between the two files.
/-FF	Does not compare form-feed characters.
/LI:3	Finds three identical lines before a match occurs.
/LN	Generates numbered lines.
/MB	Compares all blank and tab characters.
/-SL	Does not generate an output file suitable for use as SLP command input.
/-SP	Does not spool the output file.
/TB	Compares all trailing blanks.
/VB:041	Sets the exclamation point (ASCII 041) as the change bar character (used with the /CB switch).

4.1.2 Formats of CMP Output Files

CMP uses the two input files you specify on the command line to create an output file. CMP compares each line in infile1 to its sequential counterpart in infile2. When there are differences between the two files, CMP displays those differences in one of the following three output formats:

- Differences format (/DI switch)
- Change bar format (/CB switch)
- SLP command input format (/SL switch)

The default is the Differences format.

Sections 4.1.2.1 to 4.1.2.3 give examples of each of these formats. The following files are used as infile1 (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

4.1.2.1 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, and 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.

Refer to Section 4.1.2 for a description of the contents of files TEST1.DAT and TEST2.DAT.

Example

```
CMP>TESTDIF.DAT=TEST1.DAT,TEST2.DAT [RET]
```

Contains no switches, and generates the following output file:

```
*****************************************
   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. 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 the /LI switch was not specified, the number of lines required for a match defaults to 3. Thus, CMP found two differences.

4.1.2.2 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.

Refer to Section 4.1.2 for a description of the contents of files TEST1.DAT and TEST2.DAT.

Example

```
CMP>TESTDIF.DAT/CB=TEST1.DAT,TEST2.DAT [RET]
```

Produces the following output file with change bars applied to lines from two files that do not match:

```
 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
```

Notice that the change bar is applied to the first line of match (line 9).

4.1.2.3 SLP Command Input Format

You use the /SL 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.

Refer to Section 4.1.2 for a description of the contents of files TEST1.DAT and TEST2.DAT.

Example

```
CMP>TESTDIF.DAT/SL:BLS001=TEST1.DAT,TEST2.DAT [RET]
```

Uses the /SL switch to generate the following output file:

```
-6,8,/;BLS001/
45678
56789
67891
-12,,/;BLS001/
EXTRA
EXTRA
EXTRA
/
```

4.2 CMP Messages

CMP issues two kinds of messages: informational and error. Informational messages provide information on the compare operation.

Error messages describe a condition that caused CMP to terminate the processing of a command. When this occurs, CMP returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, CMP will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>CMP commandline
CMP -- message
>
```

If the command is entered in response to the CMP prompt, CMP will issue an error message and prompt you with the CMP prompt, as follows:

```
CMP>commandline
CMP -- message
CMP>
```

This section lists the CMP messages, gives a brief description of the condition that causes each message, and suggests a response to the condition.

4.2.1 Informational Message

CMP issues the following informational message:

CMP—n differences found

Explanation: CMP found n differences between the two files.

User Action: This is an informational message.

4.2.2 Error Messages

CMP issues the following error messages:

CMP—Command syntax error

Explanation: CMP found an error in the command line syntax.

User Action: Check the syntax of the command line specification, specify the correct syntax, and reenter the command line.

CMP—Error reading input file #1

Explanation: An I/O error occurred while CMP was reading the first input file.

User Action: Reenter the command line.

CMP—Error reading input file #2

Explanation: An I/O error occurred while CMP was reading the second 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. The output device may be full or bad.

User Action: Check if the output device is full or bad, and 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: Specify a legal value and reenter the command line.

CMP—Illegal switch or switch value

Explanation: An illegal switch or switch value was entered in the command line.

User Action: Use a legal switch or switch value and reenter the command line.

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. Use the correct file specification and reenter the command line.

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. Use the correct file specification and reenter the command line.

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. Use the correct file specification and reenter the command line.

CMP—Too many differences for available core

Explanation: The files were too dissimilar for CMP to fit all the differences in memory.

User Action: Remove and reinstall CMP with a larger increment. For information on the DIGITAL Command Language (DCL) command INSTALL/EXTENSION, refer to the *RSX-11M-PLUS Command Language Manual.* For more information on the Monitor Console Routine (MCR) command INSTALL, refer to the *RSX-11M-PLUS MCR Operations Manual.*

Chapter 5
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, portions of blocks, or the virtual records of a file. If you are dumping a volume, a range of logical blocks may be specified (see Sections 5.1.1 and 5.1.2). 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 RET
```

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 and by rebuilding the task. See the appropriate system generation manual and your system manager for more specific information on altering various features of DMP.

5.1 DMP Command Modes

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.

5.1.1 File Mode

In file mode, one input file is specified, and all, or a specified range, of the 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 by using the Monitor Console Routine (MCR) or DIGITAL Command Language (DCL) command MOUNT.

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 to n, where n is the total number of blocks or records in the file. Use the /BL 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 5–1.)

5.1.2 Device Mode

In device mode, only the input device is specified, and a specified range of logical blocks is dumped. The /BL 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. The volume must be mounted foreign.

5.2 DMP Command Line

You can invoke DMP by using any of the methods described in Chapter 1. The command line for DMP is shown next.

Format

[outfile][/switch[...]][=inspec][/switch[...]]

Parameters

outfile

Specifies the output file. The format for entering file specifications is as follows:

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

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. The TI and LP devices (for terminal and line printer) are also acceptable outfile specifications.

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.

switch

Specifies one of the switches listed in Table 5–1. 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 switch in the following order will be the only one executed: /LW, /WD, and /HX.

inspec

Specifies the input device and file or input device only (refer to the previous description of outfile for the format of file specifications). 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 directory.

In device mode, the equal sign and input device are required, as is the /BL switch, which specifies the range of logical blocks to be dumped.

5.2.1 DMP Switch Descriptions

DMP switch specifications consist of a slash (/) followed by a switch name, which is optionally followed by a value. The value is separated from the switch by a colon (:). DMP functions are implemented by the switches listed in Table 5–1.

Table 5–1: DMP Switches

Switch Format	Function
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 (^), which is followed by the alphabetic character corresponding to the character code plus 100. For example, bell (code 7) is printed as ^G (code 107). Lowercase letters (140–177) are printed as a percent sign (%), which is followed by the corresponding uppercase letter (character code minus 40), unless the /LC switch is specified. Note that the /AS and /OCT switches are mutually exclusive when dumping bytes.

Table 5–1 (Cont.): DMP Switches

Switch Format	Function
/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 an effective block number. This switch is useful for specifying block numbers that exceed 16 bits. For example: `DMP>/BA:1:0` RET The previous command line specifies that all future block numbers will be relative to 65536_{10} (200000_8). `DMP>/BA:0:0` RET The previous command line 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 alone 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, which is 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 the /HD switch). The /BL switch and the /RC switch are mutually exclusive. The /BL 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. The values given with the /BL switch are not used to position the tape. 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 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.

Table 5–1 (Cont.): DMP Switches

Switch Format	Function
/DENS:n	Specifies the density of an input magnetic tape with 800- or 1600-bpi capability when DMP is in device mode only. The value for n can be 800 and 1600. 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 *RSX–11M–PLUS MCR Operations Manual*, the *RSX–11M–PLUS Command Language Manual*, the *Micro/RSX User's Guide, Volume 1*, and the *Micro/RSX User's Guide, Volume 2* for descriptions of the MOUNT command and its /DENS switch.)
/FI:n:sequence	In file mode, the file number can be used instead of a file name as a file specification for input.
/HD[:F] /HD: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. For example: `DMP>TI:=JMF.DAT/HD/BL:5:6` RET The previous command line dumps the header of JMF.DAT in header format and virtual blocks 5 and 6 in octal format. In addition, the /HD switch has two options: the default, F, causes a Files–11 formatted dump of the header, and 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 the following: `DMP>TI:=JMF.DAT/HD/BL:0` RET File headers are described in the *RSX–11M–PLUS and Micro/RSX I/O Operations Reference Manual*.
/HF	Specifies the format for data blocks that have the Files–11 header structure. Other blocks are output as a data dump in the format selected by switches such as /AS and /BY in default octal words. For example: `DMP>HEAD.LST=[0,0]INDEXF.SYS/HF` RET The previous command line 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.)

Table 5–1 (Cont.): DMP Switches

Switch Format	Function

/ID	Causes DMP's version to be identified. This switch may be specified on a command line by itself at any time. For example:
	
	`DMP>/ID` RET
	`DMP--DMP version M07.1B (ANSI)`
	
	For DMP built with ANSLIB, the response is as follows:
	
	`DMP--DMP version M07.1B`

/LB	Requests logical block information for a file and displays the starting block number and a contiguous or noncontiguous indication for the file. For example: `DMP>TI:=DKO:RICKSFILE.DAT;3/LB` RET `Starting block number = 0,135163 C` The previous command line indicates that the file RICKSFILE.DAT, version 3, is a contiguous file starting at block number 0,135163. (See the /BA switch for block number description.)
/LC	Specifies that the data should be dumped in lowercase letters. This switch can only be used if the output device has lowercase capability.
/LIM:n:m	Specifies the range of bytes n–m of each record or block to be dumped. The /OCT switch is still the default if no format switches are specified.
/LW	Specifies that the data be dumped in hexadecimal doubleword format.
/MD[:n]	Specifies a memory dump and allows control of line numbers. Line numbers are normally reset to 0 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.

Table 5–1 (Cont.): DMP Switches

Switch Format	Function
/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 bytes allocated to it initially. To increase the amount of space available, use the MCR command INSTALL with the /INC switch or the DCL command INSTALL /EXTENSION:n. For example, to dump a file with 1024-byte records, you must specify /INC=256_{10} (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 re-position a tape at the beginning-of-tape (BOT).
/R5	Specifies that data be dumped in Radix–50-format words.
/SB:n /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 /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.

5.3 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).

5.3.1 Multiple Format Dump

The command line shown in Example 5–1 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.

```
IDENTIFICATION AREA
   I.FNAM,
   I.FTYP,
   I.FVER          DSC       .MAC;1
   I.RVNO          1
   I.RVDT          13-OCT-86
   I.RVTI          09:52:46
   I.CRDT          13-OCT-86
   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
```

5.4 DMP Error Messages

Error messages describe a condition that caused DMP to terminate the processing of a command. When this occurs, DMP returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, DMP will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>DMP commandline
DMP -- message
>
```

If the command is entered in response to the DMP prompt, DMP will issue a error message and prompt you with the DMP prompt, as follows:

```
DMP>commandline
DMP -- message
DMP>
```

The error messages for DMP are as follows:

DMP—Bad device name

Explanation: An incorrect device name was entered in a file specification.

User Action: Specify the correct device and reenter the command line.

DMP—Block switch required in logical block mode

Explanation: The /BL switch must be specified.

User Action: Specify the /BL switch and reenter the command line.

DMP—Cannot find input file

Explanation: The requested file cannot be located in the specified directory.

User Action: Specify the correct file name and directory and reenter the command line.

DMP—Command syntax error

Explanation: A command line was entered in a format that does not conform to syntax rules.

User Action: Specify the correct syntax and reenter the command line.

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 User Identification Code (UIC) that has read access privileges to the file.

DMP—Illegal density value

Explanation: A value other than 800 or 1600 bpi was specified with the /DENS switch.

User Action: Specify the correct density and reenter the command line.

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: Specify the correct switch and reenter the command line.

DMP—Illegal use of /RC switch

Explanation: The /RC switch can be used only in file mode (see Section 5.1.1).

User Action: Specify a file name and reenter the command line.

DMP—Illegal value on /HD switch

Explanation: An option was entered other than F (the default) or U (the unformatted octal dump) for the /HD switch.

User Action: Specify the correct option and reenter the command line.

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; then, reenter the command line.

DMP—No input file specified

Explanation: A command line was entered with no input file specification.

User Action: Specify an input file and reenter the command line.

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: Specify only one input file specification and one output file specification, and reenter the command line without wildcard specifications.

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; then, 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; then, reenter the command line.

Chapter 6
Disk Save and Compress Utility (DSC)

The Disk Save and Compress Utility (DSC) copies a Files–11 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.

6.1 DSC Operations

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 stand alone. The standalone version of DSC is included in BRUSYS.SYS (refer to Chapter 3 for more information).

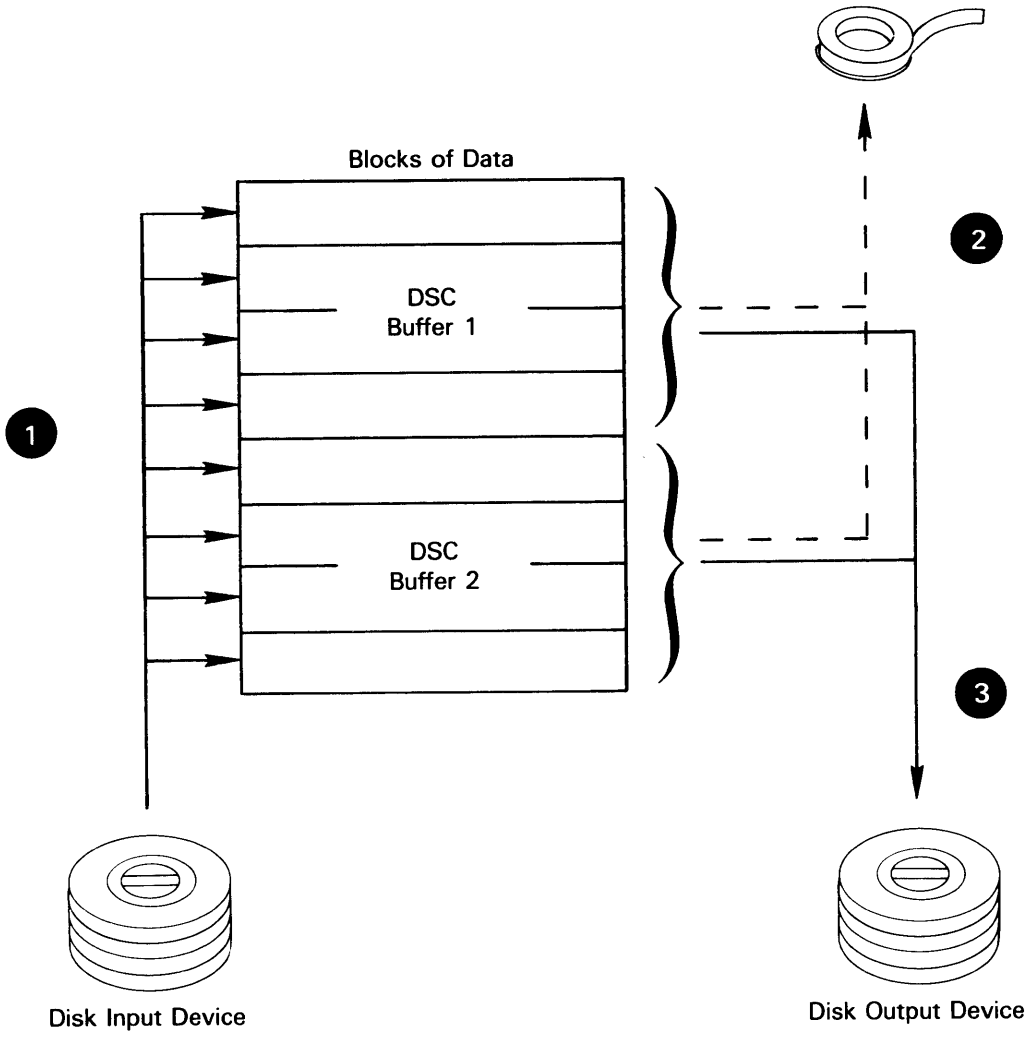
After a DSC copy operation, individual files are written in available contiguous blocks and the blocks available for new files are located in a contiguous area at the end of the new volume. If the contents of one disk are transferred to a disk with a larger capacity, the new disk takes on the attributes 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 6–1 and 6–2.) Each buffer normally is large enough to contain four disk blocks and a 16-byte buffer prefix. However, the Block Factor switch (/BL) in a DSC command line allows you to increase the number of blocks in each buffer, up to the maximum space available for DSC on your system.

In a disk-to-disk copy operation, DSC performs the following tasks:

1. Copies data from disk to a DSC buffer
2. Copies data from the DSC buffer to another disk

Figure 6–1: Data Transfer for DSC Copy Operation



❶ DSC reads eight (default) or more blocks of data from the disk input device to two buffers.

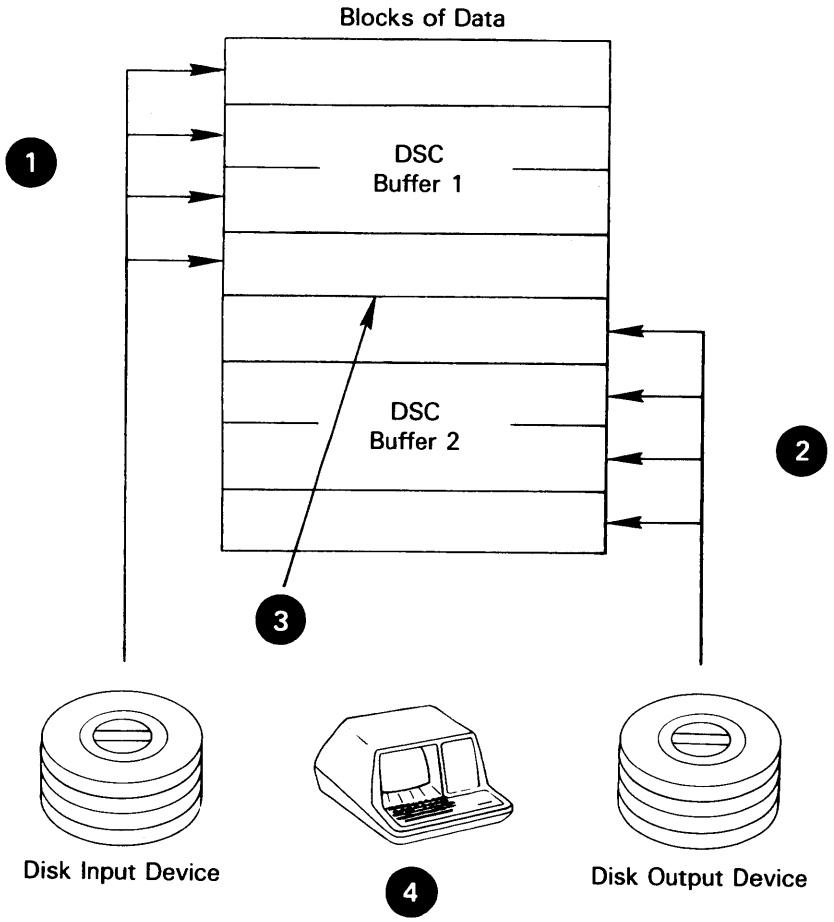
❷ In disk-to-tape copy operations, DSC writes data from the buffers to magnetic tape.

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 6–2: Data Transfer for DSC Compare Operation



❶ DSC reads four blocks (default) of data from the disk input device to a buffer.

❷ DSC reads four blocks (default) of data from the disk output device to the second buffer.

❸ DSC compares the contents of the two buffers.

❹ DSC prints the differences on your terminal.

DSC repeats steps ❶ to ❹ until it has compared the entire device.

ZK-183-81

In a disk-to-tape and tape-to-disk operation, DSC performs the following tasks:

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.

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-PLUS system image by the Monitor Console Routine (MCR) command SAVE), 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.

6.2 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. DSC requires volumes to be mounted with foreign characteristics.

Table 6-1 lists the devices that can be used with DSC operations.

Table 6-1: DSC-Supported Devices

Abbreviation	Type	Class
DB	RP04/RP05/RP06 disk pack	Block structured
DD	TU58 cassette (DECtape II)	Block structured
DF	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	RH70/RS03/RS04 and RH70/RS03 fixed-head disk	Block structured
DT	TU56 DECtape	Block structured

Table 6–1 (Cont.): DSC-Supported Devices

Abbreviation	Type	Class
DU	RA80 fixed media disk	Block structured
DX	RX01 diskette	Block structured
DY	RX02 diskette	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 TS03 9-track magnetic tape	Tape

6.3 DSC Command Line

You can invoke DSC by any of the methods described in Chapter 1 of this manual. The command line for DSC is shown next.

Format

outdev[,...][filelabel1][/switch[...]]=indev[,...][filelabel2][/switch[...]]

Parameters

outdev

Specifies the physical volume or volumes to which data is copied. The format for outdev is as follows:

`dd[nn]:`

The parameter dd specifies the American Standard Code for Information Interchange (ASCII) characters for the volume abbreviation, and [nn] is an optional 1- or 2-digit octal unit number for the volume. 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 a DSC operation.

DSC ignores multiple disk specifications.

filelabel1

Identifies the output disk's Volume ID, the tape file, or the 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 you copy 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

Specifies one or more of the optional switches described in Table 6–2. (For detailed descriptions of the switches, refer to Section 6.5.

indev

Specifies the physical volume or volumes, in the same format as outdev, from which data is copied.

filelabel2

Identifies the DSC-created tape file that is being transferred to disk or is being 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.

Table 6–2: 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	/BAD= { MAN | NOAUTO | MAN:NOAUTO | OVR | MAN:OVR }	Allows manual entry of bad block locations; it can supplement, override, or ignore the disk's bad block file.
Block Factor	/BL=n /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= /DENS: { 1600 | 800:1600 }	Overrides the DSC default storage density for magnetic tapes of 800 bpi.
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.

6.4 File Labels

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 nine 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)

Examples

```
DSC>MM01:,MM02:SYSFILE=DB1: RET
```

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, use of the Rewind switch (/RW) 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).

```
DSC>DB1:=MM01:,MM02:SYSFILE RET
```

Because the /RW switch was not specified on the input side, 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 /RE switch, the tape may or may not be positioned at the beginning before DSC begins its operation.

6.5 DSC Switch Descriptions

DSC commands can contain file labels and switches. Some switches also use options to specify values. This section describes in detail the functions of the DSC switches. Table 6–2 summarizes the DSC switches and options.

6.5.1 Append

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 /AP 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.

When you use the /AP 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
```

Example

```
DSC>MM01:,MM:SYSFILE/RW/AP=DX1: RET
```

Appends the contents of DX1 to the last DSC-created file already present on the first output volume specified, MM01. Because the Rewind switch (/RW) is specified (see Section 6.5.6), 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 /AP switch with output tape volumes. Any other use of the switch causes DSC to generate an error message and terminate the operation.

6.5.2 Bad Block

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 perform the following tasks:

- Supplement the output disk bad block file with manually entered bad block data.
- Ignore or override the bad block file on last track (manufacturer's list of bad blocks) or non-last-track devices.
- 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 Section 6.8.1, Message 59).

Format

/BAD=option

Parameter

option

Specifies one of the five possible options as follows:

MAN	Supplements the bad block file BADBLK.SYS with manually entered bad block data. This option may be combined with either NOAUTO or OVR.
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 the MANUAL option.
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 the bad block file BADBLK.SYS. This option is valid only on last-track devices. This option may also be combined with the MAN option.
MAN:OVR	Allows manual entry of bad block data to the bad block file BADBLK.SYS.

When you specify the MAN, MAN:NOAUTO, or MAN:OVR option 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 in the format shown next.

Format

n:m

Parameters

n

Specifies the logical block number (LBN), in octal, of the initial bad block in the group.

m

Specifies 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 by 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, press the RETURN key after the LBN(S)= prompt.

When you have entered all the bad blocks and terminated the entry process, DSC begins the data transfer.

Examples

```
DSC>DB1:/BAD=MAN:NOAUTO=MM01:,MM02:SYSFILE/RW [RET]
DSC>LBN(S)=702:7 [TAB]644:2 [RET]
DSC>LBN(S)=4057,5001:3 [RET]
DSC>LBN(S)= [RET]
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			

```
DSC>DB1:/BAD=NOAUTO=DBO: [RET]
```

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 the Bad Block Locator program (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.

6.5.2.1 Obtaining Bad Block Information

You can obtain bad block information in the following two ways:

- By running the Bad Block Locator program (BAD), described in this manual
- By running the DIGITAL Field Service standalone diagnostic

The BAD utility automatically provides bad block information and creates a bad block file that DSC can use.

The Field Service standalone 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 LBNs before DSC can use it.

6.5.2.2 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)+tracknumber)*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	350
Track Number	16	14
Sector Number	13	11

The LBN for the example is calculated as follows:

```
(((350.*19.)+14.)*22.)+11.=146619.
```

6.5.3 Block Factor

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.

Format

/BL=n

or

/BL:n

The value of n can be any positive integer, decimal or octal, less than or equal to the maximum block factor available to DSC. This maximum depends on the amount of memory DSC can access under the system configuration. (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.

Example

```
DSC>DB1:/BL=11=DB0: RET
```

or

```
DSC>DB1:/BL:11=DB0: RET
```

Attempts to increase the number of blocks in each of DSC's 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.

6.5.3.1 System-Dependent Requirements for the /BL Switch

DSC expands automatically if memory is available. (See the *RSX-11M-PLUS System Generation and Installation Guide* for details on building DSC with additional memory.)

For the RSX-11M-PLUS operating system, the default blocking factor is 4, the maximum DSC size is 32K words, and the maximum blocking factor is 32_{10}.

6.5.4 Compare

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 disk specifications are not. The /CMP 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 /CMP switch performs only comparison operations; no copy operation is involved.

To perform both a copy and compare operation, use the DSC Verify switch (/VE) (see Section 6.5.7).

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 6.5.6. 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 by using the next volume of the set.

6.5.5 Density

The Density switch (/DENS), with its two options, allows you to override the DSC default storage density of 800 bpi for TU16, TE16, TU77, and TU45 tape drives. The following two sections discuss these options. Although you can use other tape drives with DSC, only these drives can support the /DENS switch.

Format

/DENS=option

or

/DENS:option

Parameter

option

Specifies one of the following two options:

1600
800:1600

The 1600 option creates magnetic tapes at 1600 bpi density and the 800:1600 option (the Split Density option) creates magnetic tapes with volume headers at 800 bpi and the rest of the tape at 1600 bpi.

Note that the 1600 option is valid with TU16, TU77, TE16, or TU45 drives. The 800:1600 option is valid with TU16 or TU45 drives only when they are not controlled by the TM03 formatter.

You do not have to specify the /DENS 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.

If you specify the /DENS switch with a disk, DSC issues an error message and halts the operation.

If you specify the /DENS 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 the 800:1600 option. The TS11 (TS04) ignores the /DENS switch; therefore, do not use it with these devices.

6.5.5.1 1600 Option

The 1600-bpi option directs the TU16, TE16, TU77, 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.

Example

```
DSC>MM01:,MM02:SYSFILE/RW/AP/DENS=1600=DB1: RET
```

Writes MM01 and MM02 at 1600-bpi density.

6.5.5.2 Split Density Option

The Split Density option (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
```

Example

```
DSC>MM01:,MM02:SYSFILE/RW/DENS=800:1600=DB1: RET
```

Records the first two blocks of the first volume at 800 bpi and the remainder of the file at 1600 bpi.

Note

Magnetic tapes created by using the Split Density option do not comply with American National Standard Institute (ANSI) 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.

6.5.6 Rewind

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 6–3).

The /RW switch can be used only with magnetic tapes. If you use it with any other volume, DSC prints an error message.

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 off line. If, however, the current file ends on the first or only tape of a set, the tape is positioned to read the next file on the input tape. The /RW switch only rewinds tapes at the beginning of a DSC operation.

If you enter the /RW switch as part of the output specification, DSC 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 end-of-file (EOF) it finds on that volume.

After the output tape has been rewound, DSC determines if the tape is positioned at 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 /AP 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 6.5.1); otherwise, the copy operation will overwrite any data previously stored on the tape.

Table 6–3 summarizes the use of the /RW switch with various DSC operations, with and without a file label.

Table 6–3: The Rewind Switch and DSC Operations

Switch	Specification	File Label	Function
/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.

Table 6–3 (Cont.): The Rewind Switch and DSC Operations

Switch	Specification	File Label	Function
/RW	Output	With File labels specified when tape is the output volume are ignored when the tape is restored to disk.	DSC writes data, starting at the beginning of the first tape volume, unless the /AP switch is specified.
/-RW	Output		If the tape is not at BOT, DSC writes data, beginning after the last EOF 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 off line.

Note

When you refer to tapes after your system is booted, you must use the /RW switch. If you do not use the switch, the tape driver will return an error message.

Examples

```
DSC>MM01:SYSFILE/RW=DB1: RET
```

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.

```
DSC>DB1:=MM02:,MM01:SYSFILE/RW RET
```

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.

6.5.7 Verify

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 /VE or /CMP 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.

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 the end-of-volume (EOV) or end-of-file (EOF) on any volume of a tape set during a copy operation, it re-positions 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.

6.6 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.

Example

```
DSC>DB1:=DB2: RET
```

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 update not only involves collapsing retrieval pointers, but it 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 logical block number (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.

6.7 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.

6.7.1 Data Transfer from Disk

After you enter a DSC command line and specify 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 performs the following tasks:

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.

6.7.2 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 by 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 on line. You can specify multiple tape drives in the following way:

```
DSC>ddnn(0):,ddnn(1):,...ddnn(7):[filelabel]=indev[,...]
```

You have the option of entering a file label in this command line after you specify the last device. While you can specify only one type of tape drive, either MM-, MT-, or MS-type in a single DSC command line, you can specify up to eight individual drives on the output side of the command line; 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.

Example

```
DSC>MM0:,MM1:,MM4:,MM2:SYSFILE=DB1: RET
```

Shows the order of replacement used until an end-of-file is reached as follows:

```
MM0: MM1: MM4: MM2: MM0: MM1: MM4: MM2: ...
```

In summary, to transfer data to tape, DSC performs the following tasks:

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.

6.7.3 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 6.7.2. 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 performs the following tasks:

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.

6.7.4 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 2).

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 or disks 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. It 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 when the contents of a small disk (such as an RK05) are restored to a large disk (such as an RP04).

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 (the maximum is 256_{10}), 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 data previously occupied the disk.

In summary, to transfer data to a disk, DSC performs the following tasks:

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.

6.8 DSC Messages

DSC notifies you of fatal error conditions as well as less serious error conditions that could cause difficulties in DSC operations. Each message generated by DSC is identified by a numeric code and has the following prefix:

```
DSC --
```

DSC messages are displayed on your terminal in either a long or short form. Online DSC displays the long form, and standalone DSC displays the short form. You can determine the meaning of the short form messages from the number provided with the message. Use the number to find the long form message in Section 6.8.1. The text accompanying the long form message of that number explains the error. Also, in order to identify the error, the command line or the command line segment may be displayed after the error text.

Table 6–4 is a quick reference to the single-letter codes used in general error messages and in I/O error messages (Section 6.8.2).

Table 6–4: 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 5) with the File switch (/FI) and the Header switch (/HD) to obtain the name of the file and other information contained in the file header.

Example

```
DSC>TI:=devid:/FI:x:y/HD/BL:0 RET
```

Outputs to a terminal the header or headers of file x, y. The variables x and y, displayed in certain DSC error messages, represent the file number and file sequence number respectively.

6.8.1 DSC Error Messages

DSC error messages may or may not be fatal. Fatal error messages are displayed in the following format:

```
DSC -- *FATAL* message
```

Error messages describe a condition that caused DSC to terminate the processing of a command. When this occurs, DSC returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, DSC will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>DSC commandline
DSC -- message
>
```

If the command is entered in response to the DSC prompt, DSC will issue an error message and prompt you with the DSC prompt, as follows:

```
DSC>commandline
DSC -- message
DSC>
```

The following are the general error 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: Specify only one type of tape drive and reenter the command line.

4 Illegal switch

Explanation: The command line was entered with a switch that cannot be used with that command line.

User Action: Specify only correct switches and reenter the command line.

5 File label too long

Explanation: A file label consisting of more than 12_{10} alphanumeric characters was specified.

User Action: Specify a shorter file label and reenter the command line.

6 Syntax error

Explanation: An error occurred in the command line format.

User Action: Specify the correct order and reenter the command line.

7 Dup. dev. name;

Explanation: The same device was entered more than once in the command line.

User Action: Specify each device only once and reenter the command line.

8 Too many dev's

Explanation: More than the legal number of devices were specified on one side of the command line.

User Action: Specify no more than eight devices per side and reenter the command line.

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. Specify an appropriate device and 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. Specify an appropriate device and 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 and reenter the correct command line.

12 Bad block count too large

Explanation: There are too many groups of bad blocks (the maximum is 125_{10}) 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. Specify the correct block numbers and 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 and reenter a command line to begin a new tape set.

16 Output tape ddnn: not only reel in set

Explanation: The /AP switch was used with a tape that was not the first tape of a set created by DSC.

User Action: Change tapes and reenter the command line (see message 15).

17 Tape ddnn: not ANSI format

Output Tape

Explanation: If the medium is an output tape, the /AP 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 ddnn: not DSC tape

Explanation: An /AP switch was specified with a tape that was not created by DSC.

User Action: Reenter the command line and either omit the /AP 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: Specify input tapes in the proper order and reenter the command line.

21 Failed to find home block

Explanation: DSC failed to find the home block on the input disk. 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 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 and 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 and 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: Specify a new drive and reenter the command line.

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 and reenter the command line.

32 Input device ddnn: File ID, y, 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 tried again on a different drive.

33 Input device ddnn: File ID y, 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 user action is required.

34 Input device ddnn: File ID y, 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 Files–11 Structure Level 1). DSC only copies Files–11 Structure Level 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, y file number check

Explanation: An incorrect file header was read from the disk and caused the specified file to be lost.

User Action: Reenter the command line.

36 Input device ddnn: File ID y, 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, 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, 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 - number

Explanation: An internal error occurred, usually the result of a system overload.

User Action: Reenter the command line.

40 I/O error F on ddnn:
... (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 user 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, y VBN z, z

Explanation: An I/O error occurred that resulted in bad data being read from the specified virtual block number (VBN) 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 required.

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 /VE switch).

User Action: This is an informational message only. No user action is required.

46 Resume copying

Explanation: The verify operation (specified with the /VE switch) is complete and DSC is continuing the copy operation (if there is more material to copy).

User Action: This is an informational message only. No user action is required.

47 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: Specify a less fragmented output disk and reenter the command line.

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 or headers 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 the Peripheral Interchange Program (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 file, other than the original input file number, therefore the files will not compare when you use the /CMP 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 bit map 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 it 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 and 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 and reenter the command line.

56 Output disk ddnn: is not bootable

Explanation: LBN 0, the bootstrap block, of the specified disk or tape is bad. (This message will always be preceded by message 84 and/or 86 indicating the reason for the error.)

User Action: This is a warning only. No user 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 with another disk specified.

58 Bad block file full

Explanation: Too many bad blocks exist on the output disk.

User Action: Replace the disk and 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 with a new disk specified.

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 expected p,p,p found y

Explanation: The tape position was lost when DSC read the virtual block number (VBN) 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 was lost.

User Action: Re-create the tape or reenter the command line with a different tape drive specified.

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 y, y, y - 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: Specify a larger output disk and reenter the command line.

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: Specify a different tape drive and reenter the command line.

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 /VE 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. Reenter them in proper order after you receive 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, and it indicates that the input tape was either read incorrectly or was 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, and then reenter the command line with the correct tape and file label specified.

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 operation 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 bit map files from the specified input tape could not be restored.

User Action: Specify a different input tape drive and reenter the command line.

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 that states that the output disk will not be bootable.

User Action: This is a warning only. No user 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 preceded by message 85, which indicates the reason for the error. Other messages concerning the bootability of the output disk may precede 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 details.

6.8.2 DSC I/O Messages

In DSC, I/O errors are prefixed by DSC— and 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.

User Action: Reenter the command line.

Bad block on device

Explanation: A bad area was encountered on the device, which results in a block that cannot be read or written without error.

User Action: Reenter the command line.

Block check or CRC error

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.

User Action: Reenter the command line.

Device not ready

Explanation: The device is not ready or is not up to speed.

User Action: Reenter the command line.

Device offline

Explanation: The device is not in the system.

User Action: Check the device and the device specification in the command line, and 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: 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 by using the /RW switch in a DSC command line.

Chapter 7
Line Text Editor (EDI)

EDI is a line-oriented editor that allows you to create and modify text files. EDI operates on most American Standard Code for Information Interchange (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 eight categories:

Basic commands	Allow you to create, modify, and exit files.
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	Use, store, recall, and define 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 EDI Commands: Function Summary (Section 7.7) and EDI Commands: Detailed Reference Summary (Section 7.8). Restrictions, system device considerations, and error messages for the commands are discussed in Section 7.9 and Sections 7.11 to 7.11.4.

7.1 EDI Command Line

You can invoke EDI by using any of the methods described in Chapter 1. The command line for EDI is shown next.

Format

filespec

Parameter

filespec

Specifies a file specification in the following format.

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

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:

```
[CREATING NEW FILE]
input
```

EDI then enters input mode. (EDI control modes are described in Section 7.2.)

If the message [FILE DOES NOT EXIST] is printed, it means that the directory corresponding to the User Identification Code (UIC) is nonexistent.

EDI does not accept indirect command file specifications.

If you specify an existing file name, EDI prints the following:

```
[OOOnn LINES READ IN]
[PAGE 0]
*
```

It then 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 the Insert command. To manipulate the first line of text, on the other hand, you must issue the Next command to make that line available.

7.1.1 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 7–1.

Table 7–1: EDI Default File Specifications

Element	Default Value for Input File	Default Value for Output File
ddnn	SY0	Same as input device
directory	Directory under which EDI is currently running	Same as input directory
filename	No default—must be specified	Same as input file name
type	Unspecified	Same as input file type
version	Latest version	Latest version + 1

7.2 EDI Control Modes: Edit and Input

EDI runs in two control modes:

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

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

In edit mode, EDI issues an asterisk (*) 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, which signifies edit mode.

7.3 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 as follows:

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 previous blocks 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."

7.3.1 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 the Block Off command and is terminated by issuing the 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 by 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 the Top or Top of File command (refer to Sections 7.8.53 and 7.8.54). Top of File causes the input and output files to be closed and the output file to become the new input file. Top of File also ends line-by-line mode.

7.3.2 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 the Top of File 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 the Top of File command.

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

```
buffersize/132
```

A block is the number of lines read into the buffer by the Renew or Read command. This number is one of the following:

- Specified with the Size command (default is 38 lines if the Size command is not issued)
- 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 the Size command was not issued)

When the current line pointer reaches the 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 the Top command.

Table 7–2 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 7–2: 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 forwards and backwards 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.

7.3.2.1 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 (EOF) marker).

Two commands are provided to handle paged text: the Form Feed command, which defines a page boundary by inserting a form feed, and the Page command, 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 the 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 EOF, EDI increments the page count to 1.

7.4 Text Files

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

7.4.1 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.

7.4.2 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.

7.5 Terminal Conventions

RSX-11M-PLUS and EDI provide terminal keyboard functions that allow you to perform the following tasks:

- 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 command line interpreter (CLI) (for example, the Monitor Console Routine (MCR) or DIGITAL Command Language (DCL)).

[1] To delete the input file, use the Close & Delete command or the Exit & Delete command, or use PIP (see Chapter 12).

7.5.1 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, pressing the DELETE key does not affect previously prepared text.

When you press the DELETE key, it is echoed as a backslash (\) followed by the previously typed character. Pressing each successive DELETE key results in the echo of an earlier typed character. When you type the first new character, it is echoed as a backslash (closing the DELETE key sequence) followed by the typed character. For example:

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

For some video terminals, the DELETE key works in a more obvious way. Pressing the DELETE key causes the cursor to backspace, erasing the previous character. Your video terminal may work this way if a certain option was selected when your system was generated.

Another useful system generation option is CTRL/R. It is entered by pressing the CTRL key while simultaneously pressing the R key. On some terminals, pressing CTRL/R echoes as ^R. Your system responds to your pressing CTRL/R by printing the incomplete input line 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
```

7.5.2 Line Erase—CTRL/U

Pressing CTRL/U deletes the line being input, if typed before the line is terminated with the RETURN key. It is entered by pressing the CTRL key and the U key simultaneously. On some terminals, pressing CTRL/U echoes as ^U and is followed by a return and line feed.

7.5.3 RETURN Key

The RETURN key has the following effects, depending on how it is used:

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

7.5.4 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. ESCAPE can be used this way only in block mode, not in line-by-line mode.

When EDI is in input mode, pressing the ESCAPE key produces the same results as pressing the RETURN key: it terminates the line. If ESCAPE is the first character of an input line, EDI exits from input mode.

7.5.5 Terminating EDI—CTRL/Z

Pressing 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. Press CTRL/Z to terminate EDI in edit mode and input mode. Pressing CTRL/Z erases your last input line if you enter the command as a line terminator.

7.6 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.

7.6.1 Use of Asterisk

The asterisk (*) can be used in place of any numeric argument. It evaluates to $32{,}767_{10}$.

Example

```
P * RET
```

Prints the remainder of the block buffer or file.

7.6.2 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

Indicates any string that starts with string1, continues with any number of intervening characters, and ends with the first occurrence of string2.

Case 2

...string

Indicates any string that starts at the beginning of the current line and ends with the first occurrence of string.

Case 3

string...

Indicates the first string that starts with string and ends at the end of the current line.

Case 4

...

Indicates 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 as follows:

THIS IS A SAMPLE OF SPECIAL STRING CONSTRUCTS.

Case 1

```
*C /S A...E 0/S AN EXAMPLE 0 RET
```

Results in the following:

```
THIS IS AN EXAMPLE OF SPECIAL STRING CONSTRUCTS.
```

Case 2

```
*C /...SPEC/HERE IS AN EXAMPLE OF SPEC RET
```

Results in the following:

```
HERE IS AN EXAMPLE OF SPECIAL STRING CONSTRUCTS.
```

Case 3

```
*C /STRING.../EDI STRING CONSTRUCTS. RET
```

Results in the following:

```
HERE IS A SAMPLE OF SPECIAL EDI STRING CONSTRUCTS.
```

Case 4

```
*C /.../EXAMPLES OF SPECIAL EDI CONSTRUCTS. RET
```

Results in the following:

```
EXAMPLES OF SPECIAL EDI CONSTRUCTS.
```

7.6.3 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 command format specification for the Verify command is as follows:

```
V
```

However, the following truncations are valid for the Verify command:

```
V
VE
VER
VERI
VERIF
VERIFY
```

7.7 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." The following sections contain summaries of the command categories.

7.7.1 Basic Commands

The basic EDI commands listed in Table 7–3 allow you to create a file; to modify a file by adding, deleting, or changing its contents; and to exit a file after the desired operations have been completed. Basic commands are the most important EDI commands.

Table 7–3: Basic EDI Commands

Command	Format	Function
Add	A string	Appends string to current line.
Add & Print	AP string	Appends string to current line and prints resulting line.
ALTMODE key	Press the ALTMODE key	Prints previous line, makes it the new current line and exits from input mode (block mode only). Same as NP-1.
Bottom	BO	Moves 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]C /string1/string2[/]	Replaces string1 with string2 n times in the current line.
CTRL/Z	Press CTRL/Z	Closes files and terminates editing session.
Delete	D [n] D -n	Deletes current line and n-1 lines if n is positive; deletes n lines preceding the current line if n is negative (-n). Operates in block mode only.

Table 7–3 (Cont.): Basic EDI Commands

Command	Format	Function
Delete & Print	DP [n] DP -n	Same as the Delete command, except new current line is printed.
ESCAPE key	Press the ESCAPE key	Prints previous line, makes it the new current line, and exits from input mode (block mode only). Same as NP-1.
Exit	EX [filespec]	Closes files, renames output file, and terminates editing session.
Insert	I [string]	Enters the string immediately following the current line. If no string is specified, EDI enters input mode.
Locate	[n]L [string]	Locates nth occurrence of string. In block mode, search stops at end of current block.
Next	N [n] Next -n	Establishes new current line n lines away from current line.
Next & Print	NP [n] NP -n	Establishes and prints new current line.
Print	P [n]	Prints current line and the next n-1 lines. The last printed line is the new current line.
Renew	REN [n]	Writes current block to output file and reads new block n from input file (block mode only).
RETURN key	Press the RETURN key	Prints the next line, makes it new current line, and exits from input mode. Same as NP+1.
Retype	R [string]	Replaces current line with string or deletes current line if string is not given.
Top	T	Moves the current line pointer to the top of the current block (in block mode) or top of file (in line-by-line mode). The Top command creates a new version of the file each time it is invoked in line-by-line mode.
Top of File	TOF	Returns line pointer to top of input file and saves 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.
Type	TY [n]	Prints current line and the next n-1 lines. The last printed line is the new current line.

7.7.2 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 and Change commands. Setup commands are listed in Table 7–4.

Table 7–4: EDI Setup Commands

Command	Format	Function
Block On/Off	BL [ON] BL OFF	Switches text access modes.
Concatenation Character	CC [character]	Changes concatenation character to specified character (default is the ampersand (&)).
Open Secondary	OP filespec	Opens specified secondary file.
Output On/Off	OU [ON] OU OFF	Continues or discontinues transfer to output file (line-by-line mode).
Select Primary	SP	Reestablishes primary file as input file.
Select Secondary	SS	Selects opened secondary file as input file.
Size	SIZE n	Specifies maximum number of lines to be read into block buffer.
Tab On/Off	TA [ON] TA OFF	Turns automatic tabbing on or off.
Uppercase On/Off	UC [ON] UC OFF	Enables or disables conversion of lowercase letters entered from terminal to uppercase letters.
Verify On/Off	V [ON] V OFF	Selects whether locator and change commands are verified.

7.7.3 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 7–5.

The locator commands allow you to perform the following tasks:

- 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, pressing the RETURN, ESCAPE, or ALTMODE key acts to relocate the line pointer. Pressing the RETURN key moves the pointer to the next line. Pressing the ESCAPE or ALTMODE key moves the line pointer back one line (in block mode only). In each case, the line is printed.

If the Verify On command is in effect, the located line is printed after the Bottom, End, Find, Page Find, Page Locate, or Search & Change command.

Table 7–5: EDI Locator Commands

Command	Format	Function
ALTMODE key	Press the ALTMODE key	Prints previous line, makes it new current line, or exits from input mode (block mode only).
Begin Top	B T	Sets 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 End	BO E	Sets current line to last line in file or block buffer. The commands are equivalent.
ESCAPE key	Press the ESCAPE key	Prints previous line, makes it new current line, or exits from input mode (block mode only).
Find	[n]F [string]	Searches current block or input file, beginning at line following current line for the nth occurrence of string. String must begin in column 1. Sets line pointer to located line.
Locate	[n]L [string]	Locates nth occurrence of string. In block mode, search stops at end-of-block.
Next	N [n] N -n	Establishes new current line n lines away from current line.
Next & Print	NP [n] NP -n	Establishes and prints new current line.
Page	PAG n	Enters block mode. Reads page n into block buffer. If n is less than current page number, issue a TOF command first (block mode only). Pages are delimited by form-feed characters.
Page Find	[n]PFind string	Searches successive blocks for the nth occurrence of string (block mode only). String must start in column 1.
Page Locate	[n]PL string	Searches successive blocks for the nth occurrence of string (block mode only). String may occur anywhere in line.
RETURN key	Press the RETURN key	Prints the next line, makes it the current line, exits from input mode.
Search & Change	SC /string1/string2[/]	Locates string1 and replaces it with string2.

7.7.4 Text Modification and Manipulation Commands

The text modification and manipulation commands enable you to modify text. Table 7–6 lists these commands.

Table 7–6: EDI Text Modification and Manipulation Commands

Command	Format	Function
Add	A string	Appends string to current line.
Add & Print	AP string	Appends string to the current line and prints resulting line.
Change	[n]C /string1/string2[/]	Replaces string1 with string2 in the current line n times.
Delete	D [n] D -n	Deletes current line and n-1 lines if n is positive; deletes n lines preceding current line if n is negative (-n operates in block mode only).
Delete & Print	DP [n] DP -n	Same as the Delete command except new current line is printed.
Erase	ERASE [n]	Erases the current line if in line-by-line mode. Erases the current block buffer and the next n-1 blocks if in block mode.
Form Feed	FF	Inserts form feed into block buffer (used to delimit a page).
Insert	I [string]	Enters string following current line or enters input mode if string is not specified.
Line Change	[n]LC /string1[/string2][/]	Changes all occurrences of string1 in current line (and n-1 lines) to string2.
Overlay	O [n]	Deletes n lines, enters input mode, and inserts new line or lines as typed in place of original line or lines.
Paste	PA /string1/string2[/]	Searches all remaining lines in file or block buffer for string1 and replaces with string2.
Retype	R [string]	Replaces the current line with string or deletes the current line if string is not given.
Top of File	TOF	Returns to the top of the input file and saves all pages previously edited.
Unsave	UNS [filespec]	Inserts all lines from specified file following current line. If filespec is not given, SAVE.TMP is used.

7.7.5 Macro Commands

The Macro 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 7–7 lists the macro commands.

Table 7–7: EDI Macro Commands

Command	Format	Function
Macro	MACRO x definition	Defines macro number x. The value x may be 1, 2, or 3.
Macro Call	MC	Retrieves macro definitions stored in file MCALL;n.
Macro Execute	[n]Mx [a]	Executes macro x [n] times, while passing numeric argument [a].
Macro Immediate	[n] <definition>	Defines and executes a macro n times. Stores it as macro number 1.

7.7.6 File Input and Output Commands

The input and output commands control the movement of text to and from input and output files, and they save files. Table 7–8 lists these commands.

Table 7–8: EDI Input/Output Commands

Command	Format	Function
File	FIL filespec	Transfers lines from input file to both the output file and the specified file until a form feed or end-of-file (EOF) is encountered (line-by-line mode only).
Read	REA [n]	Reads next n blocks of text into block buffer. If buffer contains text, new text is appended to it.
Renew	REN [n]	Writes the current block to the output file and reads new block from the input file.
Save	SA [n] [filespec]	Saves current line and the next n-1 lines in the specified file. If filespec is not given, lines are saved in file SAVE.TMP. The Save command puts the temporary file in the directory on the device for the file you are editing. You can override the default by specifying a different device and directory.
Write	W	Writes contents of block buffer to output file and erases block buffer.

7.7.7 Device Output Commands

The device output commands direct output to your terminal or to a pseudo device (CL). They are listed in Table 7–9.

Table 7–9: EDI Device Output Commands

Command	Format	Function
List on Pseudo Device	LP	Same as the List on Terminal command, except that printing is performed on the pseudo device CL.
List on Terminal	LI	Prints on the terminal all lines remaining in block buffer (block mode) or input file (line-by-line mode), beginning at current line.
Print	P [n]	Prints the current line and the next n-1 lines. The last printed line is the new current line.
Type	TY [n]	Prints next n lines. In line-by-line mode, it is identical to the Print command. In block mode, line pointer remains at current line unless end-of-block was reached.

7.7.8 Close and Exit Commands

Close and exit commands terminate EDI operations and write the remainder of the input file into output file. Table 7–10 lists these commands.

Table 7–10: EDI Close Operation Commands

Command	Format	Function
Close	CL [filespec]	Transfers remaining lines in block buffer and input file to output file and closes files. If file specification is used, output file is renamed. EDI> prompt is issued.
Close Secondary	CLOSES	Closes secondary file.
Close & Delete	CD [filespec]	Same as Close except that input file is deleted. EDI> prompt is issued.
CTRL/Z	Press CTRL/Z	Closes files and terminates EDI.
Exit	EX [filespec]	Closes files, renames output file, and terminates EDI.
Exit & Delete	ED [filespec]	Transfers remaining lines in block buffer and input file to output file and closes file. Renames file if file specification is given. Deletes input file and terminates EDI.
Kill	KILL	Closes input and output files, and deletes output file. The EDI> prompt is issued.

7.8 EDI Commands: Detailed Reference Summary

This section lists each EDI command in alphabetical order. Each command description comprises the function of the command, the command format, usage information, and examples.

7.8.1 Add

The Add command causes the specified string to be appended to the current line.

Format

A string

Example

```
*A AGAIN. RET
```

Completes the line HAPPY DAYS ARE HERE.

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 include the space as part of the string, as follows:

```
*A  AGAIN. RET
```

7.8.2 Add & Print

The Add & Print command performs the same function as the Add command except that the new line is printed.

Format

AP string

Example

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

Adds the string and prints the new line.

7.8.3 ALTMODE Key

Pressing the ALTMODE key causes the system to print 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 ALTMODE keys. Pressing the ALTMODE key is equivalent to typing NP-1 (Next & Print command) or pressing the ESCAPE key.

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

7.8.4 Begin

The Begin command 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, and then opens the latest version of the file. The Begin command performs the same function as the Top command.

Format

B

Example

*B RET

Moves the current line pointer to the top of the block buffer (block mode is assumed).

7.8.5 Block On/Off

The Block On/Off command allows you to switch between block mode and line-by-line mode. When you enter the Block On command, block mode becomes active and the next block of text is brought into the block buffer. When you enter the Block Off command, 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, entering the Block On command when EDI is already in block mode), the command is ignored.

The Block On command is the default text access mode. It is assumed when neither ON nor OFF is specified.

Format

BL [ON]

or

BL OFF

Example

*BL ON RET

Switches EDI to block mode. The next block of text is read into the block buffer.

7.8.6 Bottom

The Bottom 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 the Verify On command is issued, the last line of the file block is displayed. Note, however, that if you deleted the last line before you issued the Bottom command, the line pointer will be located past the text, and thus the last line will not be printed. The Bottom command performs the same function as the End command (see Section 7.8.15).

Format

BO

Example

```
*V ON [RET]
*BO [RET]
THIS IS THE LAST LINE
```

Moves the current line pointer to the bottom of the block buffer and prints the last line.

7.8.7 Change

The Change 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 example.

If no match occurs, a [NO MATCH] message is displayed.

Format

[n]C /string1/string2[/]

Example

```
TO SEPERATE THE THOUGHTS, USE SEPERATE SENTENCES.
*2C /SEPE/SEPA/ [RET]
TO SEPARATE THE THOUGHTS, USE SEPARATE SENTENCES.
```

Changes the string SEPE to the string SEPA.

7.8.8 Close

The Close 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 filespec. 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

CL [filespec]

Example

```
*CL [RET]
EDI>
```

Closes both input and output files; then, returns to the initial command sequence.

7.8.9 Close Secondary

Use the Close Secondary command when you have finished extracting text from a secondary input file. You must enter the Close Secondary command before you can use another secondary file as input.

Format

CLOSES

7.8.10 Close & Delete

The Close & Delete command transfers all remaining lines in the block buffer and the input file (in that order) into the output file, and it closes both files. The input file is then deleted. If a file specification is included, the output file is renamed to the filespec. This command acts like the Close command 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

CD [filespec]

7.8.11 Concatenation Character

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 the Locate or Change command), you must change the concatenation character to some other character.

If the Concatenation Character command is used without an argument, the concatenation character is the ampersand.

Format

CC [character]

Example

```
*CC : [RET]
*L A&B:C /A&B/ABC/ [RET]
CONCATENATION TEST CONTAINING A&B.
CONCATENATION TEST CONTAINING ABC.
*CC [RET]
```

In the previous command lines, 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 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: the ampersand.

7.8.12 CTRL/Z

CTRL/Z is a CLI function that terminates EDI. Pressing CTRL/Z (pressing the CTRL key and the Z key simultaneously) terminates the editing session. If an output file is open when CTRL/Z is pressed, 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 pressed, all files are closed (including any open secondary input file), and EDI exits. If you press CTRL/Z as an input line terminator, that line is erased.

7.8.13 Delete

The Delete command causes a specified number of lines 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

D [n]

or

D -n

Example

```
*D -5 [RET]
```

Deletes the five previous lines in the block buffer.

7.8.14 Delete & Print

The Delete & Print command performs the same function as the Delete command except that the new current line is printed when all the specified 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

```
THIS IS LINE 1
THIS IS LINE 2
THIS IS LINE 3
THIS IS LINE 4
*DP 2 [RET]
THIS IS LINE 3
```

Deletes the first two lines and prints the current line.

7.8.15 End

The End command 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 the Verify On command was issued. Note, however, that if the last line was deleted before you issued the End command, the pointer will be located past the text; thus, the last line will not be printed. The End command performs the same function as the Bottom command.

Format

E

Example

```
*V ON [RET]
*E [RET]
THIS IS THE LAST LINE
```

Moves the current line pointer to the bottom of the block buffer (block mode is assumed).

7.8.16 Erase

In line-by-line mode, the Erase 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 RET
```

Erases the contents of the current block buffer and the next four blocks. These blocks are not written into the output file.

7.8.17 ESCAPE Key

Pressing the ESCAPE key causes the system to print 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 ESCAPE keys. Pressing the ESCAPE key is equivalent to typing NP-1 (Next & Print command) or pressing the ALTMODE key.

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

7.8.18 Exit

The Exit 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

EX [filespec]

Example

```
*EX RET
```

Terminates the editing session without renaming the output file. EDI then displays the following:

```
[EXIT]
EDI>
```

The output filename.type is the same as the input filename.type. The version number is one greater than that of the input file.

7.8.19 Exit & Delete

The Exit & Delete command functions in the same way as the Close & Delete command except that EDI also terminates.

Format

ED [filespec]

Example

```
*ED NEWFILE.DOC [RET]
[EXIT]
>
```

Closes and deletes file; then, exits from EDI.

7.8.20 File

The File 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 EOF 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.

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

FIL filespec

Example

```
*FIL SEC.DAT [RET]
```

Writes the contents of the input file, from the current line to the end, into both the output file and the file SEC.DAT.

7.8.21 Find

The Find 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 the Verify On command is in effect.

The Find command 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]F [string]

Example

```
*V ON [RET]
*F LOOK [RET]
LOOK AT THE FIRST CHARACTER IN THE LINE.
```

Searches the block buffer (or file) for a line that begins with LOOK and prints the line when the line is found.

7.8.22 Form Feed

The Form Feed 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 [RET]
THIS IS THE LAST LINE ON THE PAGE
*FF [RET]
```

Inserts a form feed into the text following the current line.

7.8.23 Insert

The Insert 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

I [string]

Examples

```
*I TEXT INSERT IN EDIT MODE [RET]
```

Inserts a line of text immediately after the current line.

```
*I [RET]
TEXT INSERT 1 IN INPUT MODE [RET]
TEXT INSERT 2 IN INPUT MODE [RET]
ETC. [RET]
[RET]
*
```

Illustrates that the Insert command followed by the RETURN key causes EDI to switch to input mode. A series of new lines are then input following the current line. Also illustrates that pressing the RETURN or ESCAPE key as the only character in an input line causes EDI to return to edit mode and to prompt for a new command.

7.8.24 Kill

The Kill 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 [RET]
EDI>
```

Deletes the output file and displays the prompt.

At this point, you can return control to your CLI by pressing CTRL/Z or by typing a file specification for a file to be edited.

7.8.25 Line Change

The Line Change command is similar to the Change command 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

```
THES ES THE LINE TO BE ESSUED.
*V ON [RET]
*LC /ES/IS [RET]
THIS IS THE LINE TO BE ISSUED.
```

Corrects the errors in the line and prints the corrected line.

7.8.26 List on Pseudo Device

The List on Pseudo Device 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 the system line printer.

Format

LP

Example

*LP [RET]

Prints on the pseudo device CL all remaining lines in the block buffer or all remaining lines in the input file.

7.8.27 List on Terminal

The List on Terminal 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 re-positioned to the top of the input file or block buffer.

If terminal host synchronization is installed at system generation, you can control printing functions by pressing CTRL/O, CTRL/S, and CTRL/Q. To suppress printing at any point, press CTRL/O. Printing can be suspended temporarily by pressing CTRL/S and resumed by pressing CTRL/Q.

Format

LI

Example

*LI [RET]

Prints on the terminal all remaining lines in the block buffer or all remaining lines in the input file.

7.8.28 Locate

The Locate 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 the Verify On command 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]L [string]

Example

*L PPY [RET]

Locates the line HAPPY DAYS ARE HERE AGAIN.

EDI searches the file or block buffer and, if the Verify On command is in effect, prints the line when it is located. The current line pointer is set to the located line.

7.8.29 Macro

The Macro 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

Parameter

x

Specifies the macro number (1, 2, or 3).

Examples

```
*MACRO 1 %L ABC&PA /ABC/DEF [RET]
*M1 10 [RET]
```

Finds the nth occurrence of the string ABC in the current block and replaces that occurrence and all remaining occurrences within the block with the string DEF. Executes the macro and searches for the tenth and succeeding occurrences of ABC.

```
*MACRO 1 PA /ABC/DEF/&PA /GHI/JKL/&REN [RET]
*5M1 [RET]
```

Changes 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).

7.8.30 Macro Call

The Macro Call 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 the Peripheral Interchange Program (PIP) with the New Version (/NV) subswitch, to rename the file. (Refer to Chapter 12 for descriptions of PIP commands.)

Format

MC

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 Macro Call 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 RET
```

Retrieves the macro definitions stored in file MCALL;n, where n represents the latest version of the file MCALL.

7.8.31 Macro Execute

The Macro Execute 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]

Parameters

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 RET
```

Executes macro number 1 twice.

```
*3M2 5 RET
```

Executes macro number 2 three times, and passes the numeric argument 5 each time the macro is executed.

The example in Section 7.10.4 illustrates how to use the EDI macro commands when editing a file.

7.8.32 Macro Immediate

The Macro Immediate 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 following two macro commands:

```
MACRO 1 definition
nM1
```

Format

[n] <definition>

Example

```
*<L ABC&C /ABC/DEF> RET
```

Searches the current block buffer for the string ABC and, when ABC is located, changes the string to DEF. This macro is stored as macro number 1.

The example in Section 7.10.3 illustrates the use of the Macro Immediate command.

7.8.33 Next

The Next command moves the current line pointer backwards and forwards 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

N [n]

or

N -n

If n is not specified, a value of +1 is assumed. A negative value for n can be used only in the block mode.

Example

```
*N -5 RET
```

Moves the current line pointer back five lines only in block mode.

7.8.34 Next & Print

The Next & Print 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

```
LINE 1 OF THE FILE
LINE 2 OF THE FILE
LINE 3 OF THE FILE
LINE 4 OF THE FILE
```

Assumes the previous four lines are contained in the file and the line pointer is at the first line. If the following command lines are issued, EDI returns the following listing:

```
*NP 2 [RET]
LINE 3 OF THE FILE
* [RET]
LINE 4 OF THE FILE
* [ESC]
LINE 3 OF THE FILE
* [ESC]
LINE 2 OF THE FILE
```

7.8.35 Open Secondary

The Open Secondary 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 (SS) command for input.

Format

OP filespec

Example

```
*OP RICKS.MAC [RET]
*SS [RET]
*REA 1 [RET]
```

Opens the file RICKS.MAC as a secondary input file and then reads in the first block.

7.8.36 Output On/Off

The Output On/Off 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

OU [ON]

or

OU OFF

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

Example

```
*BL OFF [RET]
*OU OFF [RET]
*N 5 [RET]
*OU ON [RET]
```

Shows how to bypass five lines of text in the input file so that these lines are not written into the output file. The command lines perform the following tasks:

- The first command line sets line-by-line mode.
- The second command line disables the transfer of text to the output file.
- The third command line bypasses five consecutive lines of text from the input file.
- The fourth command line reenables the transfer of text to the output file.

7.8.37 Overlay

The Overlay 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 press the RETURN key as the only character in an input line.

Format

O [n]

Example

```
*O 2 [RET]
```

Deletes two lines and causes EDI to enter input mode.

7.8.38 Page

The Page 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, and 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. This command can be used in block mode only.

Format

PAG n

Example

```
*PAG 1 [RET]
[00050 LINES READ IN]
[00050 LINES READ IN]
[00050 LINES READ IN]
[00050 LINES READ IN]
[00017 LINES READ IN]
[PAGE 1]
*
```

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 EOF mark. Thus, in a file without form feeds (that is, most files), EDI renews the block buffer until it encounters an EOF mark. Note that the final block contains 17 lines of text.

7.8.39 Page Find

The Page Find 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 the Verify On command is in effect. This command can be used only in block mode.

Format

[n]PF string

7.8.40 Page Locate

The Page Locate 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]PL 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 EOF if it cannot locate the string.

7.8.41 Paste

The Paste 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

PA /string1/string2[/]

Example

```
YIGER, YIGER, BURNING BRIGHY
IN YHE FORESYS OF YHE NIGHY
*PA /Y/T RET
```

Corrects the errors in the previous lines.

If the Verify On command is in effect, all corrected lines are printed. To discontinue printing, press CTRL/O.

7.8.42 Print

The Print command prints the current line and the next n-1 lines on the terminal. The last line printed becomes the new current line. If n is not specified, a value of 1 is assumed.

Figure 7–1 illustrates both the Print and the Type commands.

Format

P [n]

7.8.43 Read

The Read command reads the next n blocks of text into the block buffer. If a block is already in the buffer, the new block or blocks 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 command 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

REA [n]

If n is not specified, a value of 1 is assumed. The value of n must be positive.

Example

```
*SIZE 15 [RET]
*REA 4 [RET]
```

Reads four 15-line blocks of the input file into the block buffer.

7.8.44 Renew

The Renew 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 7.3 for information on how EDI block buffers are processed.

Format

REN [n]

Example

```
*REN 10 [RET]
```

Transfers consecutively 10 blocks 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.

7.8.45 RETURN Key

Pressing the RETURN key in edit mode causes the next line in the file or block buffer to be printed. 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 RETURN keys. This command is equivalent to NP+1 (Next & Print command).

In input mode, pressing the RETURN key causes EDI to change from input mode to edit mode.

7.8.46 Retype

The Retype command replaces the current line with a string. If a string is not specified, the line is deleted.

Format

R [string]

Example

```
*R THIS IS A NEW LINE RET
```

Replaces the current line with the string THIS IS A NEW LINE.

7.8.47 Save

The Save 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 directory 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

SA [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. You should proceed, as follows:

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 SA 6 command. (This copies the six lines to be saved into the file SAVE.TMP.)
4. Issue the Kill command to terminate the editing session.
5. Start a new editing session using HELP.MAC as the input file.
6. Locate each place where the six lines are to be inserted and issue the Unsave command.
7. Make further edits to the input file, as desired, or issue the Exit command.

EDI does not delete the save file. It remains on the specified volume until deleted.

7.8.48 Search & Change

The Search & Change 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 [ILL STRING CONST]. 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 EOF or at the bottom of the block buffer.

Format

SC /string1/string2[/]

Example

```
*V ON [RET]
*SC /THES/THIS/ [RET]
THIS IS THE LINE TO BE ISSUED.
```

Corrects the following text:

```
THES IS THE LINE TO BE ISSUED.
```

The corrected line is printed because the Verify On command is in effect.

7.8.49 Select Primary

The Select Primary 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

```
*OP SECOND.MAC [RET]
*SS [RET]
*REN 10 [RET]
*CLOSES [RET]
*SP [RET]
```

Directs EDI to perform the following tasks:

1. Open the secondary file SECOND.MAC.
2. Select SECOND.MAC as the secondary input file.
3. Read 10 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.

7.8.50 Select Secondary

With the Select Secondary command, you select the secondary file as the input file.

Format

SS

Refer to Section 7.8.49 for an example of the Select Secondary command.

Note

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.

7.8.51 Size

The Size 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 7.3.)

Format

SIZE n

Example

```
*SIZE 50 RET
```

Conditions EDI to read 50 lines into the block buffer during a single Read or Renew command.

7.8.52 Tab On/Off

The Tab On/Off 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

TA [ON]

or

TA OFF

If neither ON nor OFF is specified when a TAB command is issued, ON is assumed.

Examples

```
*TA ON [RET]
*I [RET]
; THIS IS A SAMPLE OF TABBING. [RET]
THIS LINE GETS A TAB [RET]
1: THIS ONE DOESN'T [RET]
END [RET]
[RET]
*TA OFF [RET]
*N -3 [RET]
*P 4 [RET]
; THIS IS A SAMPLE OF TABBING.
        THIS LINE GETS A TAB
1:  THIS ONE DOESN'T
        END
```

7.8.53 Top

The Top 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 using the Top command, you can enter lines preceding the first line in the block or file.

The Top command differs from TOF in the following ways:

- In line-by-line mode, the Top command creates a new file and moves the current line pointer to the top of the file. Unlike the TOF command, it does not cause EDI to return to block mode.
- In block mode, the Top command moves the current line pointer to the top of the current block and does not create a new output file. The TOF command moves the current line pointer to the top of the file and creates a new output file.

Format

T

Example

```
*T [RET]
```

Directs the current line pointer to the top of the current block in block mode.

7.8.54 Top of File

The Top of File (TOF) command creates a new version of the file and returns the current line pointer to the first line of the file. The TOF command copies the input file into the output file, closes both, and 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

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

Creates the following lines in the output file if the input terminal is capable of generating lowercase input:

```
this line is entered in lower case
THIS LINE IS CONVERTED TO UPPER CASE
```

However, in both instances, the letters are converted to uppercase before the file is closed.

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

7.8.58 Verify On/Off

The Verify On/Off command controls the display of lines specified by the Locate and Change commands. Use the Verify On command to display a line located by the Locate command or to display a line changed by the Change command. Use the Verify Off command to inhibit the display of these lines. The Verify On command is the default when EDI is started.

Format

V [ON]

or

V OFF

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

Example

```
*V OFF [RET]
*L VERIFY [RET]
*P [RET]
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
*N -2 [RET]
*V ON [RET]
*L VERIFY [RET]
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
```

Issues the Print command to demonstrate that the desired line has been located when the Verify Off command is in effect, but when the Locate command is reissued with the Verify On command, EDI automatically prints the line.

Examples

```
*TA ON [RET]
*I [RET]
; THIS IS A SAMPLE OF TABBING. [RET]
THIS LINE GETS A TAB [RET]
1: THIS ONE DOESN'T [RET]
END [RET]
[RET]
*TA OFF [RET]
*N -3 [RET]
*P 4 [RET]
; THIS IS A SAMPLE OF TABBING.
        THIS LINE GETS A TAB
1:  THIS ONE DOESN'T
        END
```

7.8.53 Top

The Top 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 using the Top command, you can enter lines preceding the first line in the block or file.

The Top command differs from TOF in the following ways:

- In line-by-line mode, the Top command creates a new file and moves the current line pointer to the top of the file. Unlike the TOF command, it does not cause EDI to return to block mode.
- In block mode, the Top command moves the current line pointer to the top of the current block and does not create a new output file. The TOF command moves the current line pointer to the top of the file and creates a new output file.

Format

T

Example

```
*T [RET]
```

Directs the current line pointer to the top of the current block in block mode.

7.8.54 Top of File

The Top of File (TOF) command creates a new version of the file and returns the current line pointer to the first line of the file. The TOF command copies the input file into the output file, closes both, and 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 [RET]
```

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.

7.8.55 Type

The Type command is similar to the Print command (Section 7.8.42). In line-by-line mode, the two are identical. In block mode, the Type command does not move the line pointer after displaying the requested text unless the end-of-block is encountered. In this case, the line pointer remains at the last line before the end-of-block. Figure 7–1 illustrates both the Print and Type commands. The arrow in the figures represents the line pointer.

If n is not specified, a value of 1 is assumed.

Format

TY [n]

7.8.56 Unsave

The Unsave 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 the Save command.

Format

UNS [filespec]

Example

```
*UNS SEC.DAT;1 [RET]
```

Retrieves all the lines in the file SEC.DAT;1 and inserts them after the current line.

Section 7.10.2 contains an example that uses the Save and Unsave commands.

7.8.57 Uppercase On/Off

The Uppercase On/Off command allows you to enter lowercase letters from a terminal and have them converted to uppercase letters. If the Uppercase Off command is issued, all input letters are accepted as they are entered, including the EDI commands.

Format

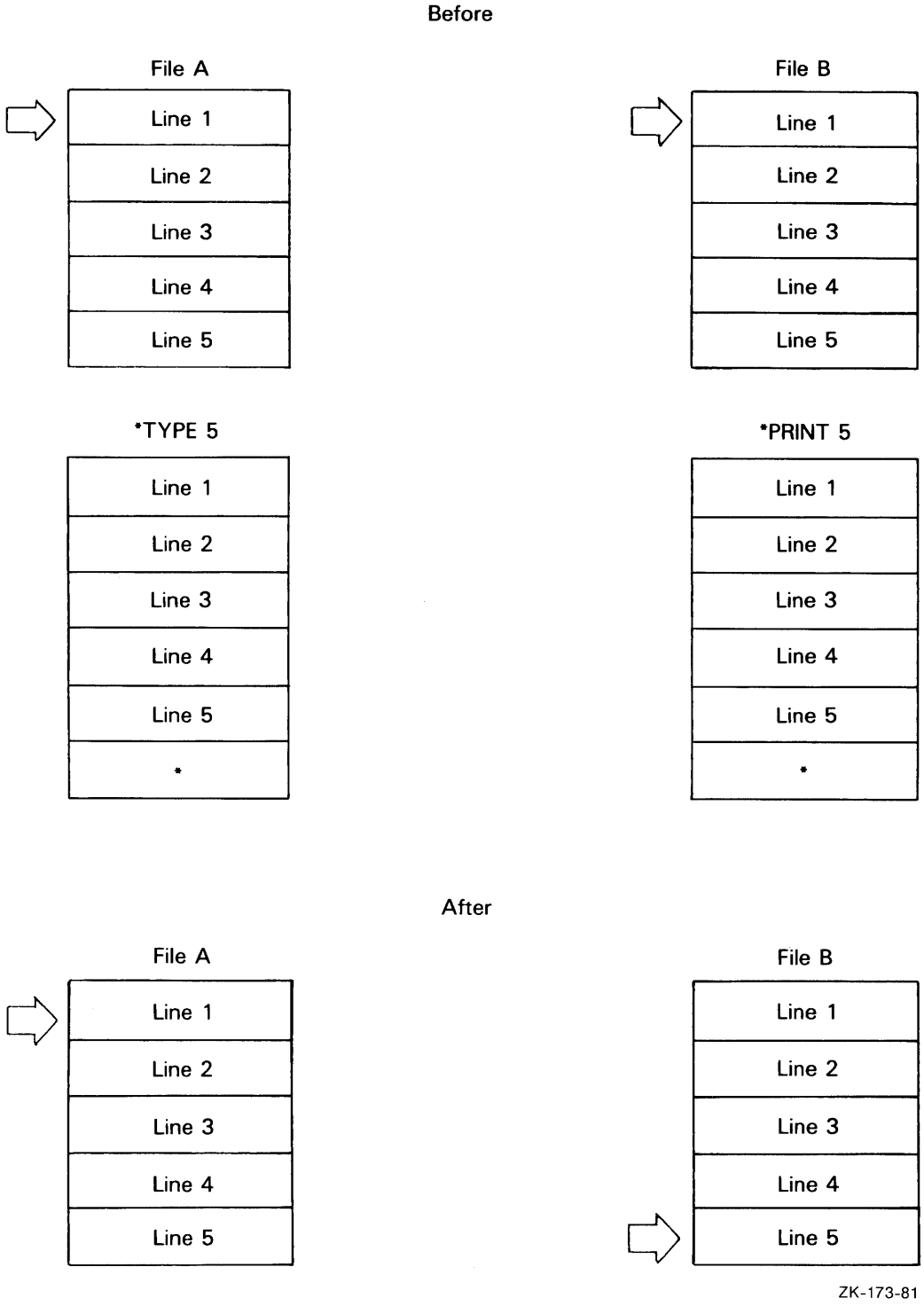
UC [ON]

or

UC OFF

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

Figure 7–1: Line Pointer Position for the Type Command and the Print Command



Example

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

Creates the following lines in the output file if the input terminal is capable of generating lowercase input:

```
this line is entered in lower case
THIS LINE IS CONVERTED TO UPPER CASE
```

However, in both instances, the letters are converted to uppercase before the file is closed.

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

7.8.58 Verify On/Off

The Verify On/Off command controls the display of lines specified by the Locate and Change commands. Use the Verify On command to display a line located by the Locate command or to display a line changed by the Change command. Use the Verify Off command to inhibit the display of these lines. The Verify On command is the default when EDI is started.

Format

V [ON]

or

V OFF

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

Example

```
*V OFF [RET]
*L VERIFY [RET]
*P [RET]
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
*N -2 [RET]
*V ON [RET]
*L VERIFY [RET]
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
```

Issues the Print command to demonstrate that the desired line has been located when the Verify Off command is in effect, but when the Locate command is reissued with the Verify On command, EDI automatically prints the line.

7.8.59 Write

The Write 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

W

Example

```
*W [RET]
*REA 2 [RET]
```

Writes the block buffer into the output file and then erases it. Next, the next two blocks are read into the block buffer.

7.9 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; it 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 the Peripheral Interchange Program (PIP) or File Transfer Utility Program (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 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 (/PU) switch or the DCL command DELETE.
- 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 they are interpreted by EDI as termination characters. You should avoid using them because 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 Task Builder (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 by using FLX. You can then use EDI to process the file.

7.10 Editing Operation Examples

Examples of the following 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 Section 7.10.2.
- An immediate macro command is defined and executed in a single step.
- A file containing errors is edited by using the macro commands.

Numbered explanations follow each example.

7.10.1 File Editing Example

```
>EDI PRTBLD.CMD                ❶
[PAGE     1]
*P *                ❷
;
; COMMAND FILE TO BUILD
; PRNT SYMBIONT
; FOR RSX-11M-PLUS 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:$DELET
/
[*EOB*]
*T                3
[PAGE     1]
*PL PRNT                4
; PRNT SYMBIONT
*C /RN/RIN/                5
; PRINT SYMBIONT
* [RET]                6
; FOR RSX-11M-PLUS MAXXED SYSTEM
*C /XX/PP/                7
; FOR RSX-11M-PLUS MAPPED SYSTEM
*NP 3                8
[1,54]PRT/MM/-CP,LP:=PRTBLD/MP
*C /-CP,/CP,                9
[1,54]PRT/MM/CP,LP:=PRTBLD/MP
*PL PAR=                10
PAR=PARK:0:10000
*C /RK/R4K/                11
PAR=PAR4K:0:10000
*NP -3                12
; OPTIONS
*AP  INPUT                13
; OPTIONS INPUT
*PL UC                14
UC=[10,1]
*C /UC/UIC/                15
UIC=[10,1]
* [RET]                16
;
* [RET]
; SPECIFY
*DP                17
; SPECIFY FLAG WHICH CONTROLS
*PL INH                18
; TO INHIBIT DELETION USE
*I                19
;
;       GBLPAT=PRT:$DELET:0
[RET]
*PL RB                20
```

```
; FILE DELETION ENARBLED
*C /R//                    ❷1
; FILE DELETION ENABLED
* [RET]
;
* [RET]                    ❷2
GBLPAT=PRT:$DELET
*AP :1                     ❷3
GBLPAT=PRT:$DELET:1
*TOF                       ❷4
[PAGE     1]
*P *                       ❷5
;
; COMMAND FILE TO BUILD
; PRINT SYMBIONT
; FOR RSX-11M-PLUS MAPPED SYSTEM
;
;
[1,54]PRT/MM/CP,LP:=PRTBLD/MP
;
; OPTIONS INPUT
;
STACK=40
PAR=PAR4K:0:10000
UNITS=4
TASK=PRT...
ASG=CO:2,LP:3
PRI=60
UIC=[10,1]
;
; SPECIFY FLAG WHICH CONTROLS
; FILE DELETION AFTER PRINTING
;
; TO ENABLE DELETION USE
;
;       GBLPAT=PRT:$DELET:1
;
; TO INHIBIT DELETION USE
;
;       GBLPAT=PRT:$DELET:0
;
; DEFAULT FROM ASSEMBLY IS
; FILE DELETION ENABLED
;
GBLPAT=PRT:$DELET:1
/
[*EOB*]
*EX                        ❷6
[EXIT]
```

Notes for File Editing Example

❶ EDI opens file PRTBLD.CMD for editing.

❷ The P command (Print) prints the contents of the file, and the following errors are detected in the text of the file:

PRNT should be PRINT.

MAXXED should be MAPPED.
/-CP should be /CP.
INPUT should be appended to the line containing the word OPTIONS.
PARK should be PAR4K.
UC should be UIC.
The line containing ; SPECIFY should be deleted.
The comment line containing the format used to inhibit deletion is missing.
ENARBLED should be ENABLED.
A :1 should be appended to the line following the word $DELET.

The end of buffer is reached and EDI prints the EOB message.

❸ The T command (Top) moves the line pointer to the top of the file and editing begins.

❹ The PL command (Page Locate) locates and displays the first line in error.

❺ The C command (Change) corrects the line and displays the corrected line.

❻ The RETURN key moves the line pointer to display the next line in error.

❼ The C command (Change) corrects the line and displays the corrected line.

❽ The NP command (Next & Print) locates and displays the next line in error.

❾ The C command (Change) corrects the line and displays the corrected line.

❿ The PL command (Page Locate) locates and displays the next line in error.

⓫ The C command (Change) corrects the line and displays the corrected line.

⓬ A line in error was bypassed by mistake; therefore, an NP command (Next & Print) is issued to move the line pointer back three lines.

⓭ The AP command (Add & Print) corrects the line and displays the corrected line.

⓮ The PL command (Page Locate) locates and displays the next line in error.

⓯ The C command (Change) corrects the line and displays the corrected line.

⓰ Pressing the RETURN key twice moves the line pointer down two lines and locates the next line in error.

⓱ The DP command (Delete & Print) deletes the line containing ; SPECIFY and prints the next line.

⓲ The PL command (Page Locate) locates the point in the file where the new comment lines are to be inserted.

⓳ The I command (Insert) switches EDI from edit mode to input mode, and allows two lines to be entered. Pressing the RETURN key twice switches input mode back to edit mode.

⓴ The PL command (Page Locate) locates the next line in error.

㉑ The C command (Change) corrects the spelling error and displays the corrected line.

㉒ Pressing the RETURN key twice moves the line pointer down two lines and locates the last line in error.

㉓ The AP command (Add & Print) appends :1 following the word $DELET.

㉔ The TOF command (Top of File) moves the line pointer to the top of the file.

㉕ The P command (Print) prints the complete file with all corrections.

㉖ The EX command (Exit) closes the file and terminates the editing session.

7.10.2 Save and Unsave Example

```
EDI>START.DAT                ❶
[PAGE     1]
*LI                ❷
THIS IS LINE 1 PAGE 1
THIS IS LINE 2 PAGE 1
THIS IS LINE 3 PAGE 1
THIS IS LINE 4 PAGE 1
THIS IS LINE 5 PAGE 1
[*EOB*]
*T                 ❸
*SA 5 SAV1.DAT               ❹
*T                 ❺
*SA 5 SAV2.DAT               ❻
*CL                ❼
EDI>SAV1.DAT                 ❽
[PAGE    1]
*LI                ❾
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/                ❿
THIS IS LINE 1 PAGE 2
THIS IS LINE 2 PAGE 2
THIS IS LINE 3 PAGE 2
THIS IS LINE 4 PAGE 2
THIS IS LINE 5 PAGE 2
```

```
*CL          ⓫
EDI>SAVE2.DAT          ⓬
[PAGE     1]
*LI          ⓭
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 3/          ⓮
THIS IS LINE 1 PAGE 3
THIS IS LINE 2 PAGE 3
THIS IS LINE 3 PAGE 3
THIS IS LINE 4 PAGE 3
THIS IS LINE 5 PAGE 3
*CL          ⓯
EDI>START.DAT          ⓰
[PAGE    1]
*BO          ⓱
THIS IS LINE 5 PAGE 1
*UNS SAV1.DAT          ⓲
*UNS SAV2.DAT
*T          ⓳
*LI          ⓴
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          ㉑
[EXIT]
```

Notes for Save and Unsave Example

❶ EDI opens file START.DAT.

❷ The LI command (List on Terminal) prints the file.

❸ The T command (Top) moves the line pointer to the top of the block.

❹ The SA command (Save) saves the five lines in a separate file.

❺ The T command (Top) moves the line pointer to the top of the file.

❻ The SA command (Save) generates a second saved file.

❼ The CL command (Close) closes the primary input file.

❽ EDI opens the first save file.

❾ The LI (List on Terminal) command displays the file.

❿ The PA command (Paste) changes PAGE 1 to PAGE 2 in all lines.

⓫ The CL command (Close) closes the first save file.

⓬ EDI opens the second save file.

⓭ The LI command (List on Terminal) displays the contents of the file.

⓮ The PA command (Paste) changes PAGE 1 to PAGE 3 in all lines.

⓯ The CL command (Close) closes the second save file.

⓰ EDI opens the original input file.

⓱ The BO command (Bottom) locates the last line in the file.

⓲ Two UNS commands (Unsave) append the two save files to the original input file.

⓳ The T command (Top) moves the line pointer to the top of the file.

⓴ The LI command (List on Terminal) displays the contents of the combined file.

㉑ The EX command (Exit) closes the input file and terminates the editing session.

7.10.3 Macro Immediate Example

```
EDI>START.DAT                ❶
[PAGE    1]
*LI                ❷
ABC IN LINE 1 - ABC
ABC IN LINE 2 - ABC
ABC IN LINE 3 - ABC
ABC IN LINE 4 - ABC
ABC IN LINE 5 - ABC
  .
  .
  .
ABC IN LINE n - ABC
[*EOB*]
*4<F ABC&C /ABC/DEF/>                ❸
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
*EX                ❹
```

Notes for Macro Immediate Example

❶ EDI opens file START.DAT.

❷ The LI command (List on Terminal) prints the file.

❸ The immediate macro is defined and executed. It finds the first four lines that start with ABC and changes the first occurrence of the string ABC to DEF. The F command (Find) causes the line to be printed before the change. The C command (Change) causes the line to be printed after the change.

❹ The EX command (Exit) closes the file and terminates the editing session.

7.10.4 Macro Commands Example

```
EDI>START.DAT          ❶
[PAGE    1]
* LI          ❷
THIS LITTLE FILE HAS
MANY CONNON ETTORS SO
WE CAN SHOW YOU HOW
YHE MACRO CONNANDS CAN
BE USED.
FIRST, YHE DESIRED MACRO
MUST BE DEFINED; YHE LINE
POINTER IS MOVED TO A LINE
WITH AN ETTOR; AND YHEN, YHE
MACRO EXECUTE CONNAND
IS ISSUED TO COTTECT YHE
ETTOR
[*EOB*]
*MACRO 1 C /NN/MM/          ❸
*MACRO 2 SC /TT/RR/
*MACRO 3 PA /YHE/THE/
*M3          ❹
THE MACRO CONNANDS CAN
FIRST, THE DESIRED MACRO
MUST BE DEFINED; THE LINE
WITH AN ETTOR, AND THEN, THE
IS ISSUED TO COTTECT THE
*NP 2          ❺
MANY CONNON ETTORS SO
*M1          ❻
MANY COMMON ETTORS SO
*M2          ❼
MANY COMMON ERRORS SO
*NP 2          ❽
THE MACRO CONNANDS CAN
```

```
*M1              ❾
THE MACRO COMMANDS CAN
*M2              ❿
WITH AN ERROR; AND THEN, THE
* [RET]            ⓫
MACRO EXECUTE CONNAND
*M1              ⓬
MACRO EXECUTE COMMAND
*M2              ⓭
IS ISSUED TO CORRECT THE
*M2              ⓮
ERROR
*T               ⓯
*LI              ⓰
THIS LITTLE FILE HAS
MANY COMMON ERRORS SO
WE CAN SHOW YOU HOW
THE MACRO COMMANDS CAN
BE USED.
FIRST, THE DESIRED MACRO
MUST BE DEFINED; THE LINE
POINTER IS MOVED TO A LINE
WITH AN ERROR; AND THEN, THE
MACRO EXECUTE COMMAND
IS ISSUED TO CORRECT THE
ERROR.
[*EOB*]
*EX              ⓱
```

Notes for Macro Commands Example

❶ EDI opens file START.DAT.

❷ The LI command (List on Terminal) displays the contents of the file and reveals any errors. The following errors are located:

> The string NN is used in place of MM (see macro 1).
> The string TT is used in place of RR (see macro 2).
> The string YHE is used in place of THE (see macro 3).

❸ Three Macro commands define the macros that will correct the errors.

❹ The M3 command (Macro Execute) changes all YHE strings to THE.

❺ The NP command (Next & Print) locates a line with errors.

❻ The M1 command (Macro Execute) changes NN to MM.

❼ The M2 command (Macro Execute) changes TT to RR.

❽ The NP command (Next & Locate) locates the next line in error.

❾ The M1 command (Macro Execute) changes NN to MM.

❿ The M2 command (Macro Execute) locates the next TT string and changes it to RR.

⓫ Pressing the RETURN key locates the next line in error.

⓬ The M1 command (Macro Execute) changes NN to MM.

⓭ The M2 command (Macro Execute) locates the next TT string and changes it to RR.

⓮ The M2 command (Macro Execute) locates the last error in the file and corrects it.

⓯ The T command (Top) moves the line pointer to the top of the file.

⓰ The LI command (List on Terminal) displays the contents of the file.

⓱ The EX command (Exit) closes the file and terminates the editing session.

7.11 EDI Messages

The four classes of EDI error messages are as follows:

- Command level information messages
- File access error messages
- Error messages requiring EDI 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.

7.11.1 EDI Command Level Informational Messages

Messages in the command level information class display 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 the asterisk prompt (*) for a new command.

The messages in this class are as follows:

[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 following command procedure:

1. Type KILL.
2. Enter a nonexistent file name when EDI prompts for a new file specification. EDI creates a new file and enters input mode.
3. Press the RETURN key to enter edit mode.

4. Reduce the number of lines read into each buffer by using the Size command.
5. Use the Kill command to abandon the file.
6. Enter the name of the desired file when EDI prompts for a new file specification.

[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 the RETURN key 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).

User Action: Reenter the correct command line.

[ILL NUM]

Explanation: Either you have supplied a nonnumeric character when a numeric character is required, or you have given a negative number when a positive number is required.

User Action: Reenter the correct command line.

[ILL STRING CONST]

Explanation: A search string specified in a Change, Line Change, Paste, or Search & Change command does not contain a matching string termination character (for example, PA /ALPHABETA, instead of PA /ALPHA/BETA).

User Action: Reenter the correct command line.

[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.

User Action: Reenter the correct command line.

[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 switch /RE or the DCL command RENAME 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 sign (%) character and thus demands a numeric argument.

User Action: Specify the appropriate numeric argument and reenter the command line.

[MCALL FILE DOES NOT EXIST]

Explanation: You have issued an Macro Call 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 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.

User Action: Open the input file and reenter the command line.

[NO MATCH]

Explanation: You have issued a Change command with a string to be changed that is not in the current line.

User Action: Specify another string and reenter the command 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.

User Action: This message is for your information. No user action is required.

[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 by using the Close Secondary command and then open the desired secondary file with the Open Secondary 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 a Select Primary command and a Close Secondary command.

[SYNTAX ERROR]

Explanation: You entered a command that is syntactically incorrect.

User Action: Reenter the correct command line.

[TOO MANY CHARS]

Explanation: A Change, Line Change, Paste, Search & Change, 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 correct command line.

[•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.

User Action: This message is for your information. No user action is required.

[•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.

User Action: This message is for your information. No user action is required.

[•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.

7.11.2 EDI File Access Error Messages

Error messages in the file access class mean that you have tried to access directories, files, or devices that are not present in the host system.

The format of the error messages is as follows:

```
EDI -- message
```

After the message is displayed, EDI returns to command level and prints the asterisk (*) prompt to request input.

The messages in this class are as follows:

EDI—BAD FILE NAME

Explanation: EDI did not accept the file name.

User Action: Ensure that the file name is correct and then reenter it.

EDI—DEVICE NOT IN SYSTEM

Explanation: You have given a File, Open Secondary, Save, or Unsave command and have specified a device that does not exist in the host system.

User Action: Specify a device available in the system and reenter the command line.

EDI—FILE DOES NOT EXIST

Explanation: You have given a File or Save command and have specified a directory that does not exist on the specified volume.

User Action: Specify the correct file specification and reenter the command line.

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

7.11.3 EDI Error Messages Requiring 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.

The format of the error messages is as follows:

```
EDI -- message
```

After the appropriate message has been displayed, EDI prompts with the following:

```
EDI>
```

You may terminate the editing session at this point by pressing the RETURN key 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 are as follows:

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 File Verification Structure Utility (VFY) to determine the extent of the damage. VFY is described in Chapter 14.

EDI—FILE 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 and 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 and 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 VFY to determine the extent of the damage. VFY is described in Chapter 14.

EDI—PRIMARY FILE NOT PROPERLY CLOSED—NOT USABLE

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 the RETURN key or CTRL/Z. Use the PIP switch /UN 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:

- The specified volume was not mounted.
- The User Identification Code (UIC) under which EDI was running did not possess the necessary privileges to access the specified directory.
- The UIC was not privileged to access the specified file.

User Action: If the volume is not mounted, then mount it by 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 switch /UN to unlock the file. Reinitiate EDI and try to recover the data in the file.

EDI—BAD DIRECTORY SYNTAX

Explanation: The directory field ([directory]) is in improper format.

User Action: Specify the directory in the correct format and reenter the command line.

Note

The remaining error messages in this class should not occur and represent failures in EDI. If such errors persist, submit an 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

7.11.4 EDI 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 the following:

EDI —

The error message is then followed by the exit message on the next line as follows:

[EXIT]

You may use the following procedure on the truncated version of an output file to save the editing performed before the fatal error occurred:

1. Use the PIP switch /RE or the DCL command RENAME to rename the truncated version of the output file to avoid confusion.
2. Restart the editing session to the original input file.
3. Issue the Open Secondary command, and specify the renamed file as the secondary file.
4. Issue the Select Secondary command to select the secondary file for input.
5. Issue the 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 and Write commands as necessary to reach the EOF on the secondary file.
7. Issue the Select Primary command to select the primary file for input.
8. Issue the Close Secondary command to close the secondary file.
9. Issue the Write command to ensure that the last block was written into the output file.
10. Issue as many Read and Erase commands as necessary to bypass all input file blocks that are complete in the renamed file.
11. Continue with the normal editing session.

The messages in this class are as follows:

EDI—CALLER'S NODE 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 or delete unwanted files, or use the DCL commands PURGE and DELETE.

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 VFY to determine the extent of the damage. VFY is described in Chapter 14.

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 exceeded this limit.

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 or delete unwanted files, or use the DCL commands PURGE or DELETE.

EDI—BAD BUFFER ADDRESS OR HANDLER NOT LOADED

Explanation: The device driver may not be loaded in the system, or an internal error has occurred.

User Action: If the device driver is not loaded, load the device driver and configure the device into the system. If the device driver was loaded, contact your DIGITAL Field Service representative.

Note

The following error messages signify hardware problems. If possible, remove all important files from the volume, and contact your 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 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 8

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) as follows:

- 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 as follows:

Device	Abbreviation
PC11 paper tape punch	PP
PC11 and PR11 paper tape readers	PR
RK05 cartridge disk	DK
TE16, TU16, TU45, and TU77 magnetic tapes	MM
TS04, TS11, TSV05, and TU80 magnetic tapes	MS
TU10 and TS03 magnetic tapes	MT
TU56 DECtape	DT
TU60 tape cassette	CT
TK50 and TU81 magnetic tape drive	MU

Valid RT–11 devices are as follows:

Device	Abbreviation
RL01/RL02 cartridge	DL
RK05 cartridge disk	DK
RK06 and RK07 cartridge disks	DM
RX01 diskette	DX
RX02 diskette	DY
RX33 and RX50 diskettes	DU
RD31, RD32, RD51, and RD52 disks	DU
RC25 and RCF25 fixed/removable disks	DU
TU56 DECtape	DT
TU58 DECtape II data cartridge	DD

FLX supports all Files–11 devices, including RSX-format cassettes. The cassettes are volumes that you have initialized using the Monitor Console Routine (MCR) command INITVOL or Digital Command Language (DCL) command INITIALIZE. DOS–11 and RT–11 volumes are initialized using FLX. On RSX–11M–PLUS operating systems, DOS–11 and RT–11 volumes must be mounted with foreign characteristics before you can use FLX.

You can use FLX interactively or by means of an indirect commmand file. FLX allows only one level of indirect command file specification.

8.1 FLX Command Line

You can invoke FLX by any of the methods described in Chapter 1. Although formats for specifying FLX functions vary, the general format for entering FLX command lines is shown next.

Format

[ddnn:[[directory]]/switch[...]=]infile[,...]/switch[...]

Parameters

ddnn

Specifies the device for the FLX output.

directory

Specifies the directory on the output device.

The [directory] field is optional; if it is not specified, FLX uses the current default directory. Do not specify a directory if the output device is in RT–11 format.

switch

Specifies one of the FLX switches described in Section 8.1.1.

infile

Specifies the input file specification.

The format for entering file specifications is as follows:

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

The directory is not specified for RT–11 volumes.

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.

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 file name), 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–PLUS MCR Operations Manual*).

8.1.1 FLX Switch Descriptions

FLX provides three types of switches for file transfers:

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 American Standard Code for Information Interchange (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 directory for an output file.

The following switches accept octal numbers by default:

- /FA[:n]
- /FB[:n]
- /IM[:n]
- /BL:n
- /BS:n
- /NU:n
- /ZE[:n]

If you want to assign decimal numbers to these switches, you must follow the value of n with a period (.). In order to assign the decimal value to the /IM[:n] switch, type the following:

```
/IM:18.
```

The /DNS switch accepts decimal numbers by default. Therefore, you do not need to follow the value of the /DNS switch with a period (.) if the value is decimal.

8.1.1.1 Volume Format

FLX has three volume format switches, as described in Table 8–1, that define the format of the specified volumes.

Table 8–1: FLX Volume Format Switches

Switch	Description
/DO	Identifies the volume as a DOS–11 formatted volume.
/RS	Identifies the volume as a Files–11 formatted volume.
/RT	Identifies the volume as an RT–11 formatted volume.

Initially, input volumes default to DOS–11 format and output volumes default to Files–11 format. FLX assumes these default volume formats if you do not specify switches in the command line.

You can change the initial default by entering the /RS or /DO switch on a command line by itself. The /RS switch sets the default for input volumes to Files–11 format and output volumes to DOS–11 format. The /DO switch sets the default for input volumes to DOS–11 format and output volumes to Files–11 format.

If the /RT switch is specified on one side of a command line, the default entry for the other side is the /RS switch.

Examples

```
FLX>/RS [RET]
```

Specifies the default transfer direction from Files–11 to DOS–11.

```
FLX>/DO [RET]
```

Specifies the default transfer direction from DOS–11 to Files–11.

```
FLX>DK0:=DT0:SYS1.MAC/RT [RET]
```

Defaults the output to the /RS switch.

```
FLX>DK0:/RT=DK0:SYS1.MAC [RET]
```

Defaults the input to the /RS switch.

8.1.1.2 Transfer Mode

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 they 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 8–2 describes the transfer mode switches.

Table 8–2: FLX Transfer Mode Switches

Switch	Format	Function
Formatted ASCII	/FA[:n]	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 combination (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 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. Pressing CTRL/Z (ASCII 032_8) is treated as the logical end-of-input file for formatted ASCII transfers from DOS–11 cassette or paper tape to Files–11.
Formatted Binary	/FB[:n]	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 they are removed when transferred to Files–11 files. If you specify /FB:n with Files–11 output, fixed-length records of size n are output (512_{10} bytes is the maximum). FLX pads records with nulls to create the specified length. If you do not specify n with Files–11 output, FLX generates variable-length records. The output record size equals the input record size.
Image Mode	/IM[:n]	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 8.5):

Mode	Switch	File Type
Image	/IM[:n]	TSK, OLB, MLB, SYS, SML, ULB, EXE, CDA
Formatted Binary	/FB[:n]	OBJ, STB, BIN, LDA
Formatted ASCII	/FA[:n]	All others

If you specify n with the /FA, /FB, or /IM switch when the output file is not a Files–11 file, FLX ignores n.

See the *RSX–11M–PLUS MCR Operations Manual, RSX–11M–PLUS Command Language Manual,* and *RSX–11M–PLUS Guide to Program Development* for definitions of the file types listed above.

8.1.1.3 Control

FLX provides a number of control switches to control file processing. Table 8–3 describes these switches.

Table 8–3: FLX Control Switches

Switch Format	Function
/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 this table). Because all RT–11 files are contiguous, the /CO switch is not required with the /BL switch for RT–11 output. If you do not specify the /BL switch, 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 the /BL switch 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 decimal bytes for cassette tape (CT) output. If you do not specify the /BS switch, a block size of 128_{10} is assumed. The /BS switch is only valid in a CT output file specification with the /RS switch 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, the /BL switch is also required. FLX transfers the file types TSK, SYS, and OLB to Files–11 volumes with the /CO switch 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 the /RT switch to delete files from an RT–11 DECtape or disk. When you specify the /DE switch, the FLX command line has no output specification.

Table 8–3 (Cont.): FLX Control Switches

Switch Format	Function
/DI	Causes a directory listing of cassettes or DOS–11 volumes to be listed on a specified output file. It is used also with the /RT switch to generate a directory listing of RT–11 volumes in a specified output file. You cannot list Files–11 volume directories by 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 8.2 for information on DOS–11 volume directory manipulation. See Section 8.3 for information on RT–11 volume directory manipulation.
/DNS:n	Specifies the density of the magnetic tape, where n is 800 or 1600 bpi. If n is any other value or is not specified, FLX prints an error message. If you do not specify the /DNS switch, the magnetic tape density defaults 1600 bpi for the TS11 and 800 bpi for all other magnetic tape devices. If you specify the /DNS switch with a nonmagnetic device, FLX ignores the switch.
/FC	Indicates, when using FORTRAN files, 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 *RSX–11M–PLUS and Micro/RSX 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 the /ID switch as part of an output or input specification or type it in response to the FLX prompt (FLX>).
/LI	Same as the /DI switch.
/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 the /NU switch, four directory blocks are allocated. The maximum number of blocks that can be allocated is 37_8 or 31_{10}.
/RW	Rewinds the magnetic tape before beginning the file transfer. Specifying the /-RW switch causes FLX to begin the transfer without first rewinding the magnetic tape. If you do not specify either rewind option, the switch defaults to the /RW switch. If you specify the /RW switch with a nonmagnetic tape device, or with the /LI, /DI, or /ZE switch; FLX ignores the /RW switch.
/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 directory as the input file. FLX ignores the /UI switch if the output specification contains an explicit directory. The /UI switch is valid only for output files in DOS–11 or Files–11 format.

Table 8–3 (Cont.): FLX Control Switches

Switch Format	Function
/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 the /RT switch and the /NU switch to initialize RT–11 volumes. Initializing erases any files already on the device. The /ZE switch does not allow a file specification. For DOS–11 DECtape, the /ZE switch creates an entry for the current User Identification Code (UIC).

8.2 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 by using the FLX switches.

DOS–11 volumes must be mounted with foreign characteristics before you can use FLX.

8.2.1 Displaying DOS–11 Directory Listings

The /LI or /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.

Examples

```
FLX>DTO:[100,100]*.MAC/LI [RET]
```

Lists on your terminal the directory of all MAC files under directory [100,100] on the DOS–11 DECtape on DT0.

```
PIP>DTO:[200,200]*.TSK/LI [RET]
```

Produces the following DOS–11 DECtape directory listing:

```
❶ DIRECTORY    ❷ DT:[200,200] ❸
❹ 19-SEP-86

❺ FLX.TSK;1          ❻ 104.     ❼ 19-SEP-86   <233> ❽
  UFD.TSK;2            8.       19-SEP-86   <233>
  TKN.TSK;1            6.       19-SEP-86   <233>
  MOU.TSK;7           14.       19-SEP-86   <233>
❾     Total of 132. blocks in 4. files
```

```
PIP>CTO:[200,200]*.TSK/LI [RET]
```

Produces the following TU60 cassette directory listing:

```
❶ DIRECTORY  ❷ CT:[200,200] ❸
❹ 19-SEP-86

❺ UFD.TSK-0          ❻ 28.      ❼ 19-SEP-86   128. ❽
  TKN.TSK-0            20.        19-SEP-86   128.
  MOU.TSK-0            52.        19-SEP-86   128.
❾     Total of 132. blocks in 4. files
```

Notes on examples:

❶ Identifies the listing as a directory listing.

❷ Specifies the device name and unit number.

❸ Specifies the directory.

❹ Specifies the date the directory was listed.

❺ Specifies the file name, file type, and version number (DECtape only) or sequence number (cassettes only).

❻ Specifies the file size in decimal blocks.

❼ Specifies the file creation date.

❽ Specifies the protection code provided by the system (DECtape only) or the record size in decimal bytes for the file (cassettes only).

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

8.2.2 Deleting DOS–11 Files

You can delete files from DOS–11 disks or DECtapes by using the /DE switch. The /DE switch requires only the file specification for the file you are deleting.

Example

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

Deletes SYS1.MAC under directory [100,100] from the DOS–11 disk on DK1.

8.2.3 Initializing DOS–11 Volumes

You can initialize cassettes and DOS–11 volumes by using the /ZE switch. This switch requires only the device specification for the volume you are initializing.

Example

```
FLX>DT1:/ZE RET
```

Initializes the DECtape on DT1 in DOS–11 format.

8.3 RT–11 Volume Directory Manipulation

You can display RT–11 directory listings, delete RT–11 files, and initialize RT–11 volumes by using the FLX switches described in this section.

RT–11 volumes must be mounted with foreign characteristics before you can use FLX.

8.3.1 Displaying RT-11 Directory Listings

The /LI or /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.

Example

```
FLX>DT0:*.MAC/LI/RT RET
```

Lists on your terminal all MAC files on the RT-11 volume on DT0.

The following example shows a sample directory listing for an RT-11 disk:

```
❶ DIRECTORY  ❷ DK1:
❸  4-JUN-87

❹ SIPBOO.MAC              ❺ 49.        ❻ 4-JUN-87
❼ < UNUSED >                6.
  SIP    .MAC               10.          4-JUN-87
  SIPCD  .MAC               7.           4-JUN-87
  < UNUSED >                21.
  SIPQIO.MAC                7.           4-JUN-87
  < UNUSED >                4686.

❽    4713. free blocks

❾     Total of 73. blocks in 4. files
```

Notes on example:

❶ Identifies the listing as a directory listing.

❷ Specifies the device name and unit number.

❸ Specifies the date the directory was listed.

❹ Specifies the file name and file type.

❺ Specifies the number of blocks in the file or amount of free space.

❻ Specifies the file creation date.

❼ <UNUSED> specifies the amount of free space.

❽ Specifies the total number of free blocks on the volume.

❾ Specifies the total number of blocks allocated to files on the volume.

8.3.2 Deleting RT-11 Files

You can delete files from RT-11 disks or DECtapes by using the /DE switch with the /RT switch. The command line on which you specify /DE/RT requires only the file specification for the file you are deleting.

Example

```
FLX>DK1:SYS1.MAC/DE/RT RET
```

Deletes SYS1.MAC from the RT-11 volume on DK1.

8.3.3 Initializing RT-11 Volumes

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

Examples

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

Initializes the DECtape on DT1 in RT-11 format. When you initialize RT-11 volumes, the /ZE switch takes an optional argument in the following format:

```
/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 (using the /ZE switch), 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}.

```
FLX>DT0:/ZE:2/NU:6/RT RET
```

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 324_{10} directory entries, each of which uses nine words.

8.4 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 (using the /RS switch) or DOS–11 format (using the /DO switch). 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 8–4.

Table 8–4: Differences Between Files–11 and DOS–11 Cassette Files Format

Files–11 Format	DOS–11 Format
Standard level 2	Standard level 0
9-character file name and 3-character file type	6-character file name and 3-character file type
Blocks of any size up to 512_{10} bytes 128_{10} 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 following occur:

- 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.

8.4.1 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.

8.4.2 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:[directory]
   ```

   The variable n specifies the unit number.

2. The cassette rewinds.

3. FLX issues an additional message as follows:

```
Mount new cassette? (Y, Z (output only) or CR)
FLX>
```

4. FLX provides you with the following three alternatives:

   a. Mount the next output cassette volume, press the Y key, and then press the RETURN key. If you select this alternative, the new output cassette rewinds, 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 the 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, press the Z key, and then press the RETURN key. The new output cassette rewinds, and FLX continues by transferring data onto it. Thus, the tape is effectively zeroed (initialized) before data is transferred to it.

   c. Press the RETURN key to terminate the transfer.

      If you select this alternative, FLX assumes that end-of-file (EOF) is desired, and it 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.

8.4.3 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:[directory]filename.type
   ```

   The variable n specifies the unit number.

2. The cassette rewinds.

3. FLX issues an additional message, as follows:

   ```
   Mount new cassette: (Y, Z (output only) or CR)
   FLX>
   ```

4. FLX provides you with the following two alternatives:

   a. Mount the next input cassette volume, press the Y key, and then press the RETURN key 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
```

FLX then returns to step 3.

b. Press the RETURN key 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 press the Z key, FLX prints the following message:

```
FLX -- Bad response
```

FLX then returns to step 3.

8.5 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 combination 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 12).

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.

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.

The RSX–11M–PLUS operating system supports paper tapes only as DOS–11 devices. Therefore, you must specify the /DO switch with paper tape file specifications.

Examples

```
FLX>DK1:/RS=PR:CRTMAC.DAT/DO [RET]
```

Writes an output file whose file name is DK1:CRTMAC.DAT.

```
FLX>PP:/DO-CRTMAC.DAT/RS [RET]
FLX>DK:/RS=PR:CRTMAC.DAT/DO [RET]
```

Shows paper tape specifications for input and output file specifications.

```
FLX>PP:/DO/IM=PR:/DO [RET]
```

Copies from one paper tape to another by using the Image Mode switch (/IM), regardless of the format of the paper tapes.

8.6 FORTRAN Direct Access Files

FORTRAN direct access files must be transferred in image mode.

Examples

```
FLX>DKO:/DO/IM=FOO.TJP/RS [RET]
```

Transfers the file FOO.TJP without specifying the record length.

```
FLX>/RS/IM:18.=DKO:FOO.TJP/DO [RET]
```

Transfers the file FOO.TJP with the record length of 18_{10}.

8.7 FLX Messages

The following sections list FLX messages, describe the meaning of each message, and suggest any action necessary to correct errors.

FLX has three types of messages: informational, warning, and error.

Errors encountered by FLX during processing are reported on the initiating terminal.

Error messages describe a condition that caused FLX to terminate the processing of a command. When this occurs, FLX returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, FLX will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>FLX commandline
FLX -- message
>
```

If the command is entered in response to the FLX prompt, FLX will issue an error message and prompt you with the FLX prompt, as follows:

```
FLX>commandline
FLX -- message
FLX>
```

8.7.1 FLX Informational Message

FLX issues the following informational message:

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: No user action is required.

8.7.2 FLX Warning Messages

FLX issues the following warning messages:

FLX—WARNING—Input file out of sequence

Explanation: A multivolume cassette file is being accessed out of sequence.

User Action: The transfer will continue unless you terminate it by means of the ABORT command.

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: No user action is required.

8.7.3 FLX Error Messages

FLX issues the following error messages:

FLX—Bad list file spec

Explanation: One of the following was specified:

- More than one output file for a directory listing operation (using either the /LI or /DI switch)
- Wildcards in the output file specification for a directory operation (using either the /LI or /DI switch)

User Action: Enter the correct command line.

FLX—Bad response

Explanation: Z was entered in response to the following message:

```
Mount new cassette (Y, Z (output only) or CR)
FLX>
```

The cassette in question is an input volume.

User Action: Enter 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—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: Use a new cassette and reenter the command line.

FLX—CO files to output device not allowed

Explanation: An illegal output device (for example, magnetic tape) was specified with /CO.

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

FLX—Command syntax error

Explanation: The command was entered in a format that does not conform to syntax rules.

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

FLX—Conflicting transfer modes specified

Explanation: Conflicting transfer mode switches were entered. For example:

```
SY0:=DT0:FOO.OBJ/IM/FB
```

User Action: Reenter the command line with only one transfer mode switch specified.

FLX—Device handler missing

Explanation: The device driver for a device specified in the command line is not loaded in the system.

User Action: Load the appropriate device driver in the system and reenter the command line.

FLX—Device size exceeds 65K blocks

Explanation: The DU device selected as an RT–11 device is not an RC25, RCF25, RD51, RD52, RD53, or RX50. Devices greater than 65K blocks cannot be supported with FLX.

User Action: Specify a valid RT–11 device and reenter the command line.

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: Identify the file structure on each volume correctly, and then reenter the command line.

FLX—DT: UFD full

Explanation: The DECtape directory is full.

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

FLX—End of volume on cassette
Mount new cassette? (Y, Z (output only) or CR)

Explanation: The 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 8.4.2 if an output transfer is being performed or Section 8.4.3 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 switch is specified incorrectly.
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 in the explanation:

1. Write-enable the volume.
2. Respecify the /DO, /RT, or /RS switch 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: Use a new or corrected file name and reenter the file specification.

FLX—File not found

Explanation: The named file does not appear, as specified, in the requested directory.

User Action: Specify the file name and directory correctly, and reenter the command line.

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. Run FLX again by 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 with the correct transfer mode switch specified.

FLX—Illegal /BS size—Use 0 <N <=512. and even

Explanation: An illegal block size was specified with the /BS switch 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 and 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 user action is 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 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 directory listing operation (using either the /LI or /DI switch). There is a hardware problem with the output device (for example, a device powered down).

User Action: Correct the condition and 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—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. The possible reasons are as follows:

- 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: Specify the correct file name and directory and reenter the command line.

FLX—Undiagnosable request

Explanation: FLX does not recognize the command line syntax.

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

FLX—/CO file 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: Specify the /BL switch and reenter the command line.

FLX—* in UIC not allowed

Explanation: A wildcard was detected in the UIC of a file specification.

User Action: Reenter the command line with the UIC explicitly specified.

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 or 1600 bpi was input.

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

Chapter 9
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–PLUS operating system that includes online 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 (DY-type) diskettes.
- 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 run (spawn) if your system permits spawned tasks.

FMT can also be invoked through the DIGITAL Command Language (DCL) command INITIALIZE/FORMAT. Refer to the *RSX–11M–PLUS Command Language Manual* for more information.

9.1 FMT Command Line

You can invoke FMT by using any of the methods described in Chapter 1. The command line for FMT is shown next.

Format

ddnn:[/switch[...]]

Parameters

ddnn

Specifies the volume you are formatting.

switch

Specifies an FMT switch. A list of FMT switches is given in Table 9–1. Detailed descriptions and examples of the switches are given in Section 9.1.1.

Note

Not all switches can be used with all device types.

Table 9–1: FMT Switches

Switch	Format	Function
Bad	/BAD	Runs the Bad Block Locator task if it is installed on the system.
Density	/DENS=option	Selects options HIGH (double) or LOW (single) density for DY- and DU-type diskettes.
Error Limit	/ERL=n	Determines the number of errors FMT will allow on the volume.
Indirect Command File	/@Y	Informs FMT that it is receiving input from an indirect command file that you have created. An FMT command in this form does not allow operator intervention in the process.
Manual	/MAN	Enters manual operating mode and formats the sector or track you specify.
Noverify	/NOVE /-VE	Inhibits the default verification of a successful FMT operation.
Override	/OVR	Overrides or ignores the Manufacturer's Detected Bad Sector File (MDBSF).
Verify	/VE	Verifies that an FMT operation was successfully completed. This switch is the default.
Write Last Track	/WLT=n	Rewrites the MDBSF (on the last track of the device) to add bad sectors found during FMT operation.

To terminate FMT, press CTRL/Z following the FMT prompt, as explained in Chapter 1 of this manual.

9.1.1 FMT Switch Descriptions

The following sections describe the switches you can use with FMT. The descriptions include information on restrictions for formatting specific devices and default values for the switches, where appropriate.

9.1.1.1 Bad

The Bad switch (/BAD) spawns the Bad Block Locator task (BAD) after FMT completes its processing. BAD 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. The RSX–11M–PLUS operating system provides spawned tasks as a system generation option.

9.1.1.2 Density

The Density switch (/DENS) sets DY-type diskettes to either HIGH (double) or LOW (single) density, and it sets DU-type (RX33) diskettes to HIGH (double) density. The default is low density. (This switch can also use SINGLE and DOUBLE as options.)

Format

ddnn:/DENS=option

Parameters

ddnn

Specifies the device to be formatted.

option

Specifies one of the two densities, as follows:

- HIGH (or double)
- LOW (or single)

9.1.1.3 Error Limit

The Error Limit switch (/ERL) sets the 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.

Format

ddnn:/ERL=n

Parameter

n

Specifies the error limit. Any value for n greater than 0 or less than or equal to 256_{10} is valid.

9.1.1.4 Indirect Command File

In the method of operation that uses the Indirect Command File switch (/@Y), FMT will not generate any operational messages or warnings to your terminal. No user intervention is possible until the FMT operation is complete.

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 not run.

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) that contains the following:

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

Then issue the following command from the prompt:

```
>@FMTIND RET
```

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

9.1.1.5 Manual

The Manual switch (/MAN) 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 number sign (#), which indicates octal values.

Manual operating mode cannot be used with DY-type devices.

In manual operating mode, FMT displays the following message and prompts:

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

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

Operation complete
```

9.1.1.6 Noverify

The Noverify switch (/NOVE or /-VE) inhibits the operation performed by the default /VERIFY switch (/VE).

9.1.1.7 Override

The Override switch (/OVR) causes FMT to ignore the MDBSF on DL-, 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 /OVR switch inhibits the reporting operation.

9.1.1.8 Verify

The Verify switch (/VE) confirms that an FMT operation was successful. It does this by reading back the headers and by determining if they were written correctly. This switch is the default.

9.1.1.9 Write Last Track

The Write Last Track (/WLT) switch, when used with the /VE 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.

Note

The /VE switch is the default. The /NOVE switch should not be specified.

The /WLT switch must be specified when using FMT on a DL-type device.

Format

ddnn:/WLT=n

Parameters

ddnn

Specifies the volume you are formatting.

n

Specifies the volume's pack serial number (in decimal).

9.2 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:

```
FMT -- Error writing header
```

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

```
FMT -- Error reading header
```

or

```
FMT -- Header compare error
```

FMT then continues the verification operation.

9.2.1 Normal Operating

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

```
** WARNING - Data will be lost on ddnn: **
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 following message:

```
Start formatting
```

It then performs the formatting functions you specify with switches in the FMT command line. 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 9.3).

9.2.2 Manual Operating

If you specify manual operating mode (using the /MAN switch), FMT prints the following warning:

```
** WARNING - Data will be lost on ddnn: **
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 following message:

```
Entering manual mode
```

It then displays the following prompts:

```
Cylinder=
Track   =
Sector  =
```

After you enter your response to the prompts, FMT formats the sector or track you specify. FMT assumes the responses are in decimal unless they are preceded by a number sign (#) 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 9–2 lists the valid ranges for FMT manual mode operations.

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

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.

Example

```
FMT>DMO:/MAN [RET]
```

Causes FMT to prompt with the following message:

```
** WARNING - Data will be lost on DMO: **
Continue? [Y or N] Y [RET]
Entering manual mode
Cylinder= 237 [RET]
Track   = 1 [RET]
```

Formats the entire track you specified.

Table 9–2: Ranges for Manual FMT Operations

Device	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

9.3 FMT-Supported Disk Volumes

The following sections describe using normal FMT operating mode with the different types of FMT-supported devices. Table 9–3 lists the disk volumes that allow formatting and their device mnemonics.

Table 9–3: FMT-Supported Disk Volumes

Disk Volumes	Device Mnemonic
RP04 disk pack	DB
RP05 disk pack	DB
RP06 disk pack	DB

Table 9–3 (Cont.): FMT-Supported Disk Volumes

Disk Volumes	Device Mnemonic
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
RX33 diskettes	DU
RX02 diskettes	DY

The status FMT requires for the disk volumes varies with the operating system.

On an RSX–11M–PLUS operating system, the disk volume must be mounted with foreign characteristics by using the MOUNT/FOREIGN command.

9.3.1 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 No Verify switch (/-VE), 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.

9.3.2 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 it reports 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.

9.3.3 DL-Type Devices (RL01/RL02 Disk Cartridges)

FMT does not format a DL-type volume, but it 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 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.

9.3.4 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.

9.3.5 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.

9.3.6 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 that the manufacturer's formatter designated as "skipped sectors," and then it 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-PLUS I/O Drivers Reference Manual*).

9.3.7 DU-Type Devices (RX33 Diskettes)

To format an RX33 diskette as an RX33, the unit must be mounted foreign. The DENSITY switch (/DENS) is the only switch that is applicable for formatting an RX33 and HIGH is the only density for RX33 diskettes.

Format

DUnn:[/DENS=HIGH]

The actual formatting of the RX33 will be done by the controller unlike the formatting of other devices, which is done by FMT.

9.3.8 DY-type Devices (RX02 Diskettes)

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

Format

FMT DYnn:[/DENS=option]

Parameter

option

Specifies one of the two densities, as follows:

- HIGH (or double)
- LOW (or single)

Manual operating mode cannot be used with DY-type devices.

9.4 FMT Error Messages

Error messages describe a condition that caused FMT to terminate the processing of a command. When this occurs, FMT returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, FMT will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>FMT commandline
FMT -- message
>
```

If the command is entered in response to the FMT prompt, FMT will issue an error message and prompt you with the FMT prompt, as follows:

```
FMT>commandline
FMT -- message
FMT>
```

This section describes the error messages FMT issues and lists possible user responses.

FMT—Command I/O error

Explanation: A hardware transmission error occurred from the keyboard.

User Action: Reenter the command.

FMT—Command too long

Explanation: The command was longer than 80_{10} characters.

User Action: Enter a shorter command.

FMT—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.

FMT—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.

FMT—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 line.

FMT—Device not ready

Explanation: The disk volume was not at operating speed when FMT attempted to access it.

User Action: Allow the volume to reach operating speed, and then reenter the FMT command.

FMT—Device offline

Explanation: The device is not in the system hardware configuration.

User Action: Ensure that the device name is correct, put the device on line, and then reenter the command line.

FMT—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.

FMT—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.

FMT—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.

FMT—Error reading data

Explanation: FMT encountered an error when it tried to read data from a disk.

User Action: No user action is required. FMT retries the operation and continues the verification.

FMT—Error setting diskette density

Explanation: FMT tried to format an RX02 diskette but the operation failed.

User Action: Check the syntax and reenter the command with the density reset.

FMT—Error writing data

Explanation: FMT encountered an error when it tried to write sector headers.

User Action: No user action is required. FMT retries the operation and continues the verification.

FMT—Error writing header

Explanation: FMT encountered an error when it tried to write a header.

User Action: No user action is required. FMT retries the operation.

FMT—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.

FMT—Failed to format - Volume has been altered

Explanation: An error occurred while the disk was being formatted. The FMT operation did not complete, but the volume may have been altered.

User Action: Reenter the FMT command line.

FMT—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: Include the /OVR switch and reenter the command.

FMT—Fatal hardware error

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

User Action: Contact your DIGITAL Field Service representative.

FMT—Header compare error

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

User Action: No user action is required. FMT tries the operation again.

FMT—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.

FMT—Invalid value for the /DENS switch

Explanation: The value for the /DENS switch was invalid for the device specified.

User Action: Refer to Section 9.1.1.2 for information on options for the /DENS switch. Specify the correct density and reenter the command line.

FMT—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 /OVR switch to prevent FMT from trying to use the corrupt bad block data.

FMT—Marked bad in Manufacturer's Bad Sector File

Explanation: Indicates that bad block information is recorded in the MDBSF on the disk.

User Action: No user action is required. This message is for your information only.

FMT—Marked defect in Skip Sector File

Explanation: The sector flagged previous to this message was marked as bad in the Skip Sector File.

User Action: No user action is required.

FMT—Media format error - Volume has been altered

Explanation: A transmission error occurred while the volume was being formatted. The FMT operation has not completed, but the volume may have been altered.

User Action: Reenter the FMT command line.

FMT—Parity error on device

Explanation: A parity error occurred during the FMT operation.

User Action: Reenter the command line. If the error persists, the volume may need to be replaced.

FMT—Privilege violation

Explanation: FMT attempted an operation on a device that was mounted or allocated to another user, was mounted Files–11, or was not mounted foreign.

User Action: Ensure that the device is mounted foreign by the user and reenter the command line.

FMT—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 9–2, (Section 9.2.2) for valid parameters and reenter the command.

FMT—Skip Sector File is corrupt, reading next

Explanation: The current copy of the Skip Sector File is corrupt. FMT ignores the copy and proceeds to the next copy.

User Action: No user action is required.

FMT—Syntax error

Explanation: FMT detected a syntax error in the command line.

User Action: Determine the correct command syntax and reenter the command.

FMT—Unable to read any Skip Sector File

Explanation: FMT was unable to read any copy of the Skip Sector File found on the first track of the CE cylinder area on the RM80 disk.

User Action: No user action is required.

FMT—Unable to run badblock utility

Explanation: An FMT command specified the /BAD switch, but the Bad Block Locator task (BAD) could not be spawned. Either the operating system does not spawn tasks or BAD is not installed.

User Action: Run BAD separately (see Chapter 2 for more information).

FMT—Unrecoverable error - n

Explanation: An I/O error (n) caused FMT to terminate.

User Action: Reenter the FMT command and, if the error occurs again, try the command with a different device specified, or refer to the error codes in the *RSX-11M-PLUS and Micro/RSX I/O Operations Reference Manual.*

FMT—Vector not multiple of four

Explanation: The message occurs only when using standalone BRU (BRUSYS).

The vector specified was not a multiple of four.

User Action: Specify the correct vector and reenter the command line.

FMT—Verify operation of FMT failed

Explanation: FMT detected that the controller initiated format was not sucessful and that some of the blocks on the volume were not readable.

User Action: Reenter the command line. If the error persists, the volume may need to be replaced.

FMT—WLT switch is missing

Explanation: The /WLT switch was not specified in the command line for formatting a DL-type device.

User Action: Specify the /WLT switch and reenter the command line.

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 with the same file type, into library modules so that you have rapid and convenient access to your files.

10.1 Library 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 ordered alphabetically.

The following sections describe the three types of libraries: object, macro, and universal.

10.1.1 Object Library Files (OLB)

Object library files contain object files (OBJ type). 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 (TKB). 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.

10.1.2 Macro Library Files (MLB)

Macro library files contain source macro files (MAC type). The module names are derived from .MACRO directives. From each macro definition, LBR extracts the name and creates an entry in the MNT. The entry in the MNT 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.

10.1.3 Universal Library Files (ULB)

Universal library files contain modules inserted from any kind of file, whether it be a program or text. The module names are either user specified with the Insert switch (/IN) 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. This routine, 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 *RSX–11M–PLUS and Micro/RSX System Library Routines Reference Manual.*)

You can invoke LBR by using any of the methods for invoking a utility, as described in Chapter 1.

10.1.4 Format of Library Files

A library file consists of a library header, an EPT, an MNT, the library modules and their headers, and any available space. The EPT has zero length for macro and universal libraries. Figure 10–1 illustrates object and macro library file formats, and Figure 10–2 illustrates universal library file format.

10.1.4.1 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, to insert, compress, and delete files). 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.

10.1.4.2 Entry Point Table

The entry point table (EPT) consists of 4-word elements that contain an entry point name (words 0 to 1) and a pointer to the module header of the module where the entry point is defined (words 2 to 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 EPT applies only to object library files. The Byte in Block in Figure 10–4 is the record size word followed by the module header (see Figures 10–6 and 10–7).

10.1.4.3 Module Name Table

The module name table (MNT) 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 two 4-word elements: a module name (words 0 to 1) and a pointer to the module header (words 2 to 3). (See Figure 10–5.) The MNT presents the module names in ascending order. The Byte in Block in Figure 10–5 is the record size word followed by the module header (see Figures 10–6 and 10–7).

10.1.4.4 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 Figures 10–6 and 10–7.

The File Descriptor Block (FDB) of the original file from which the module was created has five sections of information; the first is "file attributes." These attributes are as follows:

- Record type
- Record attribute
- Record size
- Highest virtual block
- End-of-file (EOF) block number
- Optional information

When you create a file and insert it into a universal library, LBR copies the input file attributes to the module header. You can modify some of these attributes by using the Modify Header switch (/MH).

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 /MH switch.

Figure 10–1: General Format for Object and Macro Library Files

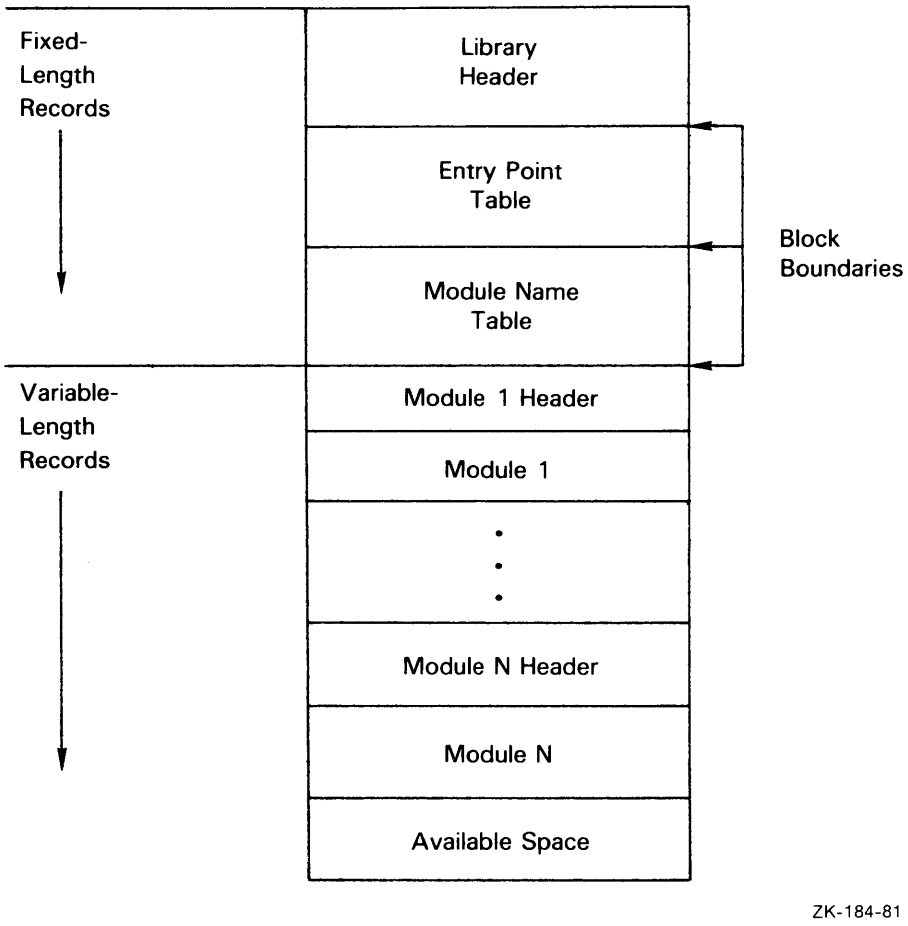
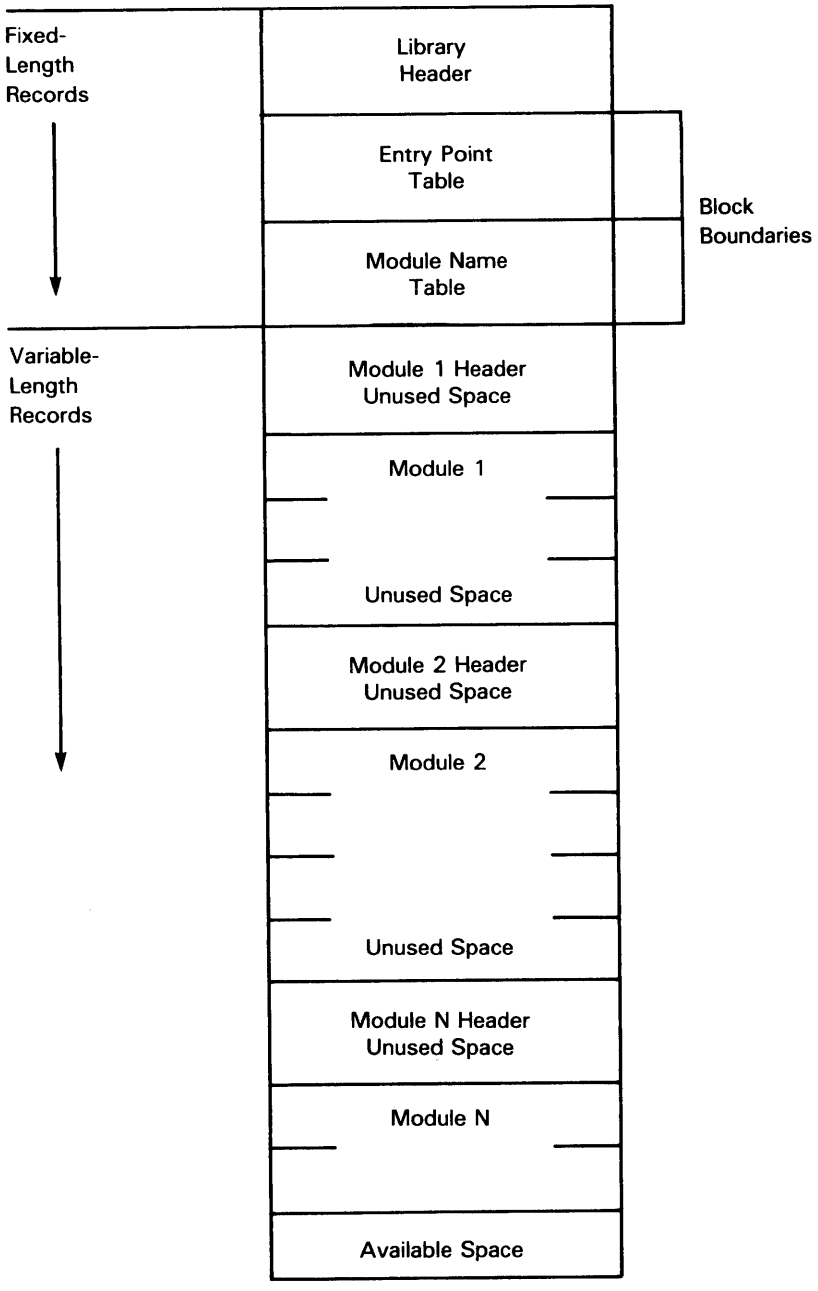


Figure 10–2: Universal Library File Format

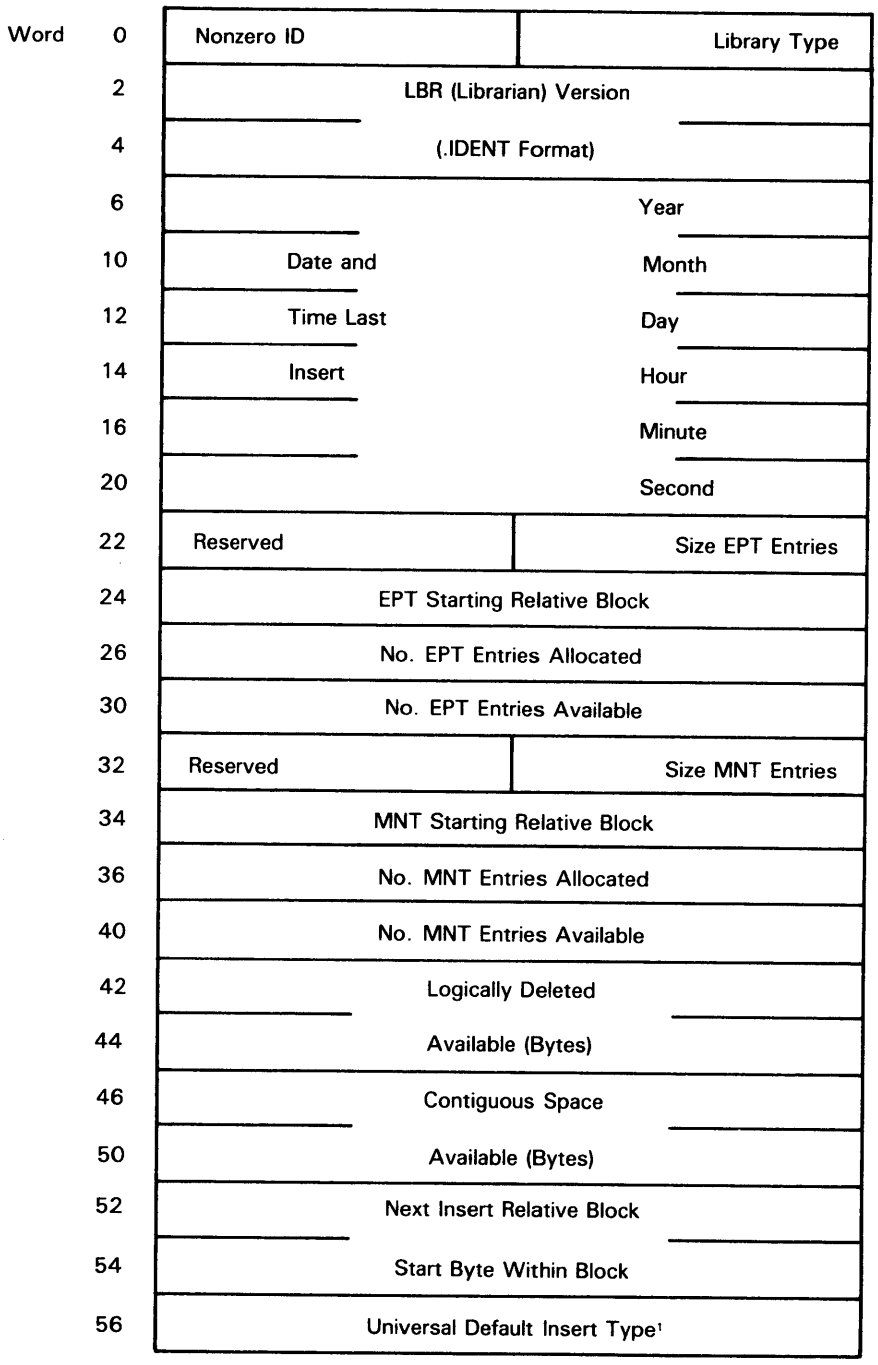


NOTE

All universal module headers and the first record of each universal module will start on a block boundary.

ZK-185-81

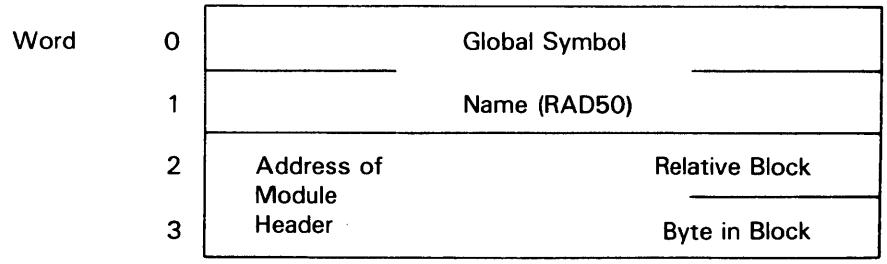
Figure 10–3: Contents of Library Header



[1] Undefined for Macro and Object Libraries

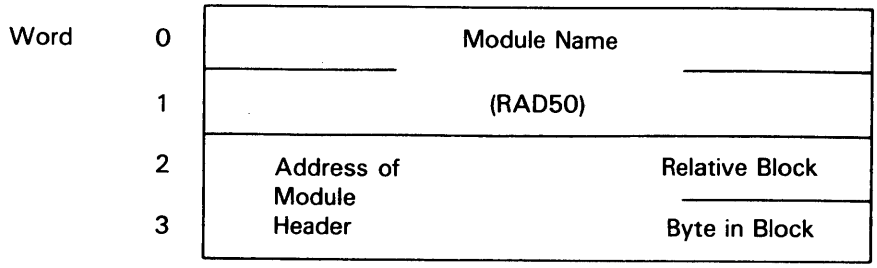
ZK-186-81

Figure 10–4: Format of Entry Point Table (EPT) Element



ZK-187-81

Figure 10–5: Format of Module Name Table (MNT) Element



ZK-188-81

Figure 10–6: Module Header Format for Object and Macro Libraries

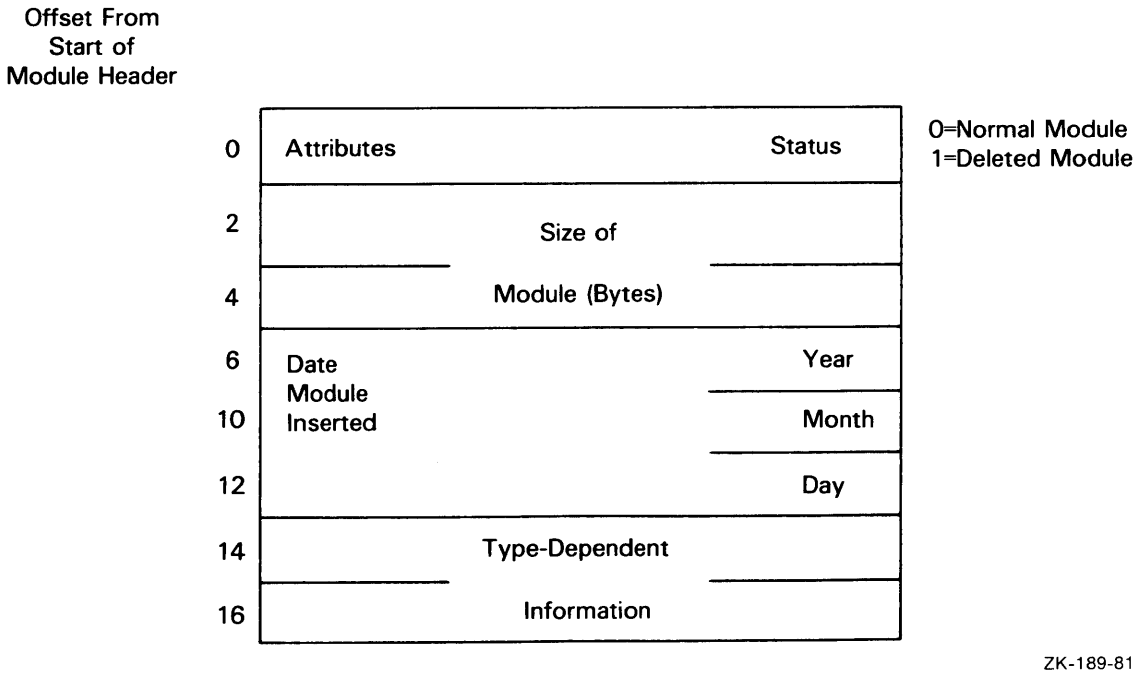
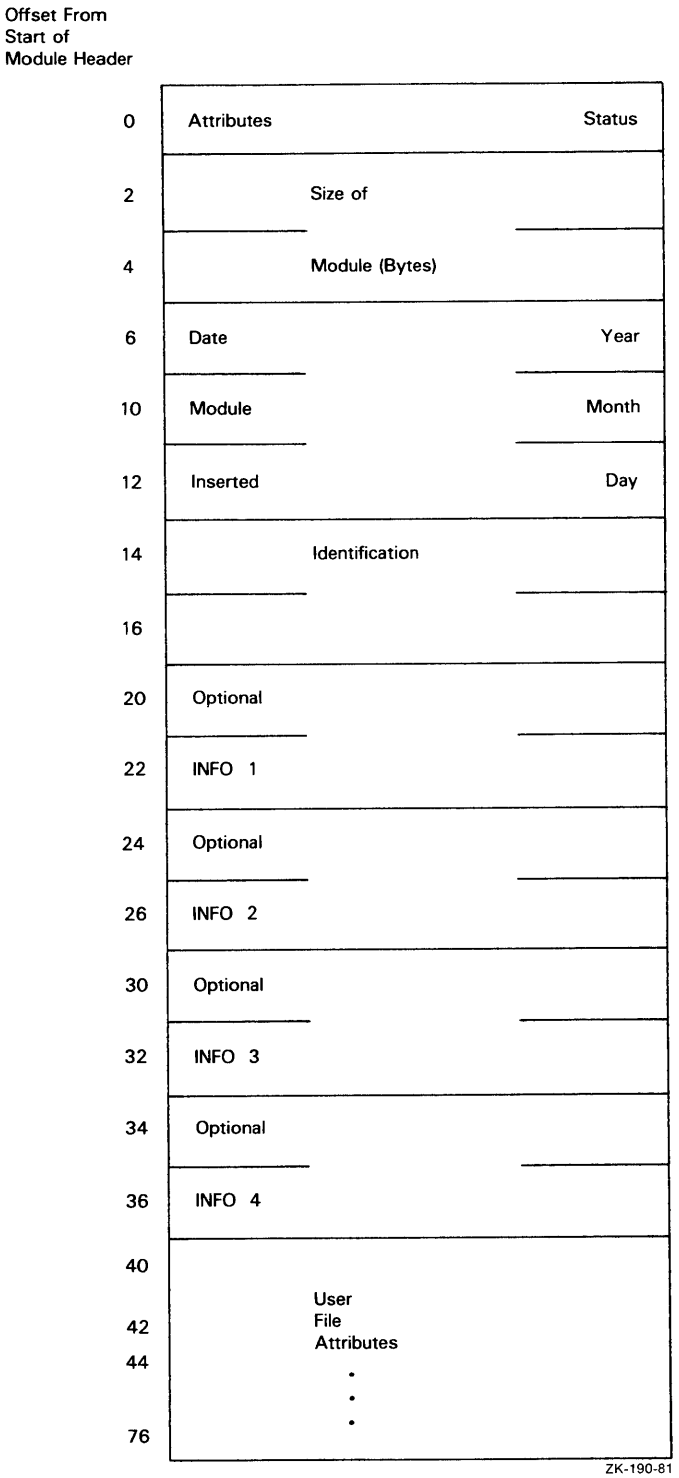


Figure 10–7: Module Header Format for Universal Libraries



10.2 LBR Restrictions

The following restrictions apply when using LBR:

- Limit of 65,536 (64_{10}K) words per module.
- Limit of 65,536 (64_{10}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, but with the same entry point, as the inserted module. For further information, refer to the discussion of the Insert switch (/IN) in Section 10.3.2.8.
- The use of wildcards in file specifiers is not allowed (that is, forms such as *.OBJ, where the asterisk (*) 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

You can invoke LBR by using any of the methods described in Chapter 1.

Note

LBR allows only one level of indirect command file nesting.

The general command line for LBR is shown next.

Format

outfile[,listfile]=infile[,...]

The format for entering file specifications is as follows:

```
ddnn:[directory]filename.type;version[/switch]
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

The LBR switches are summarized in Table 10–1.

Table 10–1: LBR Switches

Switch	Format	Function
Compress	/CO	Compresses a library file.
Create	/CR	Creates a library file.
Delete	/DE	Deletes a library module and all of its entry points.
Default	/DF	Specifies the default library file type.
Delete Global	/DG	Deletes a library module entry point.

Table 10–1 (Cont.): LBR Switches

Switch	Format	Function
Entry Point	/EP	Includes entry point elements in the library EPT.
	/-EP	Excludes entry point elements in the library EPT.
Extract	/EX	Extracts (reads) one or more modules from a library file and writes them into a specified output file.
Insert	/IN	Inserts a module.
List	/LI	Lists module names.
	/LE	Lists module names and module entry points.
	/FU	Lists module names and full module description.
Modify Header	/MH	Modifies a universal module header.
Replace	/RP	Replaces a module.
	/-RP	Does not replace a module.
Spool	/SP	Spools the listing for printing.
	/-SP	Does not spool the listing.
Selective Search	/SS	Sets the selective search attribute in the module header.
Squeeze	/SZ	Reduces the size of the macro source.
	/-SZ	Does not reduce the size of a specific macro source.

10.3.1 Defaults for LBR File Specifications

Table 10–2 describes the defaults for LBR file specifiers.

Table 10–2: LBR File Specifiers Defaults

Specifier	File	Default
ddnn	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.
directory	Output file	The directory under which LBR is running currently.
	Listing file	The directory that was specified for the output file; otherwise, the default for the output file specifier.

Table 10–2 (Cont.): LBR File Specifiers Defaults

Specifier	File	Default
	Input file	For the first input file specifier, the directory under which LBR is running currently.
		For subsequent input file specifiers, the directory 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 10.3.2.4), except when the /CO or /CR switch is specified (see Sections 10.3.2.1 and 10.3.2.2).
	Listing file	LST
	Input file	Refer to the descriptions of the /CO (Section 10.3.2.1), /IN (Sections 10.3.2.8 and 10.3.2.9), and /RP (Sections 10.3.2.12 and 10.3.2.13) switches.
version		Latest version of the file, or latest version plus 1 for the output file when the /CO, /CR, or /EX switch is specified.
switch	Output file	/IN (insert)
	List file	/SP/LI (spool and list module names)
	Input file	None.

10.3.2 LBR Switch Descriptions

LBR uses switches appended to file specifications to invoke functions. These switches are summarized in Table 10–1.

10.3.2.1 Compress

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.

Format

outfile/CO:size:ept:mnt=infile

Parameters

outfile

Specifies the file specification 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.

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 specification 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 10.3.2.4).

Example

```
LBR>RICKLIB/CO:100.:128.:64.=SHEILA.OLB [RET]
```

Compresses file SHEILA.OLB and creates a new file, RICKLIB.OLB, 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.

10.3.2.2 Create

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 EPT, and the MNT.

The /CR switch can be appended to the output file specification only.

Format

outfile/CR:size:ept:mnt:libtype:infiletype

Parameters

outfile

Specifies the library file specification 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.

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 Default switch (/DF) (see Section 10.3.2.4), or it is OBJ if the /DF switch 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.

Note

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 RET
```

Performs a combination of functions. First, the library file RICKLIB.OLB is created in the default directory on SY0. RICKLIB has the following attributes:

size = 100_{10} blocks (default size)
ept = 128_{10} entry points
mnt = 64_{10} module names
type = OBJ

Second, object modules from the input files SHEILA.OBJ, LAURA.OBJ, and JENNY.OBJ, which reside in the default directory on SY0, are inserted into the newly created library file. The Insert switch (/IN) is the default switch for input files (see Section 10.3.2.8).

10.3.2.3 Delete

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:n
```

As modules are deleted logically 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. The following message appears:

```
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 it 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 10.3.2.1).

Format

outfile/DE:module[...]

Parameters

outfile

Specifies the library file specification.

module

Specifies the name or names of the module or modules to be deleted.

Example

```
LBR>RICKLIB/DE:SHEILA:LAURA:JENNY RET
MODULES DELETED:
SHEILA
LAURA
JENNY
```

Deletes the modules SHEILA, LAURA, and JENNY and their associated entry points from the latest version of library file SY0:RICKLIB.OLB.

10.3.2.4 Default

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 results in the following:

- Sets the default file type for the /CR switch.
- 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 the /CO or /CR switch. When you specify the /CO switch, the default applies to the library input file. When you specify the /CR switch, 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 file type 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.

Format

outfile/DF:filetype

or

/DF:filetype

Parameters

outfile

Specifies the library file specification.

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 [RET]
LBR>RICKLIB= [RET]
```

Opens file RICKLIB.MLB for insertion.

```
LBR>/DF:MAC [RET]
LBR>RICKLIB/DF:OBJ=infile [RET]
```

Replaces MAC with OBJ as the default file type and opens file RICKLIB.OLB for insertion.

```
LBR>/DF:MAC [RET]
LBR>RICKLIB/CR [RET]
```

Creates the macro library RICKLIB.MLB.

```
LBR>/DF:MAC [RET]
LBR>RICKLIB/CR::::OBJ [RET]
```

Shows OBJ overriding the default (MAC) because RICKLIB.MLB is a macro library, and creates the object library RICKLIB.OLB.

```
LBR>/DF:OBJ [RET]
LBR>TEMP/CO=RICKLIB.MLB [RET]
```

Shows MAC overriding the default (OBJ) because RICKLIB.MLB is a macro library, and creates the macro library file TEMP.MLB to receive the compressed output.

```
LBR>/DF:UNI [RET]
LBR>RICHLIB=TEST [RET]
```

Opens file RICHLIB.ULB for insertion.

10.3.2.5 Delete Global

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 follows:

```
LBR -- *FATAL* - No entry point named "name"
```

The /DG switch can only be appended to the library file specification.

Format

outfile/DG:global[...]

Parameters

outfile

Specifies the library file specification.

global

Specifies the name or names of entry point or points to be deleted.

Example

```
LBR>RICKLIB/DG:SHEILA:LAURA:JENNY [RET]
ENTRY POINTS DELETED:
SHEILA
LAURA
JENNY
```

Deletes the entry points SHEILA, LAURA, and JENNY from the latest version of the library file named SY0:RICKLIB.OLB.

10.3.2.6 Entry Point

Use the Entry Point switch (/EP) to control (include or exclude) the placement of global symbols in a library entry point table (EPT).

Format

outfile/EP=infile[,...]

or

outfile=infile/EP[,...]

Parameters

outfile

Specifies the output file specification. When the /EP 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 specification. 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.

The /EP switch can be specified in either a positive or negative format as follows:

/EP Includes entry points in the EPT.

/-EP Does not include entry points in the EPT.
/NOEP

The positive switch format (/EP) causes all entry points in a module or modules to be entered in the library EPT.

Either negative switch 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 EPT.

The /EP switch is the LBR default. If the switch is not specified, all entry points are entered into the library EPT.

The /EP switch has no effect on macro or universal libraries.

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 the /EP switch applied to the output file can be overridden by applying the /-EP switch 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 EPT.

The /-EP switch is also useful in the case where TKB uses only module names to search for modules in an object module library. In this case, entries in the library EPT are not required. The /-EP switch can be used to exclude entry points from being entered in the library EPT.

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 EPT or excludes all entry points from being entered in the EPT.

When applied to an input file specification, the /EP switch has a local effect. That is, LBR either includes entry points in the EPT or excludes entries from being entered in the EPT 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 EPT.

10.3.2.7 Extract

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, Record Management Services (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.

Format

```
outfile=infile/EX[:modulename[...]]
```

Parameters

outfile

Specifies the file specification into which extracted modules are to be stored. If the input modules are object modules, the default file type for this file is OBJ. If the input modules are macro modules, the default file type is MAC. 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 specification 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.

modulename

Specifies the name or names of the module or modules to be extracted from the library.

Examples

```
LBR>DRIVERS=RSX11M/EX:DXDRV:DKDRV:TTDRV [RET]
```

Concatenates the object modules DXDRV, DKDRV, and TTDRV in alphabetical order and writes them into the file DRIVERS.OBJ.

```
LBR>TI:=[1,1]RSXMAC.SML/EX:QIO$S [RET]
```

Writes the macro QIO$S to the issuing terminal.

```
LBR>TEST.OBS=TEST/EX [RET]
```

Writes all of the modules in the library TEST.OLB into the file TEST.OBS in alphabetical order.

10.3.2.8 Insert Switch 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 the first-level macro definitions is ignored. (The /IN switch for Universal Libraries, is explained in Section 10.3.2.9.) 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
```

Format

outfile/IN=infile[,...]

Parameters

outfile

Specifies the library file specification into which the input modules are to be inserted. The default file type depends on the current default (see Section 10.3.2.4). It is OLB if the current default is object libraries, and it is MLB if the current default is macro libraries.

infile

Specifies the input file specification 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 [RET]
```

Inserts the modules contained in the latest versions of files SHEILA, LAURA, and JENNY, which reside in the default directory on SY0, 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.

10.3.2.9 Insert Switch for Universal Libraries

The Insert switch (/IN) works 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 (EOF) indicator (F.EFBK of the file's File Descriptor Block (FDB)) are included in the input file's user file attributes. LBR makes the high block indicator equal to the EOF 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.

Format

outfile=infile/IN:name[:op[...]]

Parameters

outfile

Specifies the universal library file specification into which the infile is to be inserted.

infile

Specifies the input file specification 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 10.3.2.2.)

name

Specifies the module name (up to six Radix–50 characters) optionally. 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 [RET]
```

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.

10.3.2.10 List Switches

Use the list switches (/LI, /LE, and /FU) to produce a printed listing of the contents of a library file. Three switches allow you to select the type of listing desired. The functions of these switches are as follows:

/LI Lists the names of all modules in the library file.

/LE Lists the names of all modules in the library file and their corresponding entry points.

/FU Lists 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.

Format

outfile[,listfile]/switch

Parameters

outfile

Specifies the library file specification whose contents are to be listed.

listfile

Specifies the listing file specification optionally. If not specified, the listing is directed to the initiating terminal.

switch

Specifies the list option or options selected.

Examples

```
LBR>RICKLIB/LI [RET]
```

Prints a listing of the names of all the modules contained in file SY0:RICKLIB.OLB on the initiating terminal.

```
LBR>RICKLIB/LE [RET]
```

Prints a listing of the names of all the modules and their entry points (contained in file SY0:RICKLIB.OLB) on the initiating terminal.

```
LBR>RICKLIB/FU [RET]
```

Prints a listing of the names of all the modules in file SY0:RICKLIB.OLB and a full description of each module on the initiating terminal.

```
LBR>DK1:[200,200]RICKLIB,LP.LST/LE/FU [RET]
```

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.

10.3.2.11 Modify Header

The Modify Header switch (/MH) pertains only to universal libraries and allows you to modify the optional user-specified information in the module header.

Format

outfile/MH:module[:op[...]]

Parameters

outfile

Specifies an output file specification for the universal library. The file type defaults to ULB.

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 number sign (#) clears the corresponding information field.

Example

```
LBR>RICKLIB/MH:A:FCHTS:#:: [RET]
```

Changes the optional descriptive information for module A of RICKLIB.ULB.

The optional descriptive information for module A of RICKLIB.ULB was as follows:

```
"MODA" "FCHCD" "OF" "FCH"
```

The previous command line changes the optional descriptive information for module A to the following:

```
"FCHTS" " "OF" "FCH"
```

10.3.2.12 Replace Switch 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 letters 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	Appends the /RP switch to the library file specification. Assumes all of the input files contain replacement modules.
Local format	Appends the /RP switch to an input file specification. Considers only the file appended with the /RP switch contains replacement modules.

Global Format

outfile/RP=infile[,...]

Parameters

outfile

Specifies the library file specification. The default file type depends on the current default (see Section 10.3.2.4). It is OLB if the current default is object libraries or MLB if the current default is macro libraries.

infile

Specifies the input file specification 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 the /-RP or /NORP switch to the desired input file specification.

Local Format

outfile=infile[/RP][,...]

Parameters

outfile

Specifies the library file specification. The local format default is the same as the global format default.

infile

Specifies the input file specification that contains replacement modules for the output library file. The local format default is the same as the global format default.

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 or modules contained in the specified file as replacement modules.

Examples

The output library file used in the following four examples contains the following object modules:

Output Library File	Object Modules
RICKLIB.OLB;1	JENNY1 JENNY2 LAURA1 LAURA2 SHEILA

The input files used in the following four examples contain the modules listed as follows:

Input File	Object Modules
SHEILA.OBJ;1	SHEILA
LAURA.OBJ;1	LAURA1 LAURA2 LAURA3
JENNY.OBJ;1	JENNY1 JENNY2 JENNY3
CHRIS.OBJ;1	CHRIS1 CHRIS2

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 previously.

1. The following command line uses the global format for the /RP switch. 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.

```
LBR>RICKLIB/RP=SHEILA,LAURA,JENNY [RET]
MODULE "SHEILA" REPLACED
MODULE "LAURA1" REPLACED
MODULE "LAURA2" REPLACED
MODULE "JENNY1" REPLACED
MODULE "JENNY2" REPLACED
```

Produces the following library file:

Output Library File	Object Modules
RICK.OLB;1	JENNY1 JENNY2 JENNY3[1] LAURA1 LAURA2 LAURA3[1] SHEILA

[1]These modules did not exist in the library file prior to the execution of Example 1, but they did exist in the input files. LBR assumed they were to be inserted. Since LBR handled these modules as a normal insert, no message was printed on the input terminal.

2. The following command line uses the local format of the /RP switch. 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 10.3.2.8.)

```
LBR>RICKLIB=CHRIS,SHEILA/RP [RET]
MODULE "SHEILA" REPLACED
```

Produces the following library file:

Output library file	Object modules
RICK.OLB;1	CHRIS1[1] CHRIS2[1] JENNY1 JENNY2 LAURA1 LAURA2 SHEILA[2]

[1]These modules are inserted.

[2]This module is replaced.

3. The following command line uses the /-RP switch 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.

```
LBR>RICKLIB/RP=SHEILA,LAURA,JENNY,CHRIS/-RP [RET]

MODULE "SHEILA" REPLACED
MODULE "LAURA1" REPLACED
MODULE "LAURA2" REPLACED
MODULE "JENNY1" REPLACED
MODULE "JENNY2" REPLACED
```

Produces the following library file:

Output Library File	Object Modules
RICKLIB.OLB;1	CHRIS1[1] CHRIS2[1] JENNY1 JENNY2 JENNY3[2] LAURA1 LAURA2 LAURA3[2] SHEILA

[1] These modules were specified to be inserted. If a module of the same name was present, a fatal error would have been issued (refer to Example 4).

[2] These modules were inserted by default.

4. The following command line replaces module SHEILA from file SHEILA. The user specifies that the modules in file LAURA are not to be replaced (/NORP) but inserted. One of the modules contained in file LAURA duplicates an already existing module in file RICKLIB. Therefore, LBR issues the fatal error message and terminates the processing of the current command line.

```
LBR>RICKLIB/RP=SHEILA,LAURA/NORP,JENNY [RET]

MODULE "SHEILA" REPLACED
LBR -- *FATAL* - Duplicate module "LAURA1" in LAURA.OBJ;1
```

10.3.2.13 Replace Switch for Universal Libraries

Use the Replace switch (/RP) 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 10.3.2.9).

As with macro and object libraries, you can specify the /RP switch with either the output file specification or the input file specifications.

Global Format

outfile/RP:name[:op[...]]=infile[,...]

Local Format

outfile=infile/RP:name[:op[...]][,...]

Parameters

outfile

Specifies the universal library file specification.

infile

Specifies the input file specification 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 10.3.2.2).

name

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

Output Library File	Modules
TEXT.ULB;1	DEBBIE BERNIE

Shows the initial state of the library file.

Input File	Modules
DEBBIE.TXT	

Shows the contents of the input file.

```
LBR>TEXT.ULB=DEBBIE.TXT/RP::THIS:IS:JAN3:UPDATE [RET]

MODULE "DEBBIE" REPLACED
```

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 is inserted in the module header. Note that the optional name is omitted.

The resulting library file is as follows:

Output library file	Modules
TEXT.ULB;1	DEBBIE[1] BERNIE

[1]The module DEBBIE was replaced. If a different input file had been specified, that file would have become module DEBBIE and would have occupied the same location in TEXT.ULB.

10.3.2.14 Selective Search

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 (TKB). Global symbols, defined in modules with the selective search attribute, are included in TKB's symbol table only if they are previously referenced by other modules. Thus, only referenced symbols will be listed with the module in TKB 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-PLUS and Micro/RSX Task Builder Manual* for more information.

Format

outfile=infile/SS[,...]

Parameters

outfile

Specifies the library file specification.

infile

Specifies the input file specification that contains modules to be selectively searched.

Example

```
LBR>ANGEL=JOHN,JILL/SS,MARK/SS,MARY [RET]
```

Inserts the object files JOHN.OBJ, JILL.OBJ, MARK.OBJ, and MARY.OBJ into object library ANGEL.OLB. The selective search attribute bit is set in both the JILL and MARK object module header.

10.3.2.15 Spool

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-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.

Format

outfile,listfile/SP

or

outfile,listfile/-SP

Parameters

outfile

Specifies the library file specification.

listfile

Specifies the listing file specification.

Example

```
LBR>RICKLIB/DE:SHEILA,RICKLST/-SP [RET]
```

Causes the following two actions:

- Deletes the module SHEILA and its associated entry points from the library file SY:RICKLIB.
- Writes the listing of the contents of the resulting library file RICKLIB to the list file SY:RICKLST.LST. Because the /-SP switch is specified, the file is not automatically printed.

10.3.2.16 Squeeze

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 the two following formats:

Global format	Appends the /SZ switch to the library file specification. Assumes all of the input files contain modules to be squeezed.
Local format	Appends the /SZ switch to an input file specification. The /SZ switch works only on the file to which you append it.

Global Format

outfile/SZ=infile[,...]

Local Format

outfile=infile/SZ[,...]

Parameters

outfile

Specifies the library file specification.

infile

Specifies the input file specification 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 the /-SZ or /NOSZ switch to the desired input file specification.

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.

Examples

```
        .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
```

Shows the contents of the input file.

```
        .MACRO  MOVSTR RX,RY,?LBL
;***    - - NOTE :
;       BOTH ARGUMENTS MUST BE REGISTERS
LBL:    MOVB    (RX)+,(RY)+
        BNE     LBL
        DEC     RY
        .ENDM
```

Shows the actual text inserted into the library file after the input file was squeezed by using the /SZ switch.

10.4 Combining Library Functions

Two or more library functions may be requested in the same command line. The only exceptions are that the /CO switch cannot be specified with anything else except the /LI switch, and the /CR and /DE switches 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), and Entry Point (/EP) switches
7. Compress switch (/CO))
8. Extract switch (/EX)
9. List switches (/LI, /LE, and /FU) and Spool switch (/SP)

Example

```
LBR>FILE/DE:XYZ:$A,LP.LST:/LE/FU=MODX,MODY/RP [RET]
```

Performs the following functions in order:

1. Deletes modules XYZ and $A.
2. Inserts all modules from MODX and replaces duplicate modules of MODY.
3. Produces a listing of the resultant library file on the line printer with full module descriptions and all entry points.

10.5 LBR 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 following 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 Monitor Console Routine (MCR) prompt, LBR will issue the fatal error message, exit, and prompt you with the MCR prompt, as follows:

```
>LBR commandline
LBR -- message
>
```

If the command is entered in response to the LBR prompt, LBR will issue a fatal error message and prompt you with the LBR prompt, as follows:

```
LBR>commandline
LBR -- message
LBR>
```

Fatal error messages are issued to your terminal in the following 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.

10.5.1 LBR Diagnostic Error Messages

The diagnostic error messages for LBR are as follows:

LBR—*DIAG* - 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 user action is required. The command will be executed as if the information had not been supplied.

LBR—*DIAG* - Mark for delete failure on LBR work file

Explanation: When LBR begins processing commands, it automatically creates a work file and marks it for deletion. For some reason, this marking for deletion 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 14 for information on VFY.

LBR—*DIAG* - 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—*DIAG* - 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 user action is required. LBR continues as though the extra file or files had not been specified.

10.5.2 LBR Fatal Error Messages

The status of a library file after fatal errors is as follows:

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.

The fatal error messages for LBR are as follows:

LBR—•FATAL• - 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 14).

If the volume is corrupted, it must be reconstructed before it can be used.

LBR—•FATAL• - 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: Specify a module in a universal library and reenter the command line.

LBR—•FATAL• - Command I/O error

Explanation: One of the following conditions may exist:

- A problem exists on the physical device (for example, it is 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—•FATAL• - Command syntax error
commandline

Explanation: A command was entered in a format that does not conform to syntax rules.

User Action: Use the correct syntax and reenter the command line.

LBR—•FATAL• - 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, specify the correct input file and reenter the command line. If the input file is the correct file, you can delete the duplicate entry point from the library and reenter the command line.

LBR—•FATAL• - 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—•FATAL• - 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 and increase the table space by means of the /CO switch. Reenter the command line.

LBR—•FATAL• - 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—•FATAL• - 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—•FATAL• - 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• - 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—•FATAL• - 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—•FATAL• - Illegal device/volume commandline

Explanation: The device specifier entered does not conform to syntax rules. A device specifier consists of two American Standard Code for Information Interchange (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—•FATAL• - Illegal directory commandline

Explanation: The directory entered does not conform to syntax rules. Proper directory syntax consists of a left square bracket ([), followed by one to three octal digits, a comma, and one to three octal digits, which terminates by a right square bracket (]), or it may consist of a named directory, which contains one to nine characters enclosed in brackets.

User Action: Reenter the command line with the correct directory syntax.

LBR—•FATAL• - Illegal filename commandline

Explanation: One of the following was entered:

- A file specifier that contained a wildcard.
- A file specifier that contained neither a file name nor a file type.

User Action: Reenter the command line correctly.

LBR—•FATAL• - 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 an SPR along with the related console dialog and any other pertinent information.

LBR—•FATAL• - Illegal switch commandline

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—•FATAL• - Illegal switch combination

Explanation: Switches were entered that cannot be executed in combination (see Section 10.4).

User Action: Specify the switches in the proper combination and reenter the command line.

LBR—•FATAL• - Indirect command syntax error
commandline

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—•FATAL• - Indirect file depth exceeded
commandline

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—•FATAL• - Indirect file open failure
commandline

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.
- There is insufficient dynamic memory in the Executive.

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

LBR—•FATAL• - 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: Find and fix the problem, and reenter the command line.

LBR—•FATAL• - 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 descriptions of the INSTALL command in the *RSX-11M-PLUS Command Language Manual* and *RSX-11M-PLUS MCR Operations Manual.*)

LBR—•FATAL• - Invalid EPT and/or MNT specification

Explanation: An EPT or MNT value greater than 4096_{10} was entered with the /CR or /CO switch.

User Action: Reenter the command line with the correct value specified.

LBR—•FATAL• - Invalid filetype

Explanation: The specified file type contained illegal RAD50 characters.

User Action: Specify a valid file type and reenter the command line.

LBR—•FATAL• - 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—•FATAL• - Invalid library type specified

Explanation: An invalid library type was specified when using the /CR or /DF switch. The values OBJ, MAC, and UNI are the only valid specifications. See Sections 10.3.2.2 and 10.3.2.4.

User Action: Reenter the command line with OBJ, MAC, or UNI specified.

LBR—•FATAL• - 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—•FATAL• - Invalid name "name"

Explanation: 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 to Z, the numbers 0 to 9, and the special characters period (.) and dollar sign ($).

User Action: Reenter the command line and specify a valid name.

LBR—•FATAL• - Invalid RAD50 character in "character string"

Explanation: A character supplied as part of information when using the /IN, /RP, or /MH switch for a universal library is not a Radix–50 character.

User Action: Determine which character of the corresponding switch value is not a Radix–50 character. Reenter a Radix–50 character in place of the invalid character.

LBR—•FATAL• - 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, it is 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—•FATAL• - 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—•FATAL• - Missing output file specifier

Explanation: The command line was issued without the output file specification.

User Action: Specify the output file and reenter the command line.

LBR—•FATAL• - 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—•FATAL• - 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—•FATAL• - 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 opening a file.
- 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.
- There is insufficient contiguous space to allocate the library file (see the /CO and /CR switches).
- There is insufficient dynamic memory in the Executive.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

LBR—•FATAL• - 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—•FATAL• - 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—•FATAL• - Privilege violation on file filename

Explanation: LBR was unable to open file because of privilege violations.

User Action: Reset privileges and reenter the command line.

LBR—•FATAL• - 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—•FATAL• - RMS module 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—•FATAL• - Too many input files

Explanation: The LBR switch /EX was used with more than one input file.

User Action: Specify one input file with the /EX switch and reenter the command line.

LBR—•FATAL• - 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, and 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—•FATAL• - 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
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 (TKB), either overlays or is appended to the input object module. Unlike TKB 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 TKB.

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.

11.1 PAT Command Line

You invoke PAT by using any of the methods for invoking a utility, as 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.

The command line for PAT is shown next.

Format

[outfile]=infile[/CS[:n]],correctfile[/CS:[n]]

Parameters

outfile

Specifies the file specification for the output file. The format for entering file specifications is as follows:

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

If you do not specify an output file, PAT does not generate one.

infile

Specifies the file specification for the input file. This file can contain one or more concatenated object modules.

correctfile

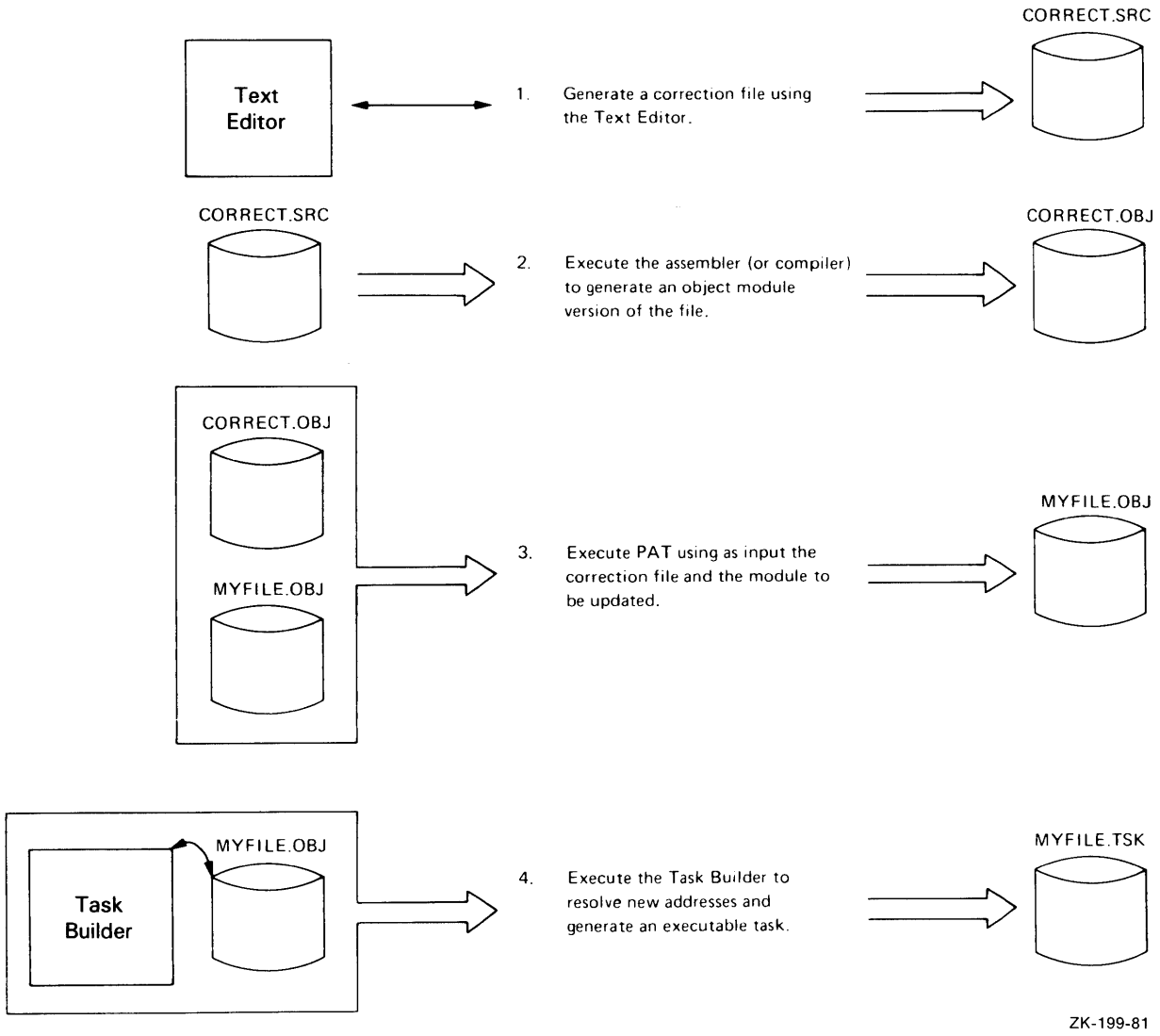
Specifies the file specification for the correction file. This file contains the updates to be applied to one module in the input file.

/CS[:n]

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 11.2.4 for information on how to use the /CS switch.) You can optionally specify an octal number, n, with the /CS switch. Then, after PAT calculates the checksum value, it compares that value with the number specified. If the values are not the same, PAT informs you with an error message. You must then rerun PAT with the correct checksum specified.

Using PAT to update a file involves several steps. First, you create the correction file by 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 TKB to resolve global symbols and to create an executable task. Figure 11–1 shows the steps involved in using PAT to generate an updated task file.

Figure 11–1: Processing Steps Required to Update a Module Using PAT



11.2 How PAT Applies Updates

This section describes the PAT input and correction files, gives information on how to create the correction file, and provides examples of how PAT applies the corrections to a module.

11.2.1 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.

11.2.2 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 11–1, 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.

Format

```
.TITLE inputname
.IDENT updatenum
inputline
inputline
.
.
.
```

Parameters

inputname

Specifies the name of the module to be corrected by the PAT update. You must specify the module that you are updating for inputname.

updatenum

Specifies any value acceptable to the MACRO–11 assembler .IDENT directive. Generally, this value reflects the updated version of the file to be processed by PAT (as shown in the example provided in Sections 11.2.3.1 and 11.2.3.2).

Note

The .IDENT assembler directive is a required part of the correction file. Failure to include an .IDENT assembler directive in the file produces unusable output.

inputline

Specifies 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 if the following conditions are met:

- 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.

11.2.3 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 TKB, along with a source-like representation of how the output file looks once PAT and TKB complete processing. Two techniques are described: one for overlaying lines in a module and the other for adding a subroutine to a module.

11.2.3.1 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, TKB generates a task image from the patched object modules.

The following file is the input file:

```
.TITLE  ABC
.IDENT  /01/
ABC::
MOV     A,C
CALL    XYZ
RETURN
.END
```

To add the ADD A,B instruction 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 TKB 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 ".=.+12" in the program counter field to determine where to begin overlaying instructions in the program. It overlays the RETURN instruction with the following patch code:

```
ADD     A,B
RETURN
```

11.2.3.2 Adding a Subroutine to a Module

The following 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, to execute that code, and then to return to inline processing.

The input file is as follows:

```
.TITLE  ABC
.IDENT  /01/
ABC::
MOV     A,B
CALL    XYZ
MOV     C,R0
RETURN
    .
    .
    .
.END
```

The correction file for this example is as follows:

```
.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 shown in the example in Section 11.2.3.1. TKB 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 instruction is included because this is a case where a 2-word instruction replaces a 3-word instruction and the NOP instruction is required to maintain alignment.) TKB 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.

11.2.4 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, as follows:

```
=MYFILE/CS,CORRECT.POB
```

The command directs PAT to calculate the checksum for the input file; MYFILE.PAT then responds with the following message:

```
PAT -- Input module checksum is checksum
```

PAT generates a similar message when you request the checksum for the correction file, as follows:

```
=MYFILE,CORRECT.POB/CS
```

After calculating the checksum for the correction file, PAT responds with the following message:

```
PAT -- Correction input file checksum is checksum
```

If you specify /CS:n to validate the size of a file, PAT calculates the checksum for the file and then compares that checksum with the value, n, you specified. If the two values do not match, PAT displays the following message to report the checksum error:

```
PAT -- Error in file filename checksum
```

You might specify the following change:

```
=MYFILE,CORRECT.POB/CS:432163
```

When PAT calculates the checksum for the correction file, the number is different. PAT then displays the following message:

```
PAT -- Error in file CORRECT.POB checksum
```

Checksum processing always results in an octal, nonzero value.

11.3 PAT Messages

PAT generates messages that state checksum values and messages that describe error conditions. For diagnostic error messages, PAT prefixes the messages with the following:

```
PAT -- *DIAG*- message
```

For fatal error messages, PAT uses the following prefix:

```
PAT -- *FATAL*- message
```

Fatal and diagnostic error messages describe a condition that caused PAT to terminate the processing of a command. When this occurs, PAT returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine

(MCR) prompt, PAT will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>PAT commandline
PAT -- message
>
```

If the command is entered in response to the PAT prompt, PAT will issue the error message and prompt you with the PAT prompt, as follows:

```
PAT>commandline
PAT -- message
PAT>
```

If an error occurs during the processing of an indirect command file, the command file is closed, the error message—followed by the command line in error—is issued to your terminal, and PAT returns to the highest level of command input.

11.3.1 Informational Messages

The following messages describe results of checksum processing:

PAT—Correction input file checksum is checksum

Explanation: When you specify the /CS switch in the correction file specification, PAT informs you of the file's checksum value. The value is given in octal.

User Action: No user action is required.

PAT—Input module checksum is checksum

Explanation: When you specify the /CS switch in the input file specification, PAT informs you of the file's checksum value. The value is given in octal.

User Action: No user action is required.

11.3.2 Error Messages

The following error messages are issued by PAT:

PAT—Command line error
commandline

Explanation: The system standard command line processor (.GCML) detected an error in the command line.

User Action: Use the correct information and reenter the command line.

PAT—Command syntax error
commandline

Explanation: The command line contained a syntax error.

User Action: Use the correct syntax and reenter the command line.

PAT—Correction input file missing commandline

Explanation: The mandatory correction file was not specified.

User Action: Specify the correction file and reenter the command line.

PAT—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.

PAT—Error in file filename checksum

Explanation: The checksum that PAT calculated for the named file.

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 with the correct version of the file specified.

PAT—Error positioning file filename

Explanation: PAT attempted to position the file beyond end-of-file (EOF).

User Action: Submit a Software Performance Report (SPR) along with the related console dialog and any other pertinent information.

PAT—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 TKB. The principal causes are as follows:

- 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.

PAT—Illegal device/volume specified devicename

Explanation: The device or volume name specification contained a syntax error.

User Action: Check the rules for specifying devices and volumes; then, use the correct syntax for the device or volume specification and reenter the command line.

PAT—Illegal directory specification
directoryname

Explanation: The directory specification contained a syntax error.

User Action: Check the rules for specifying a directory; then, use the correct syntax for the directory specification and reenter the command line.

PAT—Illegal error-severity code
errordata

Explanation: An error message call, containing an illegal parameter, has been generated.

User Action: If these messages persist, submit an SPR along with related console dialog and any other pertinent information.

PAT—Illegal file specification
filename

Explanation: The file specification contained a syntax error.

User Action: Use the correct syntax for the file specification and reenter the command line.

PAT—Illegal indirect file specification
commandline

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 directory; then, reenter the command line.

PAT—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, use the correct switch or switch value, and reenter the command line.

PAT—Incompatible reference to global symbol
symbolname

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.

PAT—Incompatible reference to program section sectionname

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 by changing the section type. Reassemble the file and resubmit it to PAT for processing.

PAT—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.

PAT—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.

PAT—Invalid file specified: filename

Explanation: You specified a file that contains one of the following errors:

- Nonexistent device
- Nonexistent directory—The directory in the file name specification does not exist on the specified device (or on the default device if no device was specified).

User Action: Specify the correct device or directory, and reenter the command line.

PAT—Maximum indirect file depth exceeded commandline

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.

PAT—Multiple output files specified
commandline

Explanation: PAT accepts only one output file specification.

User Action: Specify only one output file and reenter the command line.

PAT—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 Monitor Console Routine (MCR) or the DIGITAL Command Language (DCL) command INSTALL in the *RSX-11M-PLUS MCR Operations Manual* or *RSX-11M-PLUS Command Language Manual.*)

PAT—Required input file missing
commandline

Explanation: The mandatory input file was not specified in the command line.

User Action: Specify an input file and reenter the command line.

PAT—Symbol "symbol" is multiply defined

Explanation: You have tried to define a symbol that was previously defined.

User Action: Define a different symbol.

PAT—Too many input files specified
commandline

Explanation: Too many input files were specified in the command line. PAT accepts only the input and correction file specifications.

User Action: Specify the correct files and reenter the command line.

PAT—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, specify the correct file name, and reenter the command line.

PAT—Unable to locate module
modulename

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.

PAT—Unable to open file filename

Explanation: There is insufficient workspace in the internal File Storage Region (FSR) of the PAT utility.

User Action: If possible, PAT should be reinstalled with a larger increment or in a bigger partition. (See the description of the Monitor Console Routine (MCR) or the DIGITAL Command Language (DCL) command INSTALL in the *RSX-11M-PLUS MCR Operations Manual* or *RSX-11M-PLUS Command Language Manual.*)

Chapter 12

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 as follows:

- Copying files from one device to another
- Deleting files
- Renaming files
- Listing file directories
- Setting the default device and User Identification Code (UIC) for PIP operations
- Unlocking files
- Spooling files

12.1 PIP Command Line

You can invoke the PIP utility by using any of the methods described in Chapter 1. You invoke PIP file control functions by means of switches and subswitches.

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. (See Chapter 1 for more information on using indirect command files.)

The MCR command line for PIP differs for each function. Therefore, the command line formats are described in separate sections.

12.1.1 PIP Defaults for File Specification Elements

With the exception of the version number, PIP generally uses the last value encountered in the command line as the default. That is, PIP uses values you enter to set defaults and changes the default when you change the value. Exceptions to this are noted in the descriptions of each switch.

The format for entering file specifications is shown next.

Format

ddnn:[directory]filename.type;version[[/switch][...][/subswitch]]

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

Example

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

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

Causes T1.MAC;5 to set the defaults for the subsequent file specifications in the command line. Then, T2 is specified and overrides T1 as the default file name; 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.

Table 12–1 summarizes the rules PIP uses to set defaults.

Table 12–1: PIP Default File Specifications

Element	Default Value
ddnn	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 Default switch (/DF) (see Section 12.1.4.6). For subsequent file specifications, either you explicitly specify a new device or PIP assumes the device from the previous specification.
directory	For the first file specification, your current directory (that is, the directory under which you log in), the directory you specify with the SET command, or the default you specify with the /DF switch (see Section 12.1.4.6). For subsequent file specifications, either you explicitly specify a new directory or PIP assumes the directory from the previous specification. Only the asterisk (*) is valid as a wildcard (see Section 12.1.3).
filename	No default for the first file specification. For subsequent file specifications, it is the last file name that you explicitly specified. Asterisks (*) and/or percent signs (%) are valid as wildcards (see Section 12.1.3).
type	No default for the first file specification. For subsequent file specifications, it is the last file type that you explicitly specified. Asterisks (*) and/or percent signs (%) are valid as wildcards (see Section 12.1.3).

Table 12–1 (Cont.): PIP Default File Specifications

Element	Default Value
version	The default for input files is the most recent version number. The default for output files is the next higher version number, or it is 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 12.1.3.

12.1.2 Using PIP File Control 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 the /DD, /DF, /EX, /ID, and /TD switches, which are used by themselves.

Table 12–2 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 12.1.4.

Table 12–2: PIP Switches and Subswitches

Switch	Subswitch	Function
/AP		Appends file or files to the end of an existing file.
	/FO	Specifies the owner of a file.
/BS		Defines the block size for magnetic tape.
/CD		Allows the output file to take the creation date of the input file rather than the date of transfer.
/DD		Restricts file searches to files created during a specified period of time.
/DE		Deletes one or more files.
	/LD	Lists the deleted files.
/DF		Changes PIP's default device and/or directory.
/EN		Enters a synonym for a file in a directory file.

Table 12–2 (Cont.): PIP Switches and Subswitches

Switch	Subswitch	Function
	/NV	Forces the version number of a file to one greater than the latest version.
/EOF		Specifies the end-of-file (EOF) pointer for a file.
/EX		Excludes one file specification's worth of files during file searches.
/FI		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.
	/FU[:n]	Lists a directory file in full format.
	/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.
/ME		Concatenates two or more files into one file.
	/BL	Allocates a number (n) of contiguous blocks.
	/CO	Specifies that the output file or files be contiguous.
	/FO	Specifies the owner of 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 owner of a file.
	/GR	Sets the read/write/extend/delete protection at the group level.
	/OW	Sets the read/write/extend/delete protection at the owner level.
	/SY	Sets the read/write/extend/delete protection at the system level.
	/WO	Sets the read/write/extend/delete protection at the world level.
/PU		Deletes obsolete version or versions of a file.
	/LD	Lists the deleted files.
/RE		Renames a file.
	/NV	Forces the version number of a file to one greater than the latest version.

Table 12–2 (Cont.): PIP Switches and Subswitches

Switch	Subswitch	Function
/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		Deletes files selectively by prompting for your response before deleting them.
/SP		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 EOF.
/UF		Creates a User File Directory (UFD) entry on the volume to which a file is being transferred.
	/FO	Specifies the owner of a file.
/UN		Unlocks a file.
/UP		Updates (rewrites) an existing file.
	/FO	Specifies the owner of a file.

PIP accepts some switches with no file specification. However, when you use a switch in a command line, it must follow the file or directory specification. It cannot come before the device name, the directory, the file name, the file type, or the version of the file on which it is to operate.

You may specify a switch once for a list of file specifications as follows:

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

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

You specify switch arguments as octal (default), decimal, or alphabetic characters, depending on the switch. Refer to the specific sections that explain the individual PIP switches for information on these values.

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.

If you place the subswitch in the input file specification, it usually applies only to the file specification that immediately precedes it.

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.

Examples

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

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

Applies the Contiguous Output subswitch (/CO) to both TEST.TSK and SAMP.DAT. (The /CO subswitch is used with the Copy function. See Section 12.2.1.)

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

Applies the New Version subswitch (/NV) to the file ASDG.MAC. (The /NV subswitch is being used with the /RE switch.)

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).

```
PIP>FILE1.CMD/GR:R/WO,FILE2.MAC/GR:RW [RET]
```

or

```
PIP>FILE1.CMD/GR:R/WO,FILE2.MAC/GR:RW/PR [RET]
```

Provides the following file protection (both command lines are equivalent):

FILE1	SYSTEM	Unchanged
	MEMBER	Unchanged
	GROUP	Read access
	WORLD	No access
FILE2	SYSTEM	Unchanged
	MEMBER	Unchanged
	GROUP	Read/write access
	WORLD	Unchanged

For more information on altering the protection level of a file, see Section 12.1.4.16.

12.1.3 Wildcards

PIP allows you to specify wildcards in file specifications. The wildcard characters are the asterisk (*) and the percent sign (%). You can use both wildcards in place of explicit specifications for file names and types, and you can use 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 12.1.3.2.

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

12.1.3.1 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 will be ambiguous.

Note

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

Example

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

Shows an incomplete 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 or specifications.

For the Rename (/RE) and Enter (/EN) switches, the output specification can have wildcards (asterisks 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 directory form [*,*] or [*] is used to indicate that the output directory is to be the same as the input directory.

12.1.3.2 Wildcards in Input Specifications

PIP provides the following wildcard features for input file specifications:

.;*	Specifies all versions of all files.
.DAT;	Specifies all versions of all files of file type DAT.
.D;*	Specifies all versions of all files with file types beginning with D.
TEST.*;*	Specifies all versions of all types of files named TEST.
T*.*;*	Specifies all versions of all types of files with names beginning with T.
TEST.DAT;*	Specifies all versions of file TEST.DAT.
TEST.D%T;*	Specifies all versions of files named TEST with 3-character file types beginning with D and ending with T.
T%N.*;*	Specifies all versions of all file types of all 3-character file names beginning with T and ending with N.
.	Specifies the most recent version of all files.
*.DAT	Specifies the most recent version of all files of file type DAT.
*%.DAT	Specifies the most recent version of all files that have at least one character in their names and have the file type of DAT.
TEST.*	Specifies the most recent version of all file types for files named TEST.

PIP also provides the following wildcard directory features:

[*,*]	Specifies all group/member number combinations (1 to 377_8).
[n1,*]	Specifies all member numbers under group n1.
[*,n2]	Specifies all group numbers for member n2.
[*]	Specifies all named directories and group/member number combinations.

Note

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

12.1.4 PIP Switch Descriptions

The following sections contain detailed descriptions of the PIP switches used for file control functions. Table 12–2 lists these switches and subswitches for easy reference.

12.1.4.1 Append

The Append switch (/AP) opens an existing file and appends the input file or files to the end of it.

Use the Set File Ownership subswitch (/FO) to specify that the owning UIC of the output file is the same directory to which the input file belongs. If you do not specify the /FO subswitch, the owning UIC of the output file is unchanged, regardless of the directory to which the input files belong.

Format

outfile=infile[,...]/AP[/FO]

Parameters

outfile

Specfies the output file specification. Wildcard specifications are not allowed in the output file specification. The file type and the record attributes for the output file remain the same after the input file or files have been appended to it. The file name and file type for the output file must be specified explicitly.

infile

Specfies the input file specification. If the file name, file type, and version are not specified, then *.*;* is the default.

Note

If the output file is contiguous before the appending, it may not be contiguous afterward.

Example

```
PIP>DU1:FILE1.DAT;1=FILE2.DAT;1,FILE3.DAT;1,FILE4.DAT;1/AP [RET]
```

Opens FILE1.DAT;1 on DU1 and appends the contents of FILE2.DAT;1, FILE3.DAT;1 and FILE4.DAT;1 to it.

12.1.4.2 Block Size

The Block Size switch (/BS) 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.

Format

outfile/BS:n=infile

Parameters

outfile

Specifies the output file specification.

n

Specifies, in octal or decimal, the number of bytes in a block number.

infile

Specifies the input file specification.

Note

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>MS:BA.DOC/BS:2048.=AMBER.DOC [RET]
```

Increases the block size of the output file, BA.DOC, to 2048_{10} bytes per block.

12.1.4.3 Creation Date

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 /ME switch and/or with an output magnetic tape device.

Format

outfile/CD=infile[,...]

or

outfile=infile/CD[,...]

Parameters

outfile

Specifies the output file specification.

infile

Specifies the input file specification.

Example

```
PIP>/LI [RET]

DIRECTORY DB1:[200,200]
21-NOV-86 14:02

FILE.DAT;7       12.      6-OCT-86 16:13

PIP>TEST.DAT/CD=FILE.DAT/LI [RET]

DIRECTORY DB1:[200,200]
21-NOV-86 14:05

FILE.DAT;7       12.      6-OCT-86 16:13
TEST.DAT;1       12.      6-OCT-86 16:13
```

Creates a new file, TEST.DAT, from FILE.DAT and gives it the creation date of FILE.DAT rather than the transfer date.

12.1.4.4 Default Date

The Default Date switch (/DD) restricts file searches to files created during a specific period of time.

Format

/DD:startdate:enddate

Parameters

startdate

Specifies the beginning date of the time period in the form dd-mm-yy. It may be unlimited by using the wildcard character (*).

enddate

Specifies the ending date of the time period in the form dd-mm-yy. It may be unlimited by using the wildcard character (*).

Notes

1. Specifying the wildcard for both startdate and enddate negates the /DD switch. For example:

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

2. The date restrictions for the file searches are now disabled.

Examples

```
PIP>/DD:01-JUN-86:01-JUL-86&/LI RET
```

Lists all files created from 1 June 1986 to 1 July 1986.

```
PIP>/DD:*:1-JUN-86&/LI RET
```

Lists all files created on or before 1 June 1986.

```
PIP>/DD:1-JUN-86:*&/LI RET
```

Lists all files created on or after 1 June 1986.

12.1.4.5 Delete

The Delete switch (/DE) deletes files from a directory. Optionally, you can specify that the deleted files be listed on your terminal by including the List Deleted Files subswitch (/LD) in the command line.

Format

infile[,...]/DE[/LD]

Parameter

infile

Specifies the input file specification.

Notes

1. You must specify a version number or a wildcard in its place when using the /DE switch.
2. 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.
3. Wildcards in the file name or file type fields are illegal when a version of ;-1, ;0, or ; is specified.

4. You must issue the file specification because an unspecified file name, file type, and version does not default to *.*;*.
5. The input file specification can take all the usual forms, including wildcards (even in [directory]). The only special requirement is that the version field must always be specified.

Examples

```
PIP>TEST.DAT;-1/DE [RET]
```

Deletes the oldest version of file TEST.DAT.

```
PIP>TEST1.DAT;0,TEST2.DAT;/DE [RET]
```

Deletes the latest version of files TEST1.DAT and TEST2.DAT.

```
PIP>TEST.DAT;5/DE [RET]
```

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

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

Deletes versions 1 and 2 of file TEST.DAT from the current default directory on the default device.

```
PIP>*.OBJ;*,*.TMP;*/DE/LD [RET]
```

Deletes all versions of all files of the file type OBJ and TMP from the current default directory on the default device. It lists all deleted files of both file types.

```
PIP>*.OBJ;*/LD,*.TMP;*/DE [RET]
```

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.

12.1.4.6 Default

The Default switch (/DF) changes the default device and/or directory for the current PIP task.

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

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

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

Formats

ddnn:[directory]/DF

or

ddnn:/DF

or

[directory]/DF

or

/DF

Parameters

ddnn

Specifies the new default device to be applied to subsequent PIP command lines.

directory

Specifies the new default directory to be applied to subsequent PIP command lines.

Examples

```
PIP>[27,27]/DF
```

Sets the default directory to [27,27].

```
PIP>DU1:/DF RET
```

Sets the default device to DU1.

```
PIP>DU1:[27,27]/DF RET
```

Sets the default device to DU1 and the default directory to [27,27].

```
PIP>/DF RET
```

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

12.1.4.7 Enter

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.

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.)

Format

outfile=infile[,...]/EN[/NV]

Parameters

outfile

Specifies the file specification for 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 default. 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

Specifies the input file specification.

If you specify a device in either the input or output file specification, the 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 *.*;*.

Example

```
DIRECTORY DM1:[200,100]
20-MAR-87  16:00

RICK.DAT;1              12.       1-MAR-87 16:10

    Total of 12. blocks in 1. files
```

Shows the contents of the input file directory.

```
DIRECTORY DM1:[101,101]
20-MAR-87  16:10

JEN.OBJ;2               25.       18-MAY-87 11:19
LAU.OBJ;3               18.       18-MAY-87 12:32

    Total of 43. blocks in 2. files
```

Shows the contents of the output file directory.

```
PIP>[101,101]TWIG/EN=[200,200]RICK.DAT;1 [RET]
```

Produces the following output file directory:

```
DIRECTORY DM1:[101,101]
20-MAR-87  16:15

JEN.OBJ;2               25.       18-MAY-87 11:19
LAU.OBJ;3               18.       18-MAY-87 12:32
TWIG.DAT;1              12.       20-MAR-87 16:13

    Total of 55. blocks in 3. files
```

12.1.4.8 End-of-File

The End-of-File switch (/EOF) allows you to specify where the file's end-of-file (EOF) will be. This helps in certain situations (for example, system crashes) when a file contains useful information but its EOF pointer is wrong; thus, 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.

Format

infile/EOF[:block:byte][,...]

Parameters

infile

Specifies the input file specification.

The file specification must be issued. There is no default.

block

Specifies 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

Specifies that 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.

Notes

1. 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.
2. 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 [RET]
```

or

```
PIP>A.TMP/EOF:17:253,AA.TMP/EOF:23:0 [RET]
```

Indicates the file AA.TMP has 22 blocks allocated.

12.1.4.9 File Exclusion

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 contain characters and/or wildcards. Specifying the /EX switch by itself negates it.

Format

filespec/EX

Parameter

filespec

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

Examples

```
PIP>*.CMD;*/EX&/LI [RET]

DIRECTORY DB1:[301,7]
8-JUL-87 14:50
*.CMD;*EXCLUDED

EXECM.MAC;23        45.       23-FEB-86 14:23
RUN.TSK;46          5.        29-OCT-86 11:59
FRANK.OBJ;16        33.       02-MAY-86 13:58
DEBBIE.COR;2        5.0.      14-JAN-87 12:01
```

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

```
PIP>/EX [RET]
```

Negates the /EX switch.

12.1.4.10 File Identification

The File Identification switch (/FI) allows you to access an existing file by its file ID.

Format

outfile=/FI:filenum:seqnum

Parameters

filenum
Specifies the file number.

seqnum
Specifies the sequence number of the file.

Note

The file ID is assigned by the RSX-11M-PLUS operating system when the file is created. To find the file identification number of a file, use the Full List subswitch (/FU). The /FU subswitch displays the file identification and sequence numbers and other information describing the file.

Examples

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

Uses the /FI switch to create a directory entry for a file.

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

Copies a file by using the /FI switch.

```
PIP>/FI:1275:47/LI [RET]

DIRECTORY DR2: FILE ID 001275,000047,0
8-AUG-87 15:58

MCR.CMD       1.        29-NOV-86 13:22
DCL.CMD       1.        29-NOV-86 13:24
```

Lists entries in the directory file whose file identification is 1275,47, using the /FI switch.

12.1.4.11 Free

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.

Format

[ddnn:]/FR

Parameter

ddnn

Specifies the device volume.

Notes

1. If you do not specify a device, PIP defaults to SY0.
2. The format of the information from the /FR switch is as follows:

```
ddnn: 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.
```

3. 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.
4. The number of file headers will not exceed the number of files that can be created.

Example

```
PIP>DB7:/FR RET

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.
```

Indicates how much free space there is on the disk DB7.

12.1.4.12 Identify

The Identify switch (/ID) identifies which version of PIP is being used.

When you specify this switch, the version number is listed on the input terminal in the following format:

```
PIP version Mvvee [(ANSI)]
```

Parameters

vv

Specifies the version number.

ee

Specifies the edit number.

(ANSI)

Appears if PIP is linked to an American National Standards Institute File Control Services (ANSI FCS). Otherwise, it is blank.

Example

```
PIP>/ID [RET]

PIP -- PIP version M1340 (ANSI)
```

Identifies which version of PIP is being used.

12.1.4.13 List

The List switch (/LI) lists one or more files contained in a directory, along with their status information.

The /LI switch lists the following information:

1. The file name, file type and version in the format: 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—includes the number of blocks used/allocated and files printed

Three alternate mode subswitches (/BR, /FU, and /TB) allow you a choice of directory listing formats.

Format

[listfile=]infile[,...]/LI[/subswitch]

Parameters

listfile

Specifies the file specification to be listed.

If listfile is not specified, it defaults to TI.

infile

Specifies the input file specification.

The default for infile is *.*;*.

/subswitch

Specifies the alternate mode subswitch of the /LI switch. The list of possible subswitches follows:

/BR Specifies the brief form of directory listing. This subswitch lists only the file name, file type, and version.

/FU[:n] Specifies the full directory format.

Because the /FU subswitch 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 12.1.4.16.)

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 subswitch lists the following information:

1. The file name, file type, and version in the format: filename,filetype;version
2. The file identification number in the format: file number, file sequence number
3. Number of blocks used/allocated (decimal)
4. File code:

   (null) = noncontiguous
   C = contiguous
   L = locked

5. Creation date and time
6. Owner UIC and file protection in the format: [directory][system,owner, group,world]). These protection fields can contain the values R, W, E, or D:

   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—contains the number of blocks used, the number of blocks allocated, and the number of files

/TB This subswitch specifies the total number of blocks used for a directory and outputs the summary line as follows:

```
Total of nnnn./mmmm. blocks in xxxx. files
     nnnn = blocks used
     mmmm = blocks allocated
     xxxx = number of files
```

Examples

```
PIP>/TB [RET]
```

Produces the following directory listing:

```
Storage used/allocated for directory DM1:[SMITH]
15-APR-87 15:46

       Total of 145./150. blocks in 5. files
```

```
PIP>/BR [RET]
```

Produces the following directory listing:

```
DIRECTORY DM1:[SMITH]
15-APR-87  15:50

CKTST.MAC;6
CKTXT.MAC;7
CKTST.TSK;1
IOTST.MAC;4
IOTST.TSK;1
```

```
PIP>/LI [RET]
```

Produces the following directory listing:

```
DIRECTORY DM1:[SMITH]
15-APR-87  16:01

CKTST.MAC;6             3.        15-JAN-87 17:10
CKTST.MAC;7             0.      L 15-JAN-87 18:28
CKTST.TSK;1             69.    C  15-JAN-87 17:50
IOTST.MAC;4             4.        20-FEB-87  9:13
IOTST.TSK;1             69.    C  20-FEB-87 10:22

      Total of 145. blocks in 5. files
```

```
PIP>/FU [RET]
```

Produces the following directory listing:

```
DIRECTORY DM1:[SMITH]
15-APR-87  16:05

CKTST.MAC;6          (10,10)         3./3.          15-JAN-87 17:10
  [SMITH] [RWED,RWED,RWED,R]
CKTST.MAC;7          (13,14)         0./5.        L 15-JAN-87 18:28
  [SMITH] [RWED,RWED,RWED,R]
CKTST.TSK;1          (12,13)         69./69.    C  15-JAN-87 17:50
  [SMITH] [RWED,RWED,RWED,R]
IOTST.MAC;4          (11,11)         4./4.          20-FEB-87  9:13
  [SMITH] [RWED,RWED,RWED,R]
IOTST.TSK;1          (7,12)          69./69.    L  20-FEB-87 10:22
  [SMITH] [RWED,RWED,RWED,R]

     Total of 145./150. blocks in 5. files
```

```
PIP>/LI [RET]
```

Lists the directory of the current default device and directory (equivalent to TI:=*.*;*/LI).

```
PIP>LP:=[*,*]/FU:132. [RET]
```

Lists on the line printer, in full format (132-column listing), all of the directories on the current default device.

```
PIP>TI:=TEST.DAT/FU [RET]
```

Lists on TI the full directory listing for the latest version of TEST.DAT in the current default device and directory.

```
PIP>JUL13.DIR=[200,200]*.*/LI [RET]
```

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.

```
PIP>LP:=[11,*]*.CMD;*/LI [RET]
```

Lists on the line printer all versions of all files with the file type CMD in all directories in group 11.

```
PIP>LP:/BR=[11,11]*.CMD;*,*.DAT;*,*.MAC;1 [RET]
```

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.

12.1.4.14 Merge

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 12.2.1.

12.1.4.15 No Message

The No Message switch (/NM) suppresses the PIP error messages, "No such files(s)" and "File not locked," when you are manipulating files.

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

Format

infile[,...][/switch]/NM

Parameters

infile

Specifies the input file specification.

/switch

Specifies any combination of appropriate switches and subswitches, for example, the /LI, /DE, /PU, or /UN switch and any of its respective subswitches.

12.1.4.16 Protect

The Protect switch (/PR) allows you to set the protection status of a file. File protection is provided for the following four categories.

Category	Description
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_8).
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, or /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.)

The /PR subswitches set the 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:

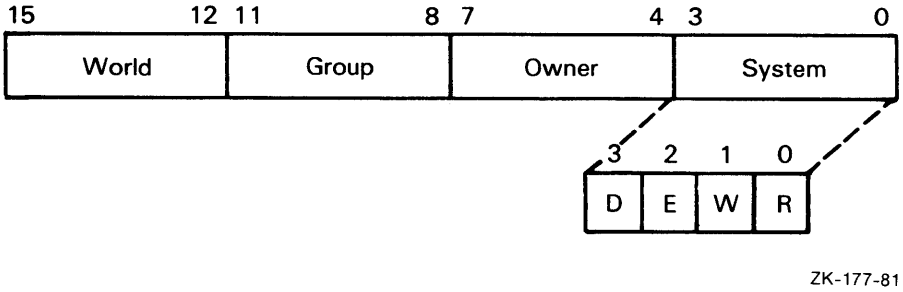
Subswitch	Description
/SY	System
/OW	Owner
/GR	Group
/WO	World

Protection can also be specified by an optional octal value on the /PR switch in the following 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-PLUS Mini-Reference* for the list of octal codes.) The format of the protection word is shown in Figure 12–1.

Figure 12–1: Format of Protection Word



The Set File Ownership subswitch (/FO) 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.)

Format

infile/PR[/SY[:RWED]][/OW[:RWED]][/GR[:RWED]][/WO[:RWED]][/FO]

Parameter

infile

Specifies the file specification whose protection is being changed.

The file specification must be issued. There is no default.

Examples

```
PIP>TEST.DAT;5/PR:3 RET
```

Denies write and read access to the system for file TEST.DAT;5.

Bits 0 and 1 are set. Bit set means no access permitted.

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

Causes all files to be owned by directory [200,200] without changing their protection if there are files in directory [200,200] that are owned by another UIC.

```
PIP>TEST.DAT;5/PR/OW:RWE/GR:RWE/WO RET
```

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.

```
PIP>[*,*]*.*;*/PR:0 RET
```

Sets the protection level of all files so that all categories are granted all access privileges.

```
PIP>DUO:[*,*]*.*;*/PR/FO RET
```

Causes all file owners to be the same UIC as the directory in which the files are entered.

12.1.4.17 Purge

The Purge switch (/PU[:n]) deletes a specified range of obsolete versions of a file. 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. 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 it 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.

Optionally, you can specify that the names of deleted files be listed on your terminal by using the List Deleted files subswitch (/LD).

Format

infile[,...]/PU[:n][/LD]

Parameter

infile

Specifies the file specification to be deleted.

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

Examples

```
DIRECTORY DM1:[200,201]
30-APR-87  10:15

GARY.MAC;1
GARY.MAC;2
GARY.MAC;3
GARY.MAC;4
GARY.MAC;5
RICK.TSK;4
RICK.TSK;5
RICK.TSK;7
```

Shows the contents of directory [200,201].

```
PIP>GARY.MAC/PU:3,RICK.TSK/PU:2 [RET]
```

Produces the following directory listing:

```
DIRECTORY DM1:[200,201]
30-APR-87  10:35

GARY.MAC;3
GARY.MAC;4
GARY.MAC;5
RICK.TSK;7
```

The three latest versions (3, 4, and 5) of file GARY.MAC are retained; versions 1 and 2 are deleted.

The two latest versions of file RICK.TSK are not deleted. Because file RICK.TSK;6 does not exist, only RICK.TSK;7 remains in the directory.

```
PIP>*.OBJ,*.MAC/PU:2/LD [RET]
```

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. All deleted files are listed.

```
PIP>*.OBJ/PU:2/LD,*.MAC [RET]
```

Deletes all but the two highest versions of all files with file types of OBJ and MAC. Lists all deleted files.

12.1.4.18 Rename

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.

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 the /NV subswitch is not necessary when both the input and output files are under the same directory file.) Note that you cannot rename a file that is copied from one device to another device.

Caution

Directories may still be accessed under the old directory name if the directory is in the Ancillary Control Processor's (ACP's) least-recently-used (LRU) list because the file ID is used to lookup directories. Renaming a directory pushes the directory out of the LRU list, thus ensuring that the directory can only be accessed by using the new name.

Format

outfile=infile[,...]/RE[/NV]

Parameters

outfile

Specifies 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 default (null). A directory, 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. Thus, the /RE switch can change one or more fields while preserving the others.

infile

Specifies the file specification to be renamed. The input file specifications are standard and allow wildcards in all fields, including directory.

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.

Examples

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

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

```
PIP>BACKUP.*;*=TEST.*;*/RE [RET]
```

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

```
PIP>*.*;1=*.*;*/RE [RET]
```

Renames all copies of all files to version 1.

```
PIP>[200,220]=[200,200]/RE [RET]
```

Renames all files from directory [200,200] to directory [200,220], preserving the file name, file type, and version of each file.

```
PIP>EXAMPLE.*;*=TEST.*;*/RE [RET]
```

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.

```
DIRECTORY DR02:[JONES]
22-MAR-87  10:30

SAVE.DAT;2
SAVE.DAT;3
SAVE.DAT;4
OUTPUT.DAT;1
OUTPUT.DAT;2
```

Shows the contents of directory [JONES].

```
PIP>SAVE.DAT/RE=OUTPUT.DAT;1 [RET]
```

Produces the folowing directory listing:

```
DIRECTORY DR02:[JONES]
22-MAR-87  10:35

SAVE.DAT;2
SAVE.DAT;3
SAVE.DAT;4
SAVE.DAT;1
OUTPUT.DAT;2
```

```
PIP>SAVE.DAT/RE/NV=OUTPUT.DAT;1 [RET]
```

Produces the folowing directory listing:

```
DIRECTORY DR02:[JONES]
22-MAR-87  10:30

SAVE.DAT;2
SAVE.DAT;3
SAVE.DAT;4
SAVE.DAT;5
OUTPUT.DAT;2
```

12.1.4.19 Remove

The Remove switch (/RM) removes an entry from a directory, but it does not delete the file associated with that entry. The /RM switch is particularly useful for deleting directory entries that, for whatever reason, point to nonexistent files. It is also used to delete synonyms generated by the /EN 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 14).

Caution

Directories may still be accessed after using the /RM switch if the directory is in the ACP's LRU list because the file ID is used to lookup directories. Renaming a file pushes the directory out of the LRU list, thus ensuring that the directory cannot be accessed by using the file name that was removed.

Format

infile[,...]/RM

Parameter

infile

Specifies the file specification for the directory file entry to be removed.

The file specification must be issued. There is no default.

Example

```
PIP>DU1:[10,10]RICKSFILE.DAT;1/RM [RET]
```

Removes the file entry RICKSFILE.DAT;1 from the directory [10,10] on DU1.

12.1.4.20 Rewind

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.

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 the /RW switch with the input specification to save search time. If you know a file is behind the tape's current position, The /RW switch 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.

Formats

outfile/RW=infile

or

outfile=infile/RW

Parameters

outfile

Specifies the output file specification.

infile

Specifies the input file specification.

Examples

```
PIP>MS:/RW=[200,200] RET
```

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

```
PIP>AMBER.DOC=MS:AB.DOC/RW RET
```

Rewinds the tape, searches the tape for the file AB.DOC, renames the file AMBER.DOC, and outputs the file.

12.1.4.21 Span Blocks

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 the /-SB switch, the records will not cross block boundaries.

Format

ddnn:outfile/SB=mmnn:infile

Parameters

ddnn
Specifies the output disk volume.

outfile
Specifies the disk output file specification.

mmnn
Specifies the input tape volume.

infile
Specifies the magnetic tape input file specification.

Example

```
PIP>DU1:FILES.DAT/-SB=MM0:FILES.DAT [RET]
```

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

12.1.4.22 Selective Delete

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 pressing the RETURN key, CTRL/Z, Y, N, G, or Q; each followed by pressing either the RETURN key or CTRL/Z. Table 12–3 describes the effect of each combination of letter and terminator.

Format

infile[,...]/SD

Parameter

infile
Specifies the input file specification.

The file specification must be issued. There is no default.

Table 12–3: Response Choices for the Selective Delete Switch

Letter	Terminator	Operation
Y	RET	Deletes file and continues.
Y	CTRL/Z	Deletes file and exits from PIP.
[N]	RET	Saves file and continues.
[N]	CTRL/Z	Saves file and exits from PIP.
Q	RET	Saves file and returns to command mode.
Q	CTRL/Z	Saves file and exits from PIP.
G	RET	Deletes current file and all remaining candidates, lists deleted files, and returns to PIP command mode.
G	CTRL/Z	Deletes current file and all remaining candidates, lists deleted files, and exits from PIP.

Examples

```
PIP>MYFILE.DAT;*/SD [RET]

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 then PIP goes to the next candidate, MYFILE.DAT;2. It deletes this file and all remaining versions of MYFILE.DAT. It lists the deleted files and then PIP prompts for the next command.

```
PIP>TEST.*;*/SD [RET]

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 and then PIP goes on to the next candidate, TEST.TXT;3. It saves this file and all remaining files with file name TEST and then exits from PIP.

12.1.4.23 Spool

The Spool switch (/SP) directs a file to a line printer for printing. This switch applies only if you have the Queue Manager installed. (For more information on the Queue Manager, see the *RSX-11M-PLUS Batch and Queue Operations Manual.*)

Format

infile[,...]/SP[:n]

Parameters

infile

Specifies the file specification to be spooled for printing.

The file specification must be issued. There is no default.

If the file is specified by its file ID, it will be printed. File IDs are discussed in Section 12.1.4.10.

n

Specifies 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 [RET]
```

Spools the files RICK1.LST;1, KATHY.LST;1, and the file whose file ID is 12:22 for asynchronous printing.

12.1.4.24 Shared Reading

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.

Format

outfile=infile/SR

Parameters

outfile

Specifies the output file specification.

infile

Specifies the input file specification.

Example

```
PIP>TI:=[210,20]FILES.DAT/SR [RET]
```

Enables you to read FILES.DAT even though another task may have already opened it for writing.

12.1.4.25 Today Default

The Today Default switch (/TD) restricts file searches to files created on the current day.

Format

/TD

Note

Specifying the wildcard for both start date and end date of the /DD switch (/DD:*:*) also negates the /TD switch.

Example

```
PIP>/TD&/LI [RET]

DIRECTORY DB2:[301,357]
20-JAN-87 11:02
DAY OF 20-JAN-87

TEST.DAT;1       1.      20-JAN-87 10:40
FILES.TXT;1      1.      20-JAN-87 10:41
INFO.DAT;1       1.      20-JAN-87 10:50

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

Lists the files created on the current date.

12.1.4.26 Truncate

The Truncate switch (/TR) allows you to truncate files back to their logical end-of-file (EOF) point. Note that RMS–11 files other than those that are fixed length, variable length, or sequenced cannot be truncated.

File Control Services (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 frees the unused blocks for use by other files.

Format

infile[,...]/TR

Parameter

infile

Specifies the input file specification.

The file specification must be issued. There is no default.

Examples

```
PIP>*.MAC/LI [RET]

DIRECTORY DR2:[301,7]
2-AUG-87 15:32

A.MAC;1        3.      20-SEP-86 14:02
B.MAC;1        2.      20-SEP-86 15:38
C.MAC;2        5.      28-SEP-86 09:54

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

Lists the files with type MAC.

```
PIP>*.MAC/TR [RET]
PIP>*.MAC/LI [RET]

DIRECTORY DR2:[301,7]
2-AUG-87 15:33

A.MAC;1        3.      20-SEP-86 14:02
B.MAC;1        2.      20-SEP-86 15:38
C.MAC;2        5.      28-SEP-86 09:54

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

Truncates, and then lists, the files with type MAC.

12.1.4.27 User File Directory

The User File Directory switch (/UF) creates a directory 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 directory in the output file specification if you want to create the directory or directories from which you are obtaining the files being transferred.

To use the /UF switch, you must have write access to the Master File Directory (MFD) 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 you can write to it:

1. Log in to the system under your UIC.
2. Reset your UIC to a privileged class by using the SET command: SET /UIC=[group,member]. Group and member specify a privileged class.

A typical use of the /UF switch is creation of a backup volume.

Format

outfile/UF[/FO]=infile[,...]

Parameters

outfile

Specifies the file specification for the output file.

infile

Specifies the file specification for the input file.

Example

```
PIP>DU6:[*,*]/UF/FO=SY:[104,20]*.MAC,*.OBJ RET
```

Writes all files with file types OBJ and MAC in directory [104,20] to a backup volume called DU6.

12.1.4.28 Unlock

The Unlock switch (/UN) unlocks (gives permission to open) a file that was locked because it was improperly closed. If a program using 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:

- Current block buffer being altered
- Record attributes that contain the EOF 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.

You must run PIP under the UIC of the file owner or under a system-level UIC.

Format

infile[,..]/UN

Parameter

infile

Specifies the file specification for the file to be unlocked.

The file specification must be given. There is no default.

Example

```
PIP>DU1:[100,100]RICK1.OBJ;3/UN RET
```

Unlocks the file RICK1.OBJ;3 in directory [100,100] on device DU1.

12.1.4.29 Update

The Update switch (/UP) is similar to the basic PIP copy function or the /ME 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 or files. Unlike the Supersede subswitch (/SU), the /UP switch does not delete the existing file before rewriting the data. Therefore, its file ID remains the same. Also, the number of blocks allocated to the output file can be the same or greater, but they can never be less than the number of blocks allocated to the existing file. However, as with the /SU subswitch, the file name, type, and version number remain the same.

Use the Set File Ownership subswitch (/FO) to transfer file ownership. The /FO subswitch 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 subswitch, the owning UIC of all new files is the UIC under which PIP is running, regardless of the UIC into which the file was entered.

Format

outfile=infile[,...]/UP[/FO]

Parameters

outfile

Specifies the file specification for the file to be rewritten.

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

Specifies the file specification for the file to be copied into the file that is being rewritten.

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

Example

```
PIP>DU1:SAMPLE.DAT;1=TEST1.DAT;1,TEST2.DAT;1,TEST3.DAT;1/UP [RET]
```

Opens SAMPLE.DAT;1 on DU1 and replaces the data currently in the file with the contents of files TEST1.DAT;1, TEST2.DAT;1, and TEST3.DAT;1.

12.2 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.

12.2.1 Copying and Merging 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 shown here.

Format

outfile=infile

Parameters

outfile

Specifies 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 the /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

Specifies 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 (/ME). The /ME switch creates a new file from two or more existing files. PIP assumes the /ME switch when you explicitly specify an output file, two or more input files, and no switches. Because the basic copy function and the /ME switch are logically related, the /ME switch is described here rather than with the other switches.

The general format of the PIP command line is shown here.

Format

outfile=infile[,...][[/ME][/subswitch]]

Parameters

outfile
Specifies the output file specification.

infile
Specifies the input file specification.

/ME
Specifies the Merge switch.

subswitch
Specifies any of the subswitches that you can enter as part of the basic command line or with the /ME switch. 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. The subswitches are as follows:

/BL:n — The Blocks Allocated 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 subswitch when you are copying a contiguous file and changing its size.

/CO — The Contiguous Output subswitch specifies that the output file be contiguous. When you are copying contiguous files from magnetic tape (for example, task images), specify both the /CO and /BL subswitches. You must specify the /BL subswitch 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 — The Noncontiguous Output subswitch specifies that the output file does not have to be contiguous.

If you do not specify the /BL subswitch, the /CO subswitch, or the /-CO subswitch, PIP defaults to the size and attributes of the input file.

/FO — The Set File Ownership subswitch specifies that the owner of the output file will be the same as the output directory. If you do not specify the /FO subswitch, 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.

When you specify the /FO subswitch, PIP must be running under a UIC that has write access to all output directories.

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

/SU — The Supersede 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 directory. The existing file is deleted and a new one is created with the data from the input file or files. 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) from the number of blocks allocated to the existing file.

Examples

```
PIP>DU1:SAMP.DAT=DU2:TEST.DAT [RET]
```

Copies the latest version of file TEST.DAT (in the current directory) from DU2 to DU1, and names it SAMP.DAT.

```
PIP>DU1:[*,*]=DU0:[11,*] RET
```

Copies all files from all members in group number 11 of DU0 to DU1. The files are copied to the same directory on DU1 that they were in on DU0. Note that the user must have write access to all group number 11 directories on DU1.

```
PIP>LP:=*.LST RET
```

Copies the latest version of all files with a type of LST in the current directory to the line printer. This command line is used if the Print Spooler is not installed on your system.

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.

```
PIP>*.LST/SP RET
```

Copies the latest version of all files with a type of LST in the current directory to the line printer. The Spool switch (/SP) is used if the Print Spooler is installed on your system.

```
PIP>DU1:SAMP.DAT=DU2:TEST.DAT;1,NEW.DAT;2/ME RET
```

Concatenates version 1 of file TEST.DAT and version 2 of file NEW.DAT from DU2, and generates file SAMP.DAT on DU1 using the current directory. Note the result would be the same if the /ME switch was not specified.

```
PIP>DU1:=DB2:TESTPROG.MAC,.OBJ RET
```

Copies the latest versions of TESTPROG.MAC and TESTPROG.OBJ from DB2 to DU1, using the current directory for both DB2 and DU1.

```
PIP>DK1:=DK0:*.DAT;* RET
```

Copies all versions of all of the files of file type DAT in the current directory from DU0 to DU1.

```
PIP>DT0:=[200,10]*.*;* RET
```

Copies all files under directory [200,10] from the default device (SY) to DT0, using the current directory.

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

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

```
PIP>DU0:[200,200]=DU1:[200,220]TEST.DAT RET
```

Creates a new file in the directory [200,200] on DU0, but the file is owned by User Identification Code (UIC) [1,1], which is the UIC under which PIP is running.

```
PIP>DU0:[200,200]=DU1:[200,220]TEST.DAT/FO RET
```

Creates a file owned by UIC [200,200], even though PIP is running under UIC [1,1].

```
PIP>DU1:[*,*]/FO=DP0:[13,10],[32,10],[34,10] RET
```

Copies all the files from the specified input directories to the corresponding directories on DU1. The file owners are the output directories.

```
PIP>DU1:[*,*]=DU0:[*,10]*.MAC/FO [RET]
```

Copies all the MAC files from all group numbers with member 10 to DU1, preserving the directory and setting the file owner for each file to that directory.

```
DIRECTORY DK1:[201,201]
20-MAR-87  11:05

RICK.DAT;1              12.       1-MAR-87 16:10

    Total of 12. blocks in 1. files
```

Shows the contents of the input directory.

```
DIRECTORY DK1:[100,100]
20-MAR-87  11:10

RICK.DAT;2              14.       3-MAR-87 17:05
RICK.DAT;3              20.      10-MAR-87 10:22

    Total of 34. blocks in 2. files
```

Shows the contents of the output directory.

```
PIP>DK1:[100,100]=DK2:[201,201]RICK.DAT;1 [RET]
```

Results in the following output directory:

```
DIRECTORY DK1:[100,100]
20-MAR-87  11:15

RICK.DAT;2              14.       3-MAR-87 10:22
RICK.DAT;4              20.      10-MAR-87 10:22
RICK.DAT;1              12.      20-MAR-87 11:12

    Total of 46. blocks in 3. files
```

```
PIP>DK1:[100,100]=DK2:[201,201]RICK.DAT;1/NV [RET]
```

Results in the following output directory:

```
DIRECTORY DK1:[100,100]
20-MAR-87  11:20

RICK.DAT;2              14.       3-MAR-87 10:22
RICK.DAT;4              20.      10-MAR-87 10:22
RICK.DAT;5              12.      20-MAR-87 11:17

    Total of 46. blocks in 3. files)
```

12.2.2 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 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 switches.

You can specify one or more commands in a line. When you use multiple commands, the ampersand (&) character separates each command.

Example

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

Performs the following steps:

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 subswitch 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.

12.3 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.>
```

Parameters

nn

Specifies one of the PIP error codes (see Section 12.3.1)

mm

Specifies one of the standard system, file primitive, or FCS codes listed in the *RSX-11M-PLUS and Micro/RSX I/O Operations Reference Manual*

qq

Specifies one of the directive error codes listed in the *RSX-11M-PLUS and Micro/RSX I/O Operations Reference Manual*

Error messages describe a condition that caused PIP to terminate the processing of a command. When this occurs, PIP returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, PIP will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>PIP commandline
PIP -- message
>
```

If the command is entered in response to the PIP prompt, PIP will issue an error message and prompt you with the PIP prompt, as follows:

```
PIP>commandline
PIP -- message
PIP>
```

The PIP error messages, their descriptions, and suggested user actions are as follows:

PIP—Allocation failure - no contiguous space

Explanation: Not enough contiguous space was available on the output volume for the file being copied.

User Action: Delete all files that are no longer required on the output volume and then reenter the command line. Also, use the Backup and Restore Utility (BRU) or the Disk Save and Compress Utility (DSC) to compress the files on your disk. BRU is described in Chapter 3 and DSC is described in Chapter 6.

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 3 and DSC is described in Chapter 6.

PIP—Bad use of wildcards/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 directory does not exist on the volume.

User Action: Specify the correct directory or the correct volume and reenter the command line.

PIP—Cannot find file(s)

Explanation: The file or files specified in the command line 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: Rename the file on the input volume and reenter the command line; 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 by using the File Structure Verification Utility (VFY) on the volume in question to determine if it is corrupted. VFY is described in Chapter 14.

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 or other privilege violation

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 User Identification Code (UIC) specified.

PIP—Explicit output file name required

Explanation: You did not specify an output filename.

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 directory are specified correctly, or request PIP under the correct UIC and reenter the command line.

PIP—Failed to find file(s)

Explanation: The file or files specified in the command line were not found in the designated directory.

User Action: Check the file specification and reenter the command line.

PIP—Failed to get time parameters

Explanation: An internal system failure occurred while PIP was trying to obtain the current date and time.

User Action: Reenter the command line. If the problem persists, submit a Software Performance Report (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 VFY against the volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 14.

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 removed a file from a directory, failed to enter it (using the /RE switch) into another directory, and failed to replace the original directory entry.

User Action: Run VFY and use the /LO switch to recover the file name. VFY is described in Chapter 14.

PIP—Failed to spool file for printing

Explanation: The Queue Manager is not installed, or the file was being spooled by file ID.

User Action: Install the Queue Manager or specify the file name, 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 VFY against the volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 14.

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 VFY against the volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 14.

PIP—File block has Forced Error Bit set

Explanation: A bad block replacement was needed during a read operation. The replacement block has a "forced error" bit set and may contain corrupt data that was read out of the bad block. The message occurs when the replacement block is being merged or appended to other files. The transfer of the file with the replacement block is then aborted. The records contained in the replacement block and all subsequent blocks will not be transferred. PIP will continue with the next element in the list, if one exists.

User Action: If the file is needed, copy the file with the replacement block to an output file and use the File Dump Utility (DMP) to examine the contents of the file or, if the problem is known, use the Task/File Patch Utility (ZAP) to correct the problem.

PIP—File is lost

Explanation: PIP removed a file from its directory, failed to delete it, and failed to restore the directory entry.

User Action: Run VFY and use the /LO switch to recover the file name. The VFY utility is described in Chapter 14.

PIP—File not locked

Explanation: The /UN switch was entered for a file that was not locked.

User Action: Specify the correct file and reenter the command line.

PIP—Get Command Line - bad @ file name

Explanation: An illegal indirect command file name was specified.

User Action: Specify the correct name for the indirect command file and reenter the command line.

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 an 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 your response 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, which is not allowed.

User Action: Rename the files or copy them into a different directory; then, reenter the command line.

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 with a Merge, Update, or Supersede switch had conflicting attributes, or the attributes of the input file or files specified with an Append switch 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 switch, the attributes of the output file will be those of the first input file. With an Append switch, 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; then, 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 DIRECTORY.

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 using the /INC switch with the MCR command INSTALL (see the *RSX-11M-PLUS MCR Operations Manual*) or by using the /EXTENSION qualifier with the DCL command INSTALL (see the *RSX-11M-PLUS Command Language Manual*).

PIP—No such file(s)

Explanation: The file or files specified in the command were not found in the designated directory.

User Action: Check the file specification and reenter the command line.

PIP—Only [*,*] is legal as destination UIC

Explanation: A UIC other than [*,*] was specified as the output file UIC for a copy operation.

User Action: Reenter the command line with [*,*] specified as the output UIC.

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 exists on the physical device (for example, device cycled down).
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

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; then, 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 the /NV subswitch to assign a new version number or use the /SU subswitch to supersede the output file.

PIP—Too many command switches - ambiguous

Explanation: Too many switches were specified or the switches conflict.

User Action: Specify the correct set of switches and reenter the command line.

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.

PIP—WARNING—File may contain corrupt data

Explanation: A bad block replacement was needed during a read operation. The replacement block may contain corrupt data that was read out of the bad block. This message occurs only when the input file is being copied to its own output file.

User Action: Delete the file if desired. If the file is needed, use the File Dump Utility (DMP) to examine the contents of the file or, if the problem is known, you can use the TASK/FILE Patch Utility (ZAP) to correct the problem.

12.3.1 PIP Error Codes

Table 12–4 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 12.3.

Table 12–4: 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

Table 12–4 (Cont.): PIP Error Codes

Error Code	Error Message
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 directory
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

Table 12–4 (Cont.): PIP Error Codes

Error Code	Error Message
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 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 perform the following tasks:

- Update (delete, replace, and add) lines in an existing file
- Create source files
- Run indirect command 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 is permanently applied to the output file.

You can control SLP processing with SLP control switches. These switches allow you to perform the following tasks:

- 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

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 by 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 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.

An example of the general form of command input is as follows:

```
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 13.5.2 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 command line 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 by specifying the checksum switch (/CS) with either the input or output file specification.

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 n is then reported in a message on your terminal. If you specify a value n for the checksum and it is not the same as the calculated checksum, you will get a diagnostic error message. (These messages are described in Section 13.6.3.)

13.1.3 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 SLP Output File

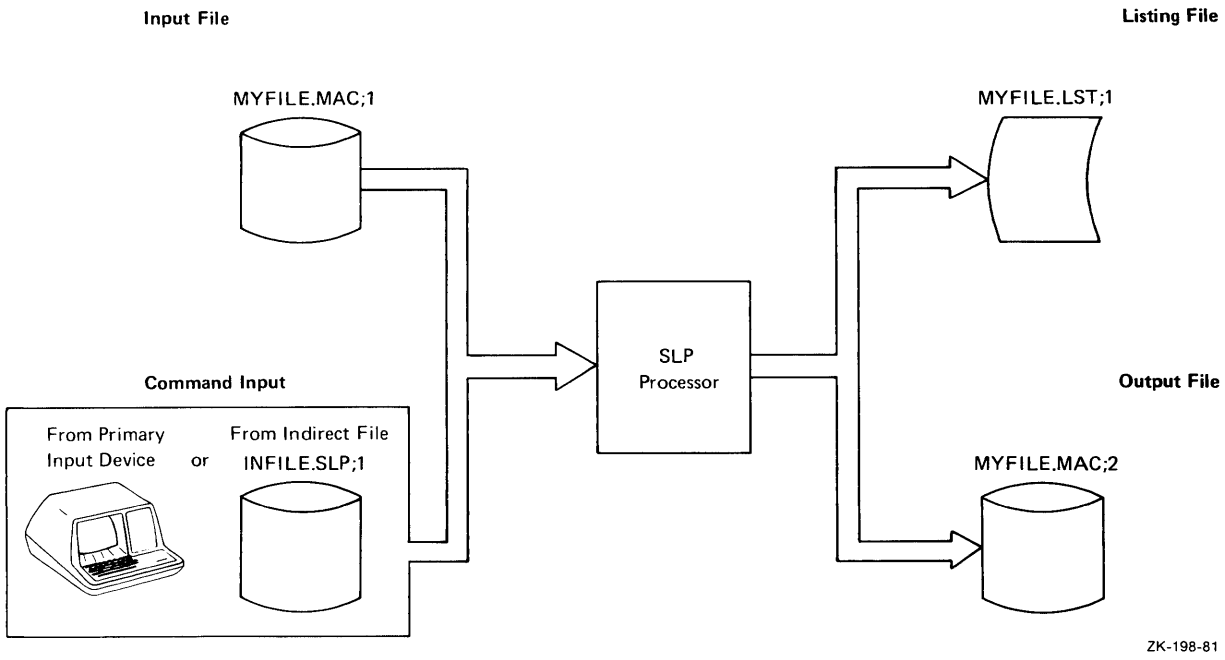
The SLP output file is the updated input file. All of the updates specified by command input are inserted in this file. The audit trail, if specified, is applied to lines changed by the update. The audit trail is included in the output file. The numbers generated by SLP for the listing file do not appear 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



The following file is the input file (MYFILE.MAC;1) before the previous SLP commands were processed:

```
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 file is the listing file (MYFILE.LST;1) produced by the previous SLP commands:

```
 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 by 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 (EOF) is reached.

13.3 SLP Edit Command Lines

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 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 the following command line:

```
>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.

All fields in the SLP edit command line are positional and commas (,) must be specified.

The SLP edit command line is shown next.

Format

```
-[locator1][,locator2][,/audittrail/][; comment]
inputline
.
.
.
```

Parameters

- (dash)
Identifies an SLP edit command line.

locator1
Specifies a line locator field 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 by using any of the locator formats, as follows:

```
-/string/[+n]
-/string...string/[+n]
-number[+n]
-.[+n]
```

The parameters for the locator formats are as follows:

string	Specifies a string of American Standard Code for Information Interchange (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	Specifies a decimal line number where the current line pointer is to be moved. The largest line number that can be specified is 9999.
(period)	Specifies the current line.
n	Specifies 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.

locator2

Specifies a line locator field that defines a range of lines (that is, the range beginning with locator1 and ending with locator2) to be deleted or replaced. This field can be specified by using any of the locator formats described previously.

audittrail

Specifies 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 audit trail.

comment

Specifies 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.

inputline

Specifies 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.

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.

The following file (BASE.MAC;1) is used as the input file for the example in Section 13.3.1:

```
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$
```

13.3.1 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 following 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. Then you enter the edit commands and input lines.

When you have entered all the corrections, enter the slash (/) to terminate the editing session. SLP processes the files and returns control to you with the following prompt:

```
SLP>
```

This returns SLP to command mode. You can then enter another input file and begin another editing session.

Example

```
SLP>BASE.MAC;2/AU:48./TR,BASE.LST=BASE.MAC;1 [RET]
-3 [RET]
TST     R1 [RET]
-4,4 [RET]
BEQ     10$ [RET]
-6,,/;JM010/ [RET]
CLR     R2 [RET]
/
SLP>
```

Instructs 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.

The following file is the listing file (BASE.LST;1) produced by the previous SLP command lines:

```
 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 in the previous 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 Entering SLP Commands by 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 it responds with the following 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 as follows:

```
@BASE.SLP [RET]
```

SLP processes the files just as if you entered the commands and input lines interactively, returning control to you with the following prompt:

```
SLP>
```

You can also specify the following command line:

```
SLP @BASE.SLP [RET]
```

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.

13.3.3 Entering and 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. You 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.

13.4 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.

13.4.1 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, and then enter the command line in the format shown next.

Format

```
,listfile=infile
/
```

Parameters

listfile

Specifies the name you assign to the listing file SLP produces.

infile

Specifies the name of the input file whose lines are to be numbered.

/

Terminates edit mode.

Example

```
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
```

Processes each line in the previous input file and generates the following numbered list file (listfile;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
```

13.4.2 Adding Lines to a File

There are three SLP edit command formats for adding lines to a file, as follows:

Format

```
-locator1
inputline
.
.
.
```

or

```
-locator1,,
inputline
.
.
.
```

or

```
locator1,,/audittrail/
inputline
.
.
.
```

Examples

```
SLP>MYFILE.MAC;2/AU:48.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1 [RET]
-3 [RET]
CMP     (R1)+,B [RET]
-4,4 [RET]
DEC     R2 [RET]
-6,,,/;JM010/ [RET]
INC     R3 [RET]
-9,,/;BJ008/ [RET]
BEQ     10$ [RET]
/
```

Shows how to add lines to a file.

```
SLP>MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1 [RET]
-/BEQ/ [RET]
CALL  WRITE [RET]
/
```

Indicates where new lines should be added or deleted by using text rather than line numbers.

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.

```
SLP>MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1 [RET]
-/#SIZ/+2 [RET]
CMP     (R1)+,B [RET]
/
```

Uses the same input file as the previous example to generate 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/.

13.4.3 Deleting Lines from a File

The SLP edit command format for deleting lines from a file is shown next.

Format

-[locator1],[locator2],[/audittrail/][; comment]

Parameters

locator1

Specifies, in any of the forms of the locator fields described in Section 13.3, the line where SLP is to begin deleting lines.

locator2

Specifies, in any of the forms of the locator fields described in Section 13.3, the last line to be deleted.

SLP deletes all lines from locator1 to locator2, inclusive.

Examples

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (R0)+,A
BNE     20$
INC     R1
```

Shows the contents of the input file.

```
SLP>MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1 [RET]
-/MOV...R1/,/NUMC/ [RET]
/
```

Deletes lines from the input file and generates the following listing file (MYFILE;1):

```
1.  MOV     #BUF1,R0
2.  CMPB     (R0)+,A
3.  BNE     20$
4.  INC     R1                          ;**-6
```

In the previous command lines, 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, which is 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.

```
SLP>MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1 [RET]
-/MOV #SIZ,R1/,. [RET]
/
```

Uses the same input file from the previous example to delete a single line by using the period locator, and it 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.

13.4.4 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.

Examples

```
-4,.+4
```

Instructs SLP to move the line pointer to line 4, and replaces line 4 and the next four lines with new input lines.

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
```

Shows the contents of the input file.

```
SLP>MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1 [RET]
-2,.+1 [RET]
CMP  (R1)+,B [RET]
INC  R2 [RET]
/
```

Deletes lines from the previous input file and replaces them with new lines. The edit command, -2,.+1, instructs SLP to delete lines 2 and 3 and insert two new lines. SLP processing then generates the following listing file (LISTING;1):

```
1.  MOV     #BUF1,RO
2.  CMP     (R1)+,B                 ;**NEW**
3.  INC     R2                      ;**NEW**
4.  TST     R2                      ;**-2
5.  BEQ     END
6.  CLR     R1
7.  MOV     R2,NUMC
```

13.5 Creating Source Files Using SLP

Using SLP to create source files is possible, but it is 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.

Format

outfile/-AU[/switch][,listfile][/switch]=[primary_input_device:[/switch]]

Parameters

outfile

Specifies the file specification for the output file.

The format for entering file specifications is as follows:

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

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.

switch

Specifies any optional SLP switches.

listfile

Specifies 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.

Example

```
SLP>MYFILE.MAC/-AU=TI: RET
```

Creates the file MYFILE.MAC on SY0 from your terminal without generating an audit trail.

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 press CTRL/Z.

13.5.1 SLP Switches

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.

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 Format	Function
[/AU:position:length] /-AU	Allows you to generate an audit trail or suppress the /-AU switch audit-trail generation and specify the beginning field (position) and length of the audit trail. The /AU switch is the default. 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. The /BF switch is the default.
/CM[:n]	Deletes audit trails and any trailing spaces or tabs, and truncates the text at a specified horizontal position, n. The value n given for the beginning position of the audit trail is the default value for this switch. See Section 13.5.2.4 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. The /-DB switch is the default.

Table 13–2 (Cont.): SLP Switches

Switch Format	Function
/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.
/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.
[/SP] /-SP	Spools the listing file to the printer. The /SP switch is the default. This switch applies only if you have the queue management system on your system.
/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–11 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.
/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.

13.5.2 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).

13.5.2.1 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 by using the /AU switch.

Format

/AU:position:length

Parameters

position

Specifies 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 13.5.2 for more information about the audit trail.)

length

Specifies the length of the audit trail. The default value for length is 8_{10} characters; no more than 14_{10} characters may be specified.

Example

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
```

Shows the contents of the input file.

```
SLP>MYFILE.MAC;2/AU:30.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1 [RET]
-2,.+1,/;CHANGE001/ [RET]
CMP     (R1)+,B [RET]
DEC     R2 [RET]
/
```

Specifies the beginning position and length of the audit trail when used with the previous input file.

The listing file MYFILE.LST;1 resulting from SLP processing is as follows:

```
1.  MOV     #BUF1,R0
2.  CMP     (R1)+,B         ;CHANGE001
3.  DEC     R2              ;CHANGE001
4.  TST     R2              ;**-2
5.  BEQ     END
```

13.5.2.2 Changing the Value of the Audit Trail

To change the value of the audit trail, specify the format shown next.

Format

-[locator1],[locator2],/;new value/

Parameters

locator1

Specifies, in any of the forms of the locator fields described in Section 13.3, the line where SLP is to begin deleting lines.

locator2

Specifies, in any of the forms of the locator fields described in Section 13.3, the last line to be deleted.

new value

Specifies the new audit trail value.

Example

```
MOV     #BUF1,RO
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (RO)+,A
BNE     20$
INC     R1
```

Shows the contents of the input file.

```
SLP>MYFILE.MAC;2/AU:48.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1 [RET]
-3 [RET]
CMP     (R1)+,B [RET]
-4,4 [RET]
DEC     R2 [RET]
-6,,/;JM010/ [RET]
INC     R3 [RET]
-9,,/;BJ008/ [RET]
BEQ     10$ [RET]
/
```

Inserts a line after line 3, deletes and replaces line 4, and inserts new lines after lines 6 and 9 with new audit trail values in the previous input file.

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
```

13.5.2.3 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.

Example

```
SLP>BAK.MAC;26/AU/-BF=BAK.MAC;25 [RET]
/
-2,2 [RET]
        .IDENT  /05.03/ [RET]
-23,23 [RET]
; VERSION 05.03 [RET]
-37,, [RET]
;  J. MATTHEWS      11-NOV-80 [RET]
; [RET]
;       JM011      CORRECT OUT-OF-BOUNDS CONDITION FOR INPUT-BUFFER [RET]
;                  SIZE [RET]
; [RET]
% [RET]
-106,106,/;JM011/ [RET]
   CMP    #132.,R3          ; IS INPUT-BUFFER SIZE IN RANGE? [RET]
   BLT    30$               ; IF LT, NO [RET]
   . [RET]
   . [RET]
   . [RET]
/
```

Shows the lines between the backslash and the percent sign are not affected by audit-trail processing. The lines following the percent sign are affected.

13.5.2.4 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 it then deletes any trailing spaces or tabs. The remaining text is copied to the output file.

Format

/CM[:n]

Parameter

n

Specifies a number designating the beginning character position of the audit trail (or other text) to be deleted.

Examples

```
SLP>SLPR11.MAC;12/CM:119.=SLPR11.MAC;11 [RET]
/
```

Truncates the input lines 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.

```
SLP>SLPR11.MAC;12=SLPR11.MAC;11/CM [RET]
/
```

Truncates input lines to the default position of the audit trail, column 80_{10}.

13.6 SLP Messages

SLP messages are divided into three groups: informational, fatal error, and diagnostic. Diagnostic and fatal error messages describe a condition that caused SLP to terminate the processing of a command. When this occurs, SLP returns to the level of command input. For example, when the command is entered in response to the SLP prompt, SLP will issue a fatal or diagnostic error message and prompt you with the SLP prompt, as follows:

```
SLP>commandline
SLP -- message
SLP>
```

Fatal error messages are issued to your terminal in the following format:

```
SLP -- *FATAL* - message
```

Diagnostic error messages are issued to your terminal in the following format:

```
SLP -- *DIAG* - message
```

If a diagnostic or fatal error occurs during the processing of an indirect command file, the command file is closed, the error message—followed by the command line in error—is issued to your terminal, and SLP returns to the level of command input.

The SLP error message text may be followed on the next line by the command line or command line segment that caused the message. For example:

```
SLP -- *FATAL*- Illegal switch or filespec
SHIRLEY.MAC;2/CF
```

SLP error message text may also be followed on the same line by the name of the file with which the error is associated. For example:

```
SLP -- *FATAL*- Open failure line listing file filename
```

13.6.1 SLP Informational Messages

This section lists the SLP informational messages. Following each message is an explanation of the error and any recommended user action.

SLP—Command file checksum is checksum

Explanation: By specifying the /CS 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 user action is required.

SLP—n lines truncated by audit trail

Explanation: Line truncation by the audit trail was detected.

User Action: This message is for your information 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 or truncations occurred. If necessary, modify the command file so it contains commands that do not cause truncation.

13.6.2 SLP Diagnostic Error Messages

This section lists the SLP diagnostic error messages. Following each message is an explanation of the error and the recommended user action to correct the error.

SLP—*DIAG*- I/O error command input file

or

SLP—*DIAG*- I/O error command output file

or

SLP—*DIAG*- I/O error correction input file filename

or

SLP—*DIAG*- I/O error line listing file filename

or

SLP—*DIAG*- 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; then, reenter the command line.

SLP—*DIAG*- Line number error
commandline

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. Correct the error and reenter the command line.

SLP—*DIAG*- 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; then, restart the editing session and enter the edit command line with the correct line number specified.

SLP—*DIAG*- Premature EOF command input file

Explanation: This is caused by not terminating SLP command input with a slash (/) or by inadvertently pressing CTRL/Z at the terminal, which sends an end-of-file (EOF) 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 pressed.

SLP—*DIAG*- Error in command file filename 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 will read as follows:

```
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.

13.6.3 SLP Fatal Error Messages

This section lists the SLP fatal error messages. Following each message is an explanation of the error and recommended user action to correct the error.

SLP—*FATAL*- Command syntax error
commandline

Explanation: The command line format did not conform to syntax rules. Open files were closed and SLP was reinitialized.

User Action: Specify the proper syntax and reenter the command line.

SLP—*FATAL*- Illegal device name
commandline

Explanation: The device specified was not a legal device. Open files were closed and SLP was reinitialized.

User Action: Specify a legal device and reenter the command line.

SLP—*FATAL*- Illegal directory
commandline

Explanation: The directory was not legally specified. Open files were closed and SLP was reinitialized.

User Action: Specify a legal directory and reenter the command line.

SLP—*FATAL*- Illegal error/severity code p1 p2 p3

Explanation: This error message indicates an error in the SLP program.

User Action: Reenter the command line. If the error persists, submit a Software Performance Report (SPR) with the console dialog and any other related information, such as programs or listings.

SLP—*FATAL*- Illegal file name
commandline

Explanation: A file specification was greater than 19 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: Specify a legal file name and reenter the command line.

SLP—*FATAL*- Illegal Get Command Line error code

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 an SPR with the console dialog and any other related information.

SLP—*FATAL*- Illegal switch or filespec
commandline

Explanation: This error has occurred for one of the following reasons:

- The switch was not a legal SLP switch.
- A legal switch was used in an illegal manner.
- A file specification could not be parsed.

User Action: Specify the legal switch or file specification and reenter the command line.

SLP—*FATAL*- Indirect command syntax error
commandline

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: Specify the proper syntax and reenter the command line.

SLP—*FATAL*- Indirect file depth exceeded
commandline

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*- Indirect file open failure
commandline

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; then, reenter the command line.

Chapter 14

File Structure Verification Utility (VFY)

The File Structure Verification Utility (VFY) for Files–11 volumes provides the ability to perform the following tasks:

- Check the readability and validity of a file-structured volume (default function).
- Print the number of available blocks on a file-structured volume (the Free switch (/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 (the Lost switch (/LO)). (See the *RSX–11M–PLUS and Micro/RSX I/O Operations Reference Manual* for a description of the index file.)
- Validate directories against the files they list (the Directory Validation switch (/DV)).
- List all files in the index file, showing the file ID, file name, and owner (the List switch (/LI)).
- Mark as "used" all the blocks that appear to be available but are actually allocated to a file (the Update switch (/UP)).
- Rebuild the storage allocation bit map so that it properly reflects the information in the index file (the Rebuild switch (/RE)).
- Restore files that are marked for deletion (the Delete switch (/DE)).
- Delete bad file headers (the Header Delete switch (/HD)).
- Perform a read check on every allocated block on a file-structured volume (the Read Check switch (/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.

Caution

VFY must not be aborted while an /UP, /RE, /DE, or the /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.

14.1 VFY Command Line

The command line for VFY is shown next.

Format

listfile,scratchdev=indev/switch

Parameter

listfile

Specifies the output file specification as follows:

```
ddnn:[directory]filename.type;version
```

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

If you do not specify a device, the default for the output listing device is the issuing terminal (TI). The default for [directory] is the default directory to which your terminal is set. 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:

```
ddnn:
```

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.

indev

Specifies the volume to be verified in the same format as scratchdev. If you do not specify the volume, the default is SY0.

switch

Specifies the function to be performed by VFY.

The switches and their functions are summarized in Table 14–1. Detailed descriptions of the switches are given in Section 14.1.1.

Table 14–1: VFY Switches and Functions

Switch	Format	Description
Delete	/DE	Resets the marked-for-deletion 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 All 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 that 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 are actually allocated to a file.

14.1.1 VFY Switch Descriptions

VFY functions are specified with switches appended to the VFY command line. The switches and their functions are summarized in Table 14–1.

14.1.1.1 Delete

The Delete switch (/DE) resets the marked-for-deletion indicators in the file header area of files that were marked for deletion but never deleted.

VFY must be running under a system User Identification Code (UIC) and the volume must be mounted and unlocked.

14.1.1.2 Directory Validation

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.

Example

```
>VFY DX:/DV [RET]

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
```

Shows that invalid 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 by using the Peripheral Interchange Program (PIP) switch Remove (/RM).

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 switch /RM.

Reserved field was nonzero

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 the PIP switch /RM and then reenter it with the PIP switch Enter (/EN).

14.1.1.3 Free

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.
```

14.1.1.4 Header Delete

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 [RET]
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
```

14.1.1.5 Identify

The Identify switch (/ID) identifies the current version of VFY running on your system. The /ID switch can be specified alone on a command line at any time.

14.1.1.6 List

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.

Example

```
VFY>DK:/LI [RET]
Listing of index on DKO:

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]
```

Lists the index file for DK.

14.1.1.7 Lost

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 file type 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 directory [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]
```

14.1.1.8 Read Check

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 one 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 or blocks cannot be read.

When an error is detected, a file identification line is displayed in the following format:

```
File ID nn,nn filename.type;version.  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 or blocks in error. The second error message line or lines will be in the following form:

```
Error at VBN n1,n2 LBN n1,n2 - error code - n
```

If an "Error starting at" message is displayed without one or more "Error at" messages, a multiblock read operation on the selected device has failed, but the data blocks appear to be individually readable.

If the Virtual Block Number (VBN) of the unreadable block listed in the "Error at" message 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 *RSX-11M-PLUS and Micro/RSX I/O Operations Reference Manual*.

14.1.1.9 Rebuild

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 /RE switch can take effect.

You must run VFY under a system UIC and unlock the volume. The scratch file should be on another volume.

14.1.1.10 Update

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 unlock the volume. The scratch file should be on another volume.

14.2 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:

- If the /UP or /RE switch is used, VFY requires write access to the storage allocation map ([0,0]BITMAP.SYS).
- If the /DE or /HD switch is specified, VFY requires write access to the index file ([0,0]INDEXF.SYS).
- If the /LO switch is specified and lost files are found, VFY requires write access to the User File Directory (UFD) [1,3], which is the directory containing "lost" files.

If write access to the volume index or bit map files is required for the desired operation, the user must mount the unlocked volume by using the Monitor Console Routine (MCR) or Digital Command Language (DCL) command MOUNT.

VFY may be run under any User Identification Code (UIC) if only read access is required. If write access is required, VFY must run under a system UIC.

14.3 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 bit map.

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.

Format

[listfile,scratchdev=]indev

Example

```
>VFY DR0: RET

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.
```

Checks the validity and readability of of DR0.

14.4 Files Marked for Deletion

If a file has been marked for deletion 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 bit map.

14.4.1 Restoring a File Marked for Deletion

To restore a file marked for deletion, mount the disk volume by using the MCR or DCL command MOUNT and the /UNL switch.

Example

```
>MOU DKO:/UNL RET
```

Shows the MOUNT command and the /UNL switch. After entering the command line, run VFY and specify the /DE switch to reset the marked-for-deletion 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 by using the PIP Full Format subswitch (/FU).

14.4.2 Deleting a File Marked for Deletion

Files that are marked for deletion 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 14.5). The following example shows how the file ID is used with PIP to delete a file.

Example

```
>PIP /FI:12:20/DE RET
```

Deletes the file with file ID 12,20 from the system device. PIP issues the following error message:

```
PIP -- Failed to mark file for delete-no such file
```

Since the file system denies the existence of files already marked for deletion, the file is deleted.

14.4.3 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 14.1.1.4 for a description of the /HD switch.

14.4.4 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 that 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.

14.4.5 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 bit map but are actually allocated to a file. To reallocate these blocks in the storage allocation bit map, run the validity check with the /UP switch. This allocates all blocks that should have been marked as allocated. See Section 14.1.1.10 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.

14.4.6 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 VFY Validity Check (Section 14.3). 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 bit map.

If there are no errors, these two figures should agree. If the index file indicates that more blocks are free than the storage allocation bit map indicates, 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 VFY Validity Check and specifying the /RE switch. See Section 14.1.1.9 for a description of the /RE switch.

14.5 VFY Error Messages

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.

Format

File ID nn,mm filename.type;version owner [uic]

Parameters

nn,mm

Represents the unique file identification number assigned to the file by the system at file-creation time.

filename
Represents the file name.

type
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 messages.

The last error message for the file is followed by a summary line for that file.

Format

Summary: mult=mb, free=fb, bad=n

Parameters

mb
Specifies the number of multiple block allocations.

fb
Specifies the number of blocks marked free that should have been allocated.

n
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, press CTRL/O. This terminates the listing of error messages for that file; that is, all messages but the summary line.

Error messages describe a condition that may cause VFY to terminate the processing of a command. If this occurs, VFY returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, VFY will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>VFY commandline
VFY -- message
>
```

If the command is entered in response to the VFY prompt, VFY will issue an error message and prompt you with the VFY prompt, as follows:

```
VFY>commandline
VFY -- message
VFY>
```

The VFY error messages, their explanations, and suggested user actions are as follows:

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.

User Action: Specify a legal block number and reenter the command line.

Bad file header

Explanation: VFY checks on the validity of the file header indicate that the header has been corrupted.

User Action: No user action is requred.

Block is marked free n,m

Explanation: The specified LBN is allocated to the indicated file but is not marked as allocated in the storage allocation map (see Section 14.4.5).

User Action: No user action is required.

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

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; then, 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.

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 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; then, reenter the command line.

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; then, 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 bit map by file ID 2.2

Explanation: The Master File Directory (MFD) has been corrupted.

User Action: Copy the disk by using the BRU utility (see Chapter 3).

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 as it 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; then, reenter the command line.

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 14.4).

User Action: No user action is required.

Header map error

Explanation: VFY detected an error in the header map area that also indicates a corrupted file header.

User Action: No user action is required.

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—Illegal response - try again

Explanation: A response was entered that was not legal.

User Action: Enter a legal response.

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

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; then, reenter the command line.

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.

User Action: No user action is required.

Multiple allocation n,m

Explanation: The specified (double-precision) logical block number (LBN) 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 14.4.3).

User Action: No user action is required.

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

Explanation: One of the following conditions may exist:

- VFY is not running under a system UIC as it 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; then, 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 (using the Free switch (/FR)) can examine the block.

User Action: Copy the disk by using BRU or DSC (see Chapter 3 or 6 respectively).

VFY—They are still lost, could not find directory

Explanation: Directory [1,3] did not exist on the volume. Directory [1,3] is the "lost files" directory. VFY enters all files found by the /LO switch into this directory.

User Action: Use the MCR command UFD or the DCL command SET DEF to enter directory [1,3] on the volume.

change. For information on TKB maps, see the *RSX-11M-PLUS and Micro/RSX Task Builder Manual*. For information on MACRO-11 listings, see the *PDP-11 MACRO-11 Language Reference Manual*.

15.1 ZAP Command Line

You invoke ZAP by 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 15.1.2).

The ZAP command line format is shown next.

Format

ddnn:[directory]filename.type;version/switch

Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

The ZAP switches are summarized in Table 15–1. The /LI switch is described in detail in Section 15.1.1. The /AB and /RO switches are described in detail in Section 15.2.

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 (CLI).

Table 15–1: ZAP Switches

Switch	Format	Function
Absolute Mode	/AB	Specifies absolute mode.
List	/LI	Displays the overlay segment table for an overlaid task image file.
Read-Only	/RO	Specifies read-only mode.

15.1.1 List Switch

When using ZAP in task image mode (but not absolute), you can also specify the List switch (/LI). The /LI switch directs ZAP to display the overlay segment table for the task image file on disk that you are working with. The overlay segment 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 for each type of task image file are discussed later in this chapter.) You use this table and TKB map to locate the task segments being changed.

The /LI switch displays the overlay segment boundaries in the format shown next.

Format

block:lower-upper [name] [type]

Parameters

block:

Specifies the starting block in octal.

lower

Specifies the lower address boundary in octal.

upper

Specifies the upper address boundary in octal.

name

Specifies the segment name that appears for instruction and data 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.

type

Specifies the description of the segment-type string that appears next to the segment name in the segment table.

The following sections describe the /LI 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 (TKB) map.

See the *RSX-11M-PLUS and Micro/RSX Task Builder Manual* 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 TKB does. TKB builds the library with each segment beginning on an even block boundary, but it then compresses the segments in the task file itself. The TKB 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 TKB 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 *RSX-11M-PLUS and Micro/RSX Task Builder Manual* for more information on resident libraries.

15.1.1.4 The /LI Switch and Instruction and Data Space Tasks

For instruction and data space (I- and D-space) tasks, the /LI switch lists the starting block number and the address boundaries of each segment. The I- and 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 TKB map.

15.1.2 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_8). The V command (explained in Section 15.6.5) is used to verify that 70 (106_8) 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, type the following:

```
>ZAP @CHANGE RET
```

This command invokes ZAP, which then executes the commands in the file.

The first command used opens the task image file (LPP.TSK) in absolute mode (using the /AB switch). 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 your CLI.

15.2 ZAP Operating Modes

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 — This 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 TKB map for the file. See Section 15.3.2.1 for more information on using task image mode.

Absolute mode — ZAP uses the addresses you enter in ZAP command lines as absolute byte addresses within the file. Absolute mode is specified with the /AB switch. You must use absolute mode for any files that are not task images. See Section 15.3.2.2 for more information on using absolute mode.

Read/write mode — This is the default access mode for ZAP. In this mode, ZAP opens a file for reading and/or modification.

Read-only mode — 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. Read-only mode is specified with the /RO switch.

15.3 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, it explains how to use the addressing modes.

15.3.1 Relocation Biases

When MACRO–11 generates a relocatable object module, the base address of each program section of the module is 000000. In the assembly listing, all locations in the program section are shown relative to this base address.

TKB 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 to 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.

15.3.2 ZAP Addressing Modes

As explained in Section 15.2, 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 example shows excerpts from a MACRO–11 listing of the module MYFILE and a TKB map. These excerpts and the accompanying text show how to use ZAP in task image mode.

Example

```
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)
```

Shows the assembled instructions from a MACRO–11 source 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.
```

Shows the lines from a TKB map.

Using information from the TKB map, you can address locations in the task image file as they appear in the MACRO–11 listing.

You can determine, from the TKB map, 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 as follows:

```
002:121026/ 032767
```

Section 15.4 describes the ZAP command line formats.

15.3.2.1 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 TKB map. Unlike absolute mode, task image mode is useful for working with locations in an overlaid file because TKB and ZAP perform the calculations necessary to relate the file's disk structure to its run-time memory structure.

15.3.2.2 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.

15.4 ZAP Commands 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 it also depends 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 and close location
- General-purpose
- RETURN key

The following sections describe each type of command.

15.4.1.1 Open and Close Location Commands

Open/close location commands are nonalphanumeric American Standard Code for Information Interchange (ASCII) characters that direct ZAP to perform a sequence of functions. Open/close commands specify the following 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 Command

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 to 7 (0R to 7R)	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.
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.
Format Register (F)	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.
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.

Example

```
_$C/ RET
$C/ 000000
-
```

Directs ZAP to display the contents of the Constant Register. The slash (/) is an open command (see Section 15.5).

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 15–2 describes the operators.

You use the operators in expressions in command lines. For example, rather than manually adding all the displacements listed in the TKB map, you can specify a location by 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 15–2: ZAP Arithmetic Operators

Operator	Designation	Function
+	Plus sign	Adds a value to another value. Used in an expression that ZAP then evaluates as a command line element.
-	Minus sign	Subtracts a value from another value. Used in an expression that ZAP then evaluates as a command line element.
*	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 (,) (see Table 15–3); the percent sign (%) is an open command and is described in Table 15–4.

Example

```
_0,40/ RET
002:0,000040/  000001
_1*33 RET
_/ RET
002:0,000040/  000103
_% RET
002:0,000040%  A$
```

Opens locations, displays contents, and converts locations.

The first command opens the locations that are 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 15-3.

Table 15-3: ZAP Command Line Element Separators

Separator	Description	Function
,	Comma	Separates a Relocation Register specification from another command line element.
;	Semicolon	Separates an address from an internal register specification. Used in expressions that set values for Relocation Registers.
:	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 four formats for specifying locations in a command line. Each provides a means of indexing into a file. The formats are as follows:

- Current location symbol
- byteoffset
- blocknum:byteoffset
- n,byteoffset

15.4.5.1 Current Location Symbol Format

In command line expressions that ZAP evaluates as addresses, a period (.) represents the last open location.

15.4.5.2 Byte Offset Format

The byte offset format is shown next.

Format

byteoffset

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.

Example

```
_664/ RET
```

Opens absolute location 664 and displays its contents in octal format.

15.4.5.3 Block Number:Byte Offset Format

The block number format allows you to specify a byte offset from a specific block in the file.

Format

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 TKB map. The map gives information on the overlay segments in a task image file. See Section 15.3 for more information.

15.4.5.4 Relocation Register, Byte Offset Format

The Relocation Register, byte offset format allows you to load a Relocation Register with the address of a location. The address is then used as a relocation bias. You can then address byte offsets from the address loaded in the Relocation Register.

The format for addressing locations in a task image file is shown next.

Format

n,byteoffset

Parameter

n

Specifies the number of the Relocation Register.

Example

```
_2:001254;3R RET
_3,64/ RET
002:3,000064/ 037334
```

Loads an addess and opens a location.

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 15–4 summarizes the open and close commands.

Table 15–4: ZAP Open and Close Commands

Character	Description	Function
/	Slash	Opens a location, displays its contents in octal, and stores the contents of the location in the Quantity Register (Q). If the location is odd, it is opened as a byte.
"	Quotation marks	Opens a location, displays the contents of the location as two ASCII characters, and stores the contents of the location in the Quantity Register (Q).
%	Percent sign	Opens a location, displays the contents of the location in Radix–50 format, and stores the contents of the location in the Quantity Register (Q).
\	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.
^	Circumflex Up arrow	Closes the currently open location as modified and opens the preceding location.
_	Underscore 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 marks ("), 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.

Example

```
_/ RET
002:120000/ 000000
_44444 RET
_/ RET
002:120000/ 44444
```

Shows how to open a location, change the value of the location, and close the location.

The first command, the slash (/), displays, in octal format, the contents (000000) of the current location. The contents are changed by entering the value 44444. The location is then closed, as modified, by pressing the RETURN key. The slash and the RETURN key display the new contents of the location (see the 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 the following 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 then to open that location.

Example

```
_2:120100/ RET
002:120100/ 000000
_ RET
002:120102/ 000111
_ RET
002:120104/ 000222
_ RET
002:120106/ 000333
_^ RET
002:120104/ 000222
```

Shows how the circumflex or up-arrow command works.

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 then to open the new location.

Example

```
_2:120100/ RET
002:120100/ 000000
_ RET
002:120102/ 111111
_ RET
002:120104/ 222222
_ RET
002:120106/ 000022
__ RET
002:120132/ 123456
```

Shows how the underscore or back-arrow command works.

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 then to open that location.

Example

```
_2:120100/ [RET]
002:120100/ 000000
_ [RET]
002:120102/ 111111
_ [RET]
002:120104/ 120114
_@ [RET]
002:120114/ 114114
```

Shows how the at sign command works.

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.

Example

```
_2:120100/ [RET]
002:120100/ 005000
_ [RET]
002:120102/ 005301
_ [RET]
002:120104/ 001020
_> [RET]
002:120146/ 052712
```

Shows how the right angle bracket command works.

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.

Example

```
_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
```

Shows how the left angle bracket command works.

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. The commands are entered in response to the ZAP underscore prompt (_).

Table 15–5 describes the commands.

Table 15–5: ZAP General-Purpose Commands

Command	Function
X	Exits from ZAP; returns control to your 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.

15.6.1 The X Command

Use the X command to exit from ZAP and return control to your CLI.

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 the offset value in the Quantity Register (Q).

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 following format:

```
=reg,offset
```

Example

```
_2:1172;0R [RET]
_2:1232;1R [RET]
_2:1202/ [RET]
002:000020/ 000111
_K [RET]
=0,000010
_0,100;1K [RET]
=1,000040
```

Shows how to use the K command.

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.

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.

Example

```
_0,4534/ [RET]
0,4534/ 1234
_45660 [RET]
_000030> 000014
_4534;45660 [RET]
_000030> 000014
```

Shows how to use the O command.

The first number (000030) specifies the jump displacement and the second number (000014) specifies 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.

Format

expression=

Example

```
_2:30/ [RET]
002:000030/ 000000
_.+177756= [RET]
000006
```

Shows how to use the equal sign command.

The first command displays in octal format the content of location 000030, which is 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). Note that 177777 equals -1.

15.6.5 The V Command

Use the V command to verify that a location contains a specified value.

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.

Format

contentsV

Example

```
0,1200/ [RET]
6V [RET]
10 [RET]
```

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.

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 15.3.1 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.

Format

_contents;nR

Parameter

n

Specifies the number of the Relocation Register (0 to 7).

Example

```
_$3R/ [RET]
$3R/ 177777
_125670;3R [RET]
_$3R/ [RET]
$3R/ 125670
```

Shows how to use the R command.

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 ZAP 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.

Some of the ZAP examples in this section are based on information contained in the following excerpts from a sample TKB 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.

1. Excerpts from TKB map are as follows:

```
MAINMEO.TSK;1   Memory allocation map  TKB M40.10      Page 1
                     14-MAR-87   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-87   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-87   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-87   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
```

For 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 TKB map. The sequence of ZAP commands is as follows:

```
>ZAP [RET]
ZAP>MAINMEO/LI [RET]

ZAP Version V04.00 COPYRIGHT (c) DIGITAL EQUIPMENT CORPORATION 1987
Segment Table
000002: 000000-022137 MAIN0  I-space root
000025: 000000-014123 MAIN0  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
```

The first command line invokes ZAP and the second command line requests the segment table for the task MAINMEO. The /LI switch directs ZAP to give the starting disk block for the root segment of the task (in this example, MAIN0) and for each segment overlaid on the root of the task. The /LI 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 MAIN0. 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.

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 MACRO FILE    MACRO M1113  18-MAR-87 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 sequence of ZAP commands is as follows:

```
_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.

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-87 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 as follows:

```
_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).

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-87 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 as follows:

```
_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.

Error messages describe a condition that caused ZAP to terminate the processing of a command. When this occurs, ZAP returns to the highest level of command input. For example, if the command is entered in response to the Monitor Console Routine (MCR) prompt, ZAP will issue the error message, exit, and prompt you with the MCR prompt, as follows:

```
>ZAP @indirectcommandfile
ZAP -- message
>
```

If an error occurs during the processing of an indirect command file, the command file is closed, the error message—followed by the command line in error—is issued to your terminal, and ZAP returns to the highest level of command input.

If the command is entered in response to the ZAP prompt, ZAP will issue the error message and prompt you with the ZAP prompt, as follows:

```
ZAP>commandline
ZAP -- message
ZAP>
```

Error messages are issued to your terminal in the following format:

```
ZAP -- message
```

The ZAP error messages are as follows:

ZAP—Address not within segment

Explanation: The address specified was not within the overlay segment specified.

User Action: Specify the correct address or overlay segment number, and reenter the command line.

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: Use the correct file specification and reenter the command line.

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: Specify the correct value and reenter the command line.

ZAP—No such relocation register

Explanation: An invalid number was specified for a Relocation Register.

User Action: Relocation Registers are numbered 0 to 7. Any other numbers are illegal. Specify a valid Relocation Register number and reenter the command line.

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: Specify the correct disk block address and reenter the command line.

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, and then try invoking ZAP with the /AB switch specified.

ZAP—Not implemented

Explanation: You entered a command that is recognized by ZAP, but the command is not implemented.

User Action: Ensure that you entered the command correctly and reenter the command line.

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 the Peripheral Interchange Program (PIP) to check your file access privileges (see Chapter 12).

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 MCR or DCL command INSTALL in the *RSX-11M-PLUS MCR Operations Manual* or the *RSX-11M-PLUS Command Language Manual*.)

ZAP—Too many arguments

Explanation: You entered more arguments on the command line than are allowed.

User Action: Specify the correct syntax and reenter the command line.

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, specify the correct syntax, and reenter the command line.

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
Command Formats

A.1 Introduction to Command Formats

This appendix presents a summary of the commands and switches used by the RSX–11M–PLUS utilities described in this manual. Each section number of the appendix corresponds to the chapter discussing that utility. For example, Chapter 12 and Section A.12 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 BAD

The format for executing BAD is shown next.

Format

ddnn:[/switch[...]]

Parameters

ddnn:

Specifies a physical device.

switch

Specifies one of the following switches:

/ALO:volume label	Prompts you for blocks to be allocated to BADBLK.SYS and entered in the bad block descriptor file.
/LI	Lists bad blocks as they are located.
/MAN	Prompts you for additional bad blocks, which are entered in the bad block descriptor file.
/OVR	Creates the bad block descriptor file on a last-track device.
/PAT=m:n	Specifies the doubleword data pattern used to locate bad blocks.
/RETRY	Recovers soft errors.
/UPD	Reads the bad block descriptor file and prompts for input.

A.3 BRU

The command format for BRU is shown next.

Format

/qualifier[...] indevice[,...][filespec[,...]] outdevice[,...]

Parameters

indevice

Specifies the physical device (or devices) from which data is transferred.

filespec

Indicates the file specification used to select files to be backed up or restored.

outdevice

Specifies the output device to which data is transferred..

qualifier

Specifies one of the following qualifiers:

/APPEND	Appends 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).
BACKUP_SET:name	Specifies the name of the backup set.
/BAD { :AUTOMATIC :MANUAL :OVERRIDE }	Speccifies the locations of bad blocks on the volume. The default option is :AUTOMATIC.
/BUFFERS:n	Specifies the default number of directory File Control Blocks (FCBs) per volume kept by the Ancillary Control Processor (ACP).
/COMPARE	Specifies that the data on the output volume should be compared to the data on the input volume and any differences reported.
/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 backup or restore files created before or after a specified date and/or time.
/DENSITY: { 800 1600 6250 }	Specifies the data density at which BRU writes to tape.
/DIRECTORY	Displays information (such as backup set names, file names, and volume number) for a specified volume.
/DISPLAY	Displays at your terminal the directory and file name of each file as the header of that file is transferred by BRU.
/ERRORS:n	Terminates the restore operation after the specified number of nonfatal tape-read errors is reached. The default is 25_{10} errors.
/EXCLUDE	Excludes from a backup or restore operation all files specified on the command line.

/EXTEND:n	Specifies the default number of blocks by which a file is extended when that file has exhausted its allocated space.
/HEADERS:n	Specifies the number of file headers to allocate initially to the index file.
/IDENTIFICATION	Directs BRU to identify itself by displaying its version.
/IMAGE { SAVE | RESTORE }	Specifies that you want to do a multiple disk-to-disk backup (SAVE) or restore (RESTORE) operation.
/INITIALIZE	Directs BRU to initialize the output disk during a tape-to-disk or disk-to-disk operation.
/INVOLUME:name	Specifies the volume label of the input disk.
/LENGTH:n	Specifies the length of the output tape in decimal feet.
/MAXIMUM:n	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 as a Files–11 volume.
/NEW_VERSION	Directs BRU to resolve file specification conflicts that occur during restore operations and during backups to a mounted disk 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 IDs.

/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. The /NOSUPERSEDE qualifier is the default.

/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:n }

Specifies the location of the index file on the output disk volume being initialized.

/PROTECTION: { SYSTEM:value | OWNER:value | GROUP:value | WORLD:value }

Specifies the default protection status for all files created on the output volume being initialized. The protection value can be any combination of the following: R (read), W (write), E (extend), or D (delete).

/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 back up or restore files revised or created before, or after, a specified date and/or time.

/REWIND

Rewinds the first tape of a tape set before executing a backup or restore operation.

/SUPERSEDE

Specifies that when file specifications on the mounted input 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. The default is the /NOSUPERSEDE qualifier.

/TAPE_LABEL:label

Specifies the 6-character volume identifier on the ANSI 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.

/UFD	Directs BRU to create directories (if they do not already exist) on a mounted output volume, and then to copy the files from the same directories on the input volume into the created directories.
/VERIFY	Verifies that the output volume was written correctly by comparing the input volume to the output volume and reporting any differences.
/WINDOWS:n	Specifies the default number of mapping pointers to be allocated for file windows when initializing an output disk. The default number of mapping pointers is the same for the input disk.

A.4 CMP

The format of the CMP command line is shown next.

Format

[outfile[/switch[...]]=]infile1,infile2

Parameters

outfile

Indicates the file specification for the output file.

infile1

Specifies the file specification for the input file to be compared to infile2.

infile2

Specifies the file specification for the output file to be compared to infile1.

switch

Specifies one of the following switches applied to the output file specification:

/BL [/-BL]	Specifies that blank lines in both files be included in compare processing. If the /-BL switch is specified, blank lines are not included in compare processing. The /-BL switch is the default.
/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. The /-CB switch is the default.
[/CO] /-CO	Specifies that CMP include comments (that is, text preceded by a semicolon (;)) in compare processing. The /CO switch is the default.
[/DI] /-DI	Specifies that CMP list the differences between the two files (rather than marking the lines in infile2). The /DI switch is the default.
/FF [/-FF]	Specifies that CMP include records consisting of a single form-feed character in compare processing. The /-FF switch is the default.
/LI:n [/LI:3]	Specifies that a number of lines must be identical before CMP recognizes a match. The /LI:3 switch is the default.
[/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. The /LN switch is the default.
[/MB] /-MB	Specifies that CMP include all blank and tab characters in a line in compare processing. If you specify the /-MB switch, 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. The /MB switch is the default.
/SL[:au] [/-SL]	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 in the /SL switch (:au), an audit trail is specified for SLP input. The /-SL switch is the default.
/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. The /-SP switch is the default.
[/TB] /-TB	Specifies that CMP include all trailing blanks on a line in compare processing. If you specify the /-TB switch, CMP ignores all blanks following the last nonblank character on a line. When you specify the /-CO and the /-TB switches together, blanks that precede a semicolon (;) are considered trailing blanks and are ignored. The /TB switch is the default.
/VB:nnn [/VB:041]	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 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). The /VB:041 switch (for the exclamation mark) is the default.

A.5 DMP

The DMP command line format is shown next.

Format

[outfile][/switch[...]][=inspec][/switch[...]]

Parameters

outfile

Specifies the output file specification.

inspec

Specifies the input device and file, or the input device only.

switch

Specifies one of the following DMP switches:

/AS	Specifies that data be dumped one byte at a time in ASCII mode.
/BA:n:m	Specifies a 2-word base block address, where n is the high-order base block address and m is the low-order base block address.
/BL:n:m	Specifies the range of blocks to be dumped, where n is the first block and m is the last block.
/BY	Specifies that the data be dumped in octal byte format.
/DC	Specifies that the data be dumped in decimal word format.
/DENS: { 800 / 1600 }	Specifies the density of an input magnetic tape when DMP is in device mode.
/FI:n:sequence	Specifies in file mode that the file number be used instead of the file name as a file specification for input.
/HD[:F] /HD:U	Specifies that the file header as well as the specified or implied portion of the file be dumped. The /HD:F switch is the default, and it causes a Files–11 formatted dump of the header. The /HD:U switch causes an unformatted octal dump.
/HF	Specifies the format for data blocks that have the Files–11 header structure. Other blocks are output as a data dump in the format selected by switches such as /AS and /BY in default octal words.
/HX	Specifies that data be dumped in hexadecimal byte format.
/ID	Causes the current version of DMP to be displayed.
/LB	Requests logical block information for a file and displays the starting block number and a contiguous or noncontiguous indication for the file.
/LC	Specifies that the data should be dumped in lowercase characters. This switch can be used only if the output device has lowercase capability.

/LIM:n:m	Specifies the range of bytes n to m of each record or block to be dumped.
/LW	Specifies that data be dumped in hexadecimal doubleword format.
/MD[:n]	Specifies a memory dump and allows control of line numbers.
/OCT	Specifies that the data should be dumped in octal format in addition to the other formats specified.
/RC	Specifies that data should be dumped one record at a time.
/RW	Issues a rewind command to the tape driver before referencing a specified tape. The /RW switch may be specified at any time to re-position a tape at beginning-of-tape (BOT).
/R5	Specifies that data be dumped in Radix–50-format words.
/SB:n /SB:-n	Specifies the number of blocks DMP spaces forward (n) or backward (-n) on tape.
/SF:n /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 output file to the line printer.
/WD	Specifies that data be dumped in hexadecimal word format.

A.6 DSC

The DSC command line format is shown next.

Format

outdev[,...][filelabel1][/switch[...]]=indev[,...][filelabel2][/switch[...]]

Parameters

outdev

Specifies the physical volume (or volumes) to which data is copied.

filelabel1

Specifies the output disk's Volume ID, the tape file, or the tape set that DSC creates in a file transfer.

indev

Specifies the physical volume (or volumes) from which data is copied.

filelabel2

Specifies the DSC-created tape file that is being transferred to disk or compared.

switch

Specifies one of the following switches:

/AP	Appends a DSC file to the first volume of a tape set that already contains a DSC file.
/BAD= { MAN | NOAUTO | MAN:NOAUTO | OVR | MAN:OVR }	Allows manual entry of bad block locations; it can supplement, override, or ignore the disk's own bad block file.
/BL=n /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.
/DENS= /DENS: { 800 | 1600 }	Overrides the DSC default storage density for magnetic tapes of 800 bpi.
/RW	Rewinds all magnetic tapes before DSC executes the current command.
/VE	Copies data from the input volume and compares it with data in the output volume.

A.7 EDI

The commands for the EDI editor are shown next.

Commands

A string

Causes the specified string to be appended to the current line.

ALTMODE key

Causes the system to print the previous line in the block in block mode only. That line becomes the current line.

AP string

Causes the specified string to be appended to the current line, and displays the new current line.

B

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.

BL [ON]
BL OFF

Switches text between block mode and line-by-line mode.

BO

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.

[n]C /string1/string2[/]

Searches for and replaces string1 with string2 the number of times specified in the current line.

CC [character]

Changes the current concatenation character to the specified character. The default is the ampersand character (&).

CD [filespec]

Transfers all remaining lines in the block buffer and then the input file into the output file, closes both files, and then deletes the input file. If filespec is specified, the output file is renamed.

CL [filespec]

Transfers all remaining lines in the block buffer and then the input file into the output file and closes both files If filespec is specified, the output file is renamed.

CLOSES

Closes the secondary input file.

CTRL/Z

Closes files and terminates the editing session.

D [n]
D -n

Deletes the current line 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. The Delete [-n] command is valid only in block mode.

DP [n]
DP -n

Performs the same function as the Delete command except that the new current line is displayed.

E

Performs the same function as the BOttom command.

ED [filespec]

Transfers all remaining lines in the block buffer and then the input file into the output file, closes both files, deletes the input file, and terminates the editing session. If a file specification is specified, the output file is renamed to that file name.

ERASE [n]

Erases the current line in line-by-line mode, or erases the current block buffer and the next n-1 blocks in block mode.

ESCAPE key

Prints the previous line in the block (block mode only) making it the new current line. Pressing the ESACPE key in input mode terminates the line of input and, if entered as the first character in the line, exits from input mode.

EX [filespec]

Transfers all remaining lines in the block buffer and input file (in that order) into the output files, closes the files, and terminates the editing session. If a file specification is specified, the output file is renamed to that file name.

[n]F [string]

Searches the block buffer or input file for a string. If n is specified, the nth occurrence of the string is found.

FF

Inserts a form feed into the text to delimit a pages.

FIL filespec

Transfers lines from the input file to both the output file and the specified file in line-by-line mode only.

I [string]

Inserts a string immediately following the current line. If a string is not specified, EDI enters input mode.

KILL

Closes the input file and deletes the output file.

[n]L [string]

Searches the block buffer or input file for the nth occurrence of a string or, if a string is not specified, considers the current line a match and positions the line pointer there.

[n]LC /string1/string2[/]

Changes all occurrences of string1 in current line and the next n-1 lines to string2. If string2 is not specified, all occurrences of string1 are deleted.

LI

Prints on your terminal all remaining lines in the block buffer or input file, beginning with the current line.

LP

Lists the remaining lines in the block buffer or input file on the pseudo device CL, beginning with the current line.

[n]Mx [a]

Executes n times the macro x, while passing it an optional numeric argument a.

MACRO x definition

Defines macro number x. The value of x can be 1, 2, or 3.

MC

Retrieves up to three macro definitions previously stored in the latest version of the file MCALL;n.

[n] <definition>

Defines and executes, in one step, a macro n times. The macro definition is copied into the macro 1 storage area.

N [n]
N -n

Moves the current line pointer backward (-n) or forward (n) a number of lines from the current position.

NP [n]
NP -n

Moves the current line pointer backward (-n) or forward (n) a number of lines from the current position and prints the new current line.

O [n]

Deletes the current line and the next n-1 lines, enters input mode, and inserts your typed text into the file until you enter a RETURN key as the only character in an input line.

OP filespec

Opens the specified secondary input file.

OU [ON] or
OU OFF

Continues or discontinues the transfer of text to the output file in line-by-line mode only.

P [n]

Prints the current line and the next n-1 lines on your terminal. The last line printed becomes the new current line.

PA /string1/string2[/]

Changes all occurrences of string1 to string2. Same as the 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.

PAG n

Enters block mode, if not already in block mode, and reads page n into the block buffer.

[n]PF string

Searches successive block buffers until the nth occurrence of the string is found, and copies 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 in the output file.

[n]PL string

Searches successive block buffers for the nth occurrence of the string and writes the text from the current block buffer into the output file.

R [string]

Replaces the current line with the string. If a string is not specified, the current line is deleted.

REA [n]

Reads the next n blocks of text into the block buffer. If a block is already in the buffer, the new block or blocks are appended to it.

REN [n]

Writes the current block buffer into the output file and reads a new block from the input file n times in block mode only.

RETURN key

Prints the next line in the file or block buffer and makes it the current line. In input mode, pressing the RETURN key causes EDI to change to edit mode.

SA [n] [filespec]

Saves the current line and the next n-1 lines in the file specified and creates a new version if the file already exists. If file specification is not given, the lines are saved in file SAVE.TMP.

SC /string1/string2[/]

Searches for string1 in the input file or block buffer and replaces it with string2.

SIZE n

Specifies the maximum number of lines to be read into the block buffer on a single REA or REN command.

SP

Selects or reestablishes the primary file for input.

SS

Selects the secondary file as the input file.

T

Moves the current line pointer to the top of the current block or file.

TA [ON]
TA OFF

Turns automatic tabbing on or off.

TOF

Copies the input file into the ouput file, closes both files, and opens the latest version of the output file as the input file. If TOF is entered while in line-by-line mode, EDI switches to block mode after saving the edited data. The first block is then read into the block buffer.

TY [n]

Prints the next n lines. In line-by-line mode, the last line printed becomes the new current line. In block mode, the current line pointer does not change unless end-of-buffer (EOB) is reached.

UC [ON]
UC OFF

Allows you to enter lowercase letters from a terminal and have them converted to uppercase letters.

UNS [filespec]

Retrieves all the lines in a specified file and copies them after the current line. If no file specification is given, the default file SAVE.TMP is used.

V [ON]
V OFF

Determines whether the lines specified by the L and C commands are verified or displayed.

W

Writes the entire contents of the current block buffer to the output file and then erases the block buffer in block mode only.

A.8 FLX

The FLX command line format is shown next.

Format

[ddnn:[[directory]]/switch[...]=]infile1[,...]/switch[...]

Parameters

ddnn

Specifies the device for the FLX output.

directory

Specifies the directory on the output device. If directory is not given, the default directory is used. Do not specify a directory if the output device is in RT-11 format.

infile

Specifies the file specification for input.

Compress Switch Format

outfile/CO:size:ept:mnt=infile

Parameters

outfile

Specifies the file specification that is to become the compressed version of the input file.

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. The number of EPTs in the old library file is the default value. The maximum number of entries is 4096_{10}.

mnt

Specifies the number of module name table (MNT) entries to allocate. 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 specification to be compressed.

Create Switch Format

outfile/CR:size:ept:mnt:libtype:infiletype

Parameters

outfile

Specifies the library file specification being created.

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. 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 Default switch (/DF) (see Section 10.3.2.4), or OBJ if the /DF switch 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.

Delete Switch Format

outfile/DE:module[...]

Parameters

outfile

Specifies the library file specification.

module

Specifies the name or names of the module or modules to be deleted.

Default Switch Formats

outfile/DF:filetype

or

/DF:filetype

Parameters

outfile

Specifies the library file specification.

filetype

Specifies the default library file type: OBJ for object library files, MAC for macro library files, and UNI for universal library files.

Delete Global Switch Format

outfile/DG:global[...]

Parameters

outfile

Specifies the library file specification.

global

Specifies the name or names of the entry point or points to be deleted. Up to 15_{10} entry points may be deleted with one command line.

Entry Point Switch Formats

outfile/EP=infile[,...]

or

outfile=infile/EP[,...]

Parameters

outfile

Specifies the output file specification.

infile

Specifies the input file specification.

Extract Switch Format

outfile=infile/EX[:modulename[...]]

Parameters

outfile

Specifies the file specification into which extracted modules are to be stored. If the input modules are object modules, the default file type for this file is OBJ. If the input modules are macro modules, the default file type is MAC. 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 specification 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.

modulename

Specifies the name or names of the module or modules to be extracted from the library.

Insert Switch Format for Object and Macro Libraries

outfile/IN=infile[,...]

Parameters

outfile

Specifies the library file specification into which the input modules are to be inserted. The default file type depends on the current default (see Section 10.3.2.4). It is OLB if the current default is object libraries, MLB if the current default is macro libraries.

infile

Specifies the input file specification 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.

Insert Switch Format for Universal Libraries

outfile=infile/IN:name[:op[...]]

Parameters

outfile

Specifies the universal library file specification into which the infile is to be inserted.

infile

Specifies the input file specification 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 10.3.2.2.)

name

Specifies the module name (up to six Radix–50 characters) optionally. 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.

List Switch Format

outfile[,listfile]/switch

Parameters

outfile

Specifies the library file specification whose contents are to be listed.

listfile

Specifies the listing file specification optionally. If not specified, the listing is directed to the initiating terminal.

switch

Specifies one of the list switches (/LI, /LE, and /FU) that is used to produce a printed listing of the contents of a library file. The three switches allow you to select the type of listing desired. The functions of these switches are as follows:

/LI Lists the names of all modules in the library file.

/LE Lists the names of all modules in the library file and their corresponding entry points.

/FU Lists 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.

The /LI switch is the default value.

Modify Header Switch Format

outfile/MH:module[:op[...]]

Parameters

outfile

Specifies an output file specification for the universal library. The file type defaults to ULB.

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 number sign (#) clears the corresponding information field.

Replace Switch Formats for Macro and Object Libraries

outfile/RP=infile[,...]

or

outfile=infile[/RP][,...]

Parameters

outfile

Specifies the macro or object library file specification. The default file type depends on the current default (see Section 10.3.2.4). It is OLB if the current default is object libraries or MLB if the current default is macro libraries.

infile

Specifies the input file specification 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.

Replace Switch Formats for Universal Libraries

outfile/RP:name[:op[...]]=infile[,...]

or

outfile=infile/RP:name[:op[...]][,...]

Parameters

outfile

Specifies the universal library file specification.

infile

Specifies the input file specification 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.

name

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.

Spool Switch Formats

outfile,listfile/SP

or

outfile,listfile/-SP

Parameters

outfile

Specifies the library file specification.

listfile

Specifies the listing file specification.

Selective Search Switch Format

outfile=infile/SS[,...]

Parameters

outfile

Specifies the library file specification.

infile

Specifies the input file specification that contains modules to be searched selectively.

Squeeze Switch formats

outfile/SZ=infile[,...]

or

outfile=infile/SZ[,...]

Parameters

outfile

Specifies the library file specification.

infile

Specifies the input file specification that contains modules to be squeezed during insertion into the library file.

A.11 PAT

The PAT command line format is shown next.

Format

[outfile]=infile[/CS[:n]],correctfile[/CS[:n]]

Parameters

outfile
Specifies the output file specification.

infile
Specifies the input file specification.

n
Specifies the value that PAT compares to the checksum value and returns an error message if the two values are different.

A.12 PIP

The PIP command line formats are shown next.

Append Switch Format

outfile=infile[,...]/AP[/FO]

Parameters

outfile
Specifies the output file specification.

infile
Specifies the input file specification. If the file name, file type, and version are not specified, then *.*;* is the default.

Block Size Switch Format

outfile/BS:n=infile

Parameters

outfile
Specifies the output file specification.

n
Specifies the number of bytes in a block in octal or decimal.

infile
Specifies the input file specification.

Creation Date Switch Format

outfile/CD=infile[,...]

or

outfile=infile/CD[,...]

Parameters

outfile

Specifies the output file specification.

infile

Specifies the input file specification.

Default Date Switch Format

/DD:startdate:enddate

Parameters

startdate

Specifies the beginning date of the time period in the form dd-mm-yy. It may be unlimited by using the wildcard character (*).

enddate

Specifies the ending date of the time period in the form dd-mm-yy. It may be unlimited by using the wildcard character (*).

Delete Switch Format

infile[,...]/DE[/LD]

Parameter

infile

Specifies the input file specification.

Default Switch Format

ddnn:[directory]/DF

or

ddnn:/DF

or

[directory]/DF

or

/DF

Parameters

ddnn

Specifies the new default device to be applied to subsequent PIP command lines.

directory

Specifies the new default directory to be applied to subsequent PIP command lines.

End-of-File Switch Format

infile/EOF[:block:byte][,...]

Parameters

infile

Specifies the input file specification.

The file specification must be issued. There is no default.

block

Specifies the block number where the EOF pointer is to be placed. The block number can be octal or decimal.

byte

Specifies 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.

Enter Switch Format

outfile=infile[,...]/EN[/NV]

Parameters

outfile

Specifies the file specification for the new directory entry.

infile

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

File Exclusion Switch Format

filespec/EX

Parameter

filespec

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

File Identification Switch Format

outfile=/FI:filenum:seqnum

Parameters

filenum

Specifies the file number.

seqnum

Specifies the sequence number of the file.

Free Switch Format

[ddnn:]/FR

Parameter

ddnn

Specifies the desired volume. If you do not specify a device, PIP defaults to SY0.

Identify Switch Format

/ID

List Switch Format

[listfile=]infile[,...]/LI[/subswitch]

Parameters

listfile

Specifies the file specification to be listed.

If listfile is not specified, it defaults to TI.

infile

Specifies the input file specification.

The default for infile is *.*;*.

subswitch

Specifies the alternate mode subswitch of the /LI switch as follows:

/BR	Specifies the brief form of directory listing. This subswitch lists only the file name, file type, and version.
/FU[:n]	Specifies the full directory format. If specified, n is the number of characters per line. If not specified, the number defaults to the buffer size of the output device.
/TB	This subswitch specifies the total number of blocks used for a directory and outputs the summary line.

Merge Switch Format

outfile=infile[,...][[/ME][/subswitch]]

Parameters

outfile

Specifies the output file.

infile

Specifies the input file.

subswitch

Specifies one of the following /ME subswitches as follows:

/BL:n	Specifies the number of blocks to be allocated initially to the output file.
/CO	Specifies that the output file be contiguous. The /-CO switch specifies that the ouput file does not have to be contiguous.
/FO	Specifies that the owner of the output file be the same as the output directory.
/NV	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.
/SU	Allows you to copy one or more input files to a file whose file name, file type, and version may already exist in a directory.

No Message Switch Format

infile[,...][switch]/NM

Parameters

infile

Specifies the input file specification.

switch

Specifies any combination of appropriate switches and subswitches.

Protect Switch Format

infile/PR[:n][/SY[:RWED]][/OW[:RWED]][/GR[:RWED]][/WO[:RWED]][/FO]

Parameters

infile

Specifies the file specification whose protection is being changed.

The file specification must be issued. There is no default.

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

Specifies the subswitches that set the protection level for a file. The subswitches are identified as follows:

Subswitch	Description
/SY	System
/OW	Owner
/GR	Group
/WO	World

n

Specifies a value that 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–PLUS Mini-Reference* for the list of octal codes.)

Purge Switch Format

infile[,...]/PU[:n][/LD]

Parameters

infile

Specifies the file specification to be deleted.

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

n

Specifies a range of obsolete versions of a file. If the latest version of the file is m, then all existing versions less than or equal to m-n are deleted. 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.

Rename Switch Format

outfile=infile[,...]/RE[/NV]

Parameters

outfile

Specifies the file specification to be given to the new file.

infile

Specifies the file specification to be renamed.

/NV

Specifies 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.

Remove Switch Format

infile[,...]/RM

Parameter

infile

Specifies the file specification for the directory file entry to be removed.

The file specification must be issued. There is no default.

Rewind Switch Formats

outfile/RW=infile

or

outfile=infile/RW

Parameters

outfile

Specifies the output file specification.

infile

Specifies the input file specification.

Span Blocks Switch Format

ddnn:outfile/SB=mmnn:infile

Parameters

ddnn

Specifies the output disk volume.

mmnn

Specifies the input tape volume.

outfile

Specifies the disk output file specification.

infile

Specifies the magnetic tape input file specification.

Selective Delete Switch Format

infile[,...]/SD

Parameter

infile

Specifies the input file specification.

The file specification must be issued. There is no default.

Spool Switch Format

infile[,...]/SP[:n]

Parameters

infile

Specifies the file specification to be spooled for printing.

n

Specifies the number of copies you want spooled.

Shared Reading Switch Format

outfile=infile/SR

Parameters

outfile

Specifies the output file specification.

infile

Specifies the input file specification.

Today Switch Format

/TD

Truncate Switch Format

infile[,...]/TR

Parameter

infile

Specifies the input file specification.

The file specification must be issued. There is no default.

User File Directory Switch Format

outfile/UF[/FO]=infile[,...]

Parameters

outfile

Specifies the file specification for the output file.

infile

Specifies the file specification for the input file.

Unlock Switch Format

infile[,..]/UN

Parameter

infile
Specifies the file specification for the file to be unlocked.

Update Switch Format

outfile=infile[,...]/UP[/FO]

Parameters

outfile
Specifies the file specification for the file to be rewritten.

infile
Specifies the file specification for the file to be copied into the file that is being rewritten.

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

A.13 SLP

The format of the SLP command line is shown next.

Format

outfile[/switch][,listfile][/switch]=[primary_input_device:[/switch]]

Parameters

oufile
Specifies the output file specification

listfile
Specifies the file specification for the listing file.

primary_input_device
Specifies that input for the file being created is coming from this device.

switch

Specifies one of the following SLP switches:

[/AU:position:length] /-AU	Enables or disables the audit trail, which indicates the changes made during the most recent editing session. If the audit trail is to be enabled, you can specify the beginning field (position) and the length of the field.
[/BF] /-BF	Positions the audit trail by inserting spaces instead of tabs at the end of text information.
/CM[:n]	Deletes the audit trails and any trailing spaces or tabs, and truncates the text at the specified horizontal position n.
/CS[:n]	Calculates the checksum value for the edit commands.
/DB [/-DB]	Enables or disables the double-spaced format in the listing file.
/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] /-SP	Enables or disables the spooling of listing files to a line printer.
/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 occurs when lines are truncated by the audit trail.

A.14 VFY

The format of the VFY command line is shown next.

Format

[listfile,scratchdev=][indev]/switch

Parameters

listfile

Specifies the output file specification.

scratchdev

Specifies the device in the form ddnn on which the scratch file produced by VFY is to be written.

indev

Specifies the volume to be verified in the form ddnn.

switch

Specifies one of the VFY switches as follows:

/DE	Resets the marked-for-deletion indicators.
/DV	Validates directories against the files they list.
/FR	Prints out the available space on a volume.
/HD	Deletes bad file headers on a volume. The All subswitch (/AL) allows the /HD switch to delete bad file headers without prompting the user.
/ID	Identifies the VFY version. This switch may be specified on a command line by itself at any time.
/LI	Lists the index file by file ID.
/LO	Scans the file structure looking for files that are not in any directory.
/RC	Checks the volume to see if every block of every file can be read.
/RE	Recovers blocks that appear to be allocated but are not contained in a file.
/UP	Allocates blocks that appear to be available but are actually allocated to a file.

A.15 ZAP

The general ZAP command line format is shown next.

Format

filespec/switch[...]

Parameters

filespec

Specifies the file you want changed. Refer to Chapter 1 for detailed descriptions of the parameters for file specifications.

switch

Specifies one of the following ZAP switches:

/AB	Specifies the Absolute Mode switch
/LI	Specifies the List switch which displays the overlay segment table for an overlaid task image file.
/RO	Specifies read-only mode.

A.15.1 ZAP Open and Close Commands

ZAP open and close commands allow you to open locations in a file, modify those locations, and close the locations.

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 marks)

Opens a location, displays the contents of the location as two ASCII characters, and stores the contents of the location in the Quantity Register (Q).

% (percent sign)

Opens a location, displays the contents of the location in Radix-50 format, and stores the contents of the location in the Quantity Register (Q).

\ (backslash)

Opens a location as a byte, displays the contents of the location in octal, and stores the contents of the location in the Quantity Register (Q).

' (apostrophe)

Opens a location, displays the contents as one ASCII character, and stores the contents of the location in the Quantity Register (Q).

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

ZAP general-purpose commands are used to calculate displacements, verify location contents, and exit from ZAP.

Commands

X

Exits from ZAP; returns control to the Command Line Interpreter (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.

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 (TKB) tasks. CRF is invoked by the cross-reference switch in the MACRO–11 or TKB command line. Once execution is complete, CRF builds cross-reference listings by using the execution-time symbol tables built by those tasks.

Two cross-reference listings are built for the TKB, 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 shows a page and line number reference to a type of symbol referenced during execution of the MACRO–11 assembler.

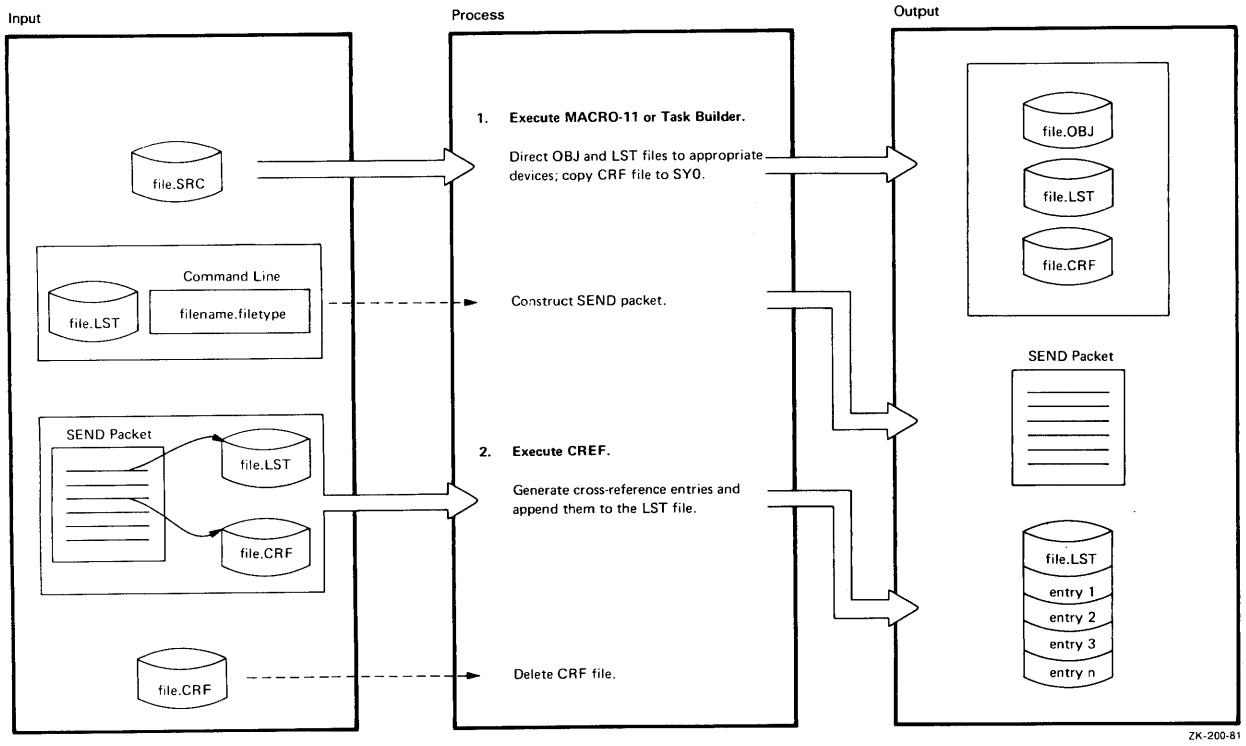
This appendix describes how CRF processes data and shows the formats of files CRF uses during that processing. Also, CRF error messages are listed.

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 TKB command line, refer to the *RSX–11M–PLUS and Micro/RSX Task Builder Manual*.

B.1 How CRF Processes Data

When the /CRO switch is specified in the MACRO–11 or TKB 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 TKB; 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.

Figure B–1: How MACRO–11, TKB, and CRF Generate Cross-Reference Listings



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 (TKB) and the /CRO switch. These tasks generate three files during execution: an object file (OBJ-type) or task image file (TSK-type), a listing file (LST-type or MAP-type), and a CRF symbol table file (CRF-type).

The object file or task image file is directed to an appropriate device and does not affect CRF processing.

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 TKB task 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 TKB.

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 (CRF-type), as follows:

```
Symbol table file name - Text file name
Symbol table file type - 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 this file.

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 file type of CRF, as follows:

```
filename.CRF
```

The header record contains the name of the originating task in 2-word Radix–50 format, which is 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.

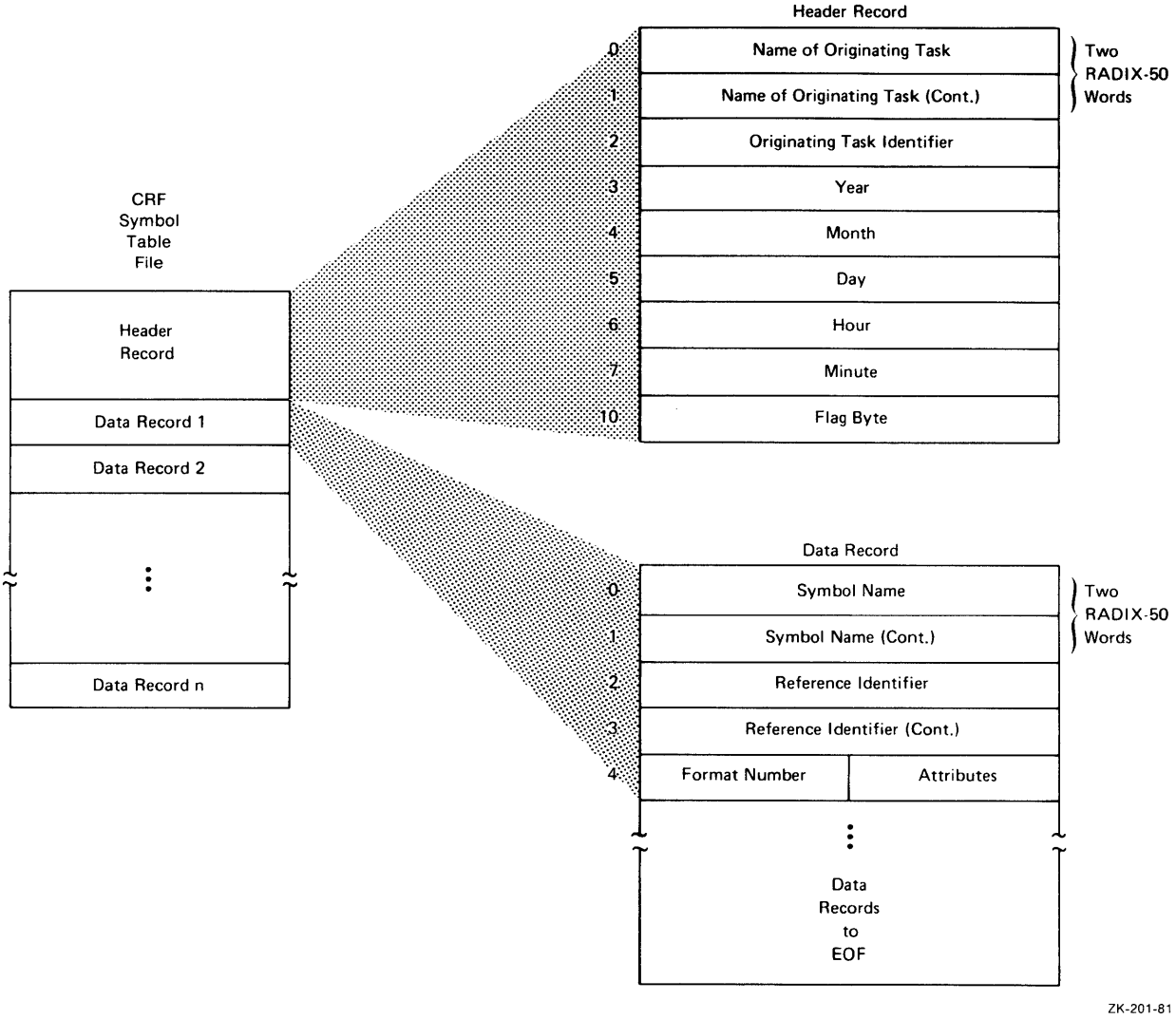
Bit 0 of the flags word controls the width of the listing. When this bit is set to zero, the listing is generated in 132-column format. When this bit is set to 1, the listing is generated in 80-column format.

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 number 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



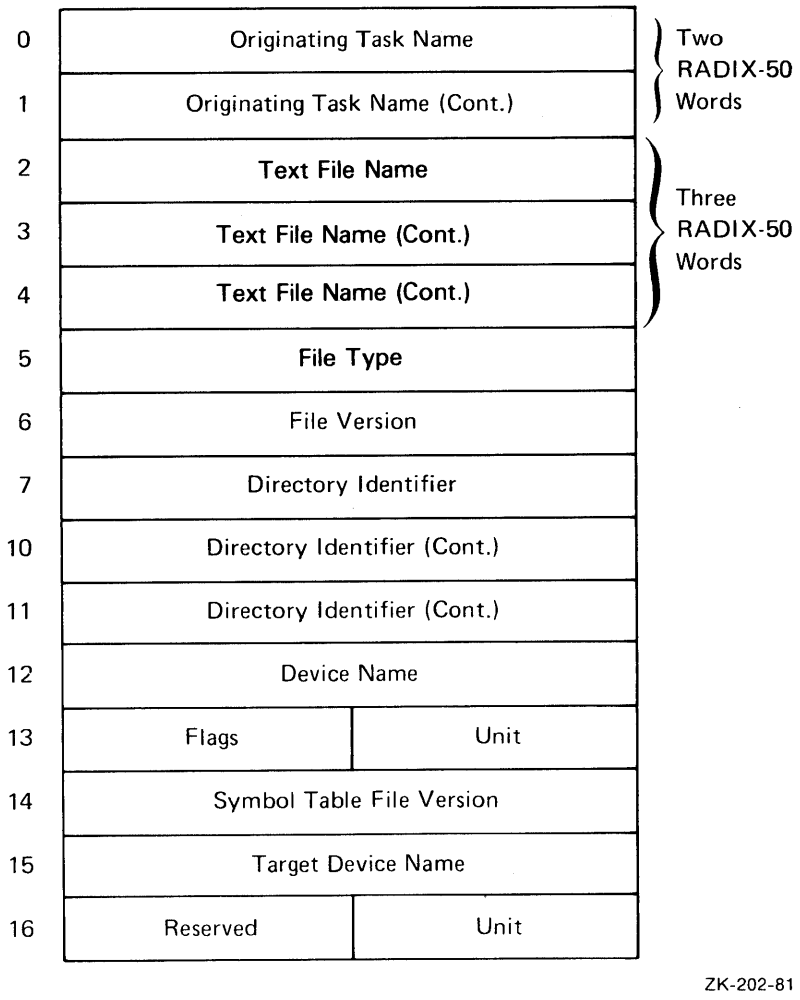
B.3 The CRF SEND Packet

The CRF SEND packet is a block of data that contains control information created by TKB 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 to 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 to 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 Messages

CRF displays fatal and diagnostic error messages. Each message is preceded by one of the following prefixes:

```
CRF -- *DIAG*  - name of originating task - message
CRF -- *FATAL* - name of originating task - message
```

Diagnostic errors will not terminate the CRF session and are displayed for your information.

Fatal error messages describe a condition that caused CRF to terminate the processing of a command. When this occurs, CRF returns to the highest level of command input. For example,

if the original command is entered in response to the Monitor Console Routine (MCR) prompt, CRF will issue the fatal error message, exit, and prompt you with the MCR prompt, as follows:

```
>utility commandline
>CRF -- *FATAL* - message
>
```

If the command is entered in response to the utility prompt, CRF will issue a fatal error message and prompt you with the utility prompt, as follows:

```
utility>commandline
CRF -- *FATAL* - message
utility>
```

The following error messages are output by CRF:

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 (SPR) with the related console dialog 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 directory 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 directory 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 an SPR 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 TKB 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 an SPR with the related console dialog 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 directory 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 an SPR with the related console dialog and any other pertinent information.

Unable to open workfile

Explanation: Possible causes are as follows:

- The work file device is not mounted.
- The work file device is write-protected.

The work file 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 work file. Possible causes are as follows:

- Device is full
- Hardware error

User Action: If the device capacity has been exceeded, delete unnecessary files to make space available. Also, reassign CRF LUN 7 to another Files–11 device.

Appendix C

Data Terminal Emulator (DTE) and the Micro/RSX File Transfer Utility (MFT)

C.1 Introduction

A local RSX terminal can log in to and conduct an interactive session with an external computer system. This session is established using the Data Terminal Emulator (DTE). The external system can be an RSX–11M–PLUS system, a VMS system running VAX–11 RSX, a Professional personal computer, or a Micro/RSX system. Once a local RSX terminal is logged in to an external system, the external system becomes the host system. The host system views the system running DTE as remote. During an interactive session, files can be transferred between the host system and the remote system. Files are transferred by using the Micro/RSX File Transfer Utility (MFT).

The following two conditions must exist for terminal emulation and file transfer to be operational:

- The two computers must be connected.
- The DTE software must be installed on the local RSX system, and MFT must be installed on the host system.

Refer to Sections C.2 and C.3 for more detailed information.

C.2 Connecting Two Computers

The two computers can be connected either on site or at remote sites through an asynchronous serial line. An RSX system is connected to an on-site external system by a cable and to a remote system by modems (DF03, DF112, DF224, and DFA01) and a telephone line. In either case, there is a dedicated line between the two systems that establishes the ports at each end of the line, which are assigned specifically for terminal emulation and data transfer.

A direct hardwired connection between two systems can be established by plugging one end of a cable into the communications port in the back of the local system unit and the other end of the cable to a host system. A permanent connection can also be established by using a DIGITAL Model H312A null modem cable and two standard EIA RS-232 connectors. When the two systems are directly connected with a cable, the terminal characteristics of the port at the local RSX system should be set to SLAVE and NOECHO. This prevents the transmission of

extraneous noise signals when the port is idle (disconnected from terminal emulation). Noise signals can create intersystem echo loops that seriously affect system performance.

Transmission speed between systems varies, depending on the systems involved. Variations are based on the following:

- Processor type
- Interface type
- System load
- Distance between systems
- Terminal support (use of higher baud rates requires that the remote system support XON /XOFF protocol)

If XON/XOFF is present, your terminal may run at a speed lower than that of the physical connection. For consistent performance, use a baud rate of 2400 for RSX systems.

Figures C–1 and C–2 show typical configurations.

Figure C–1: Terminal Emulation with an On-Site System

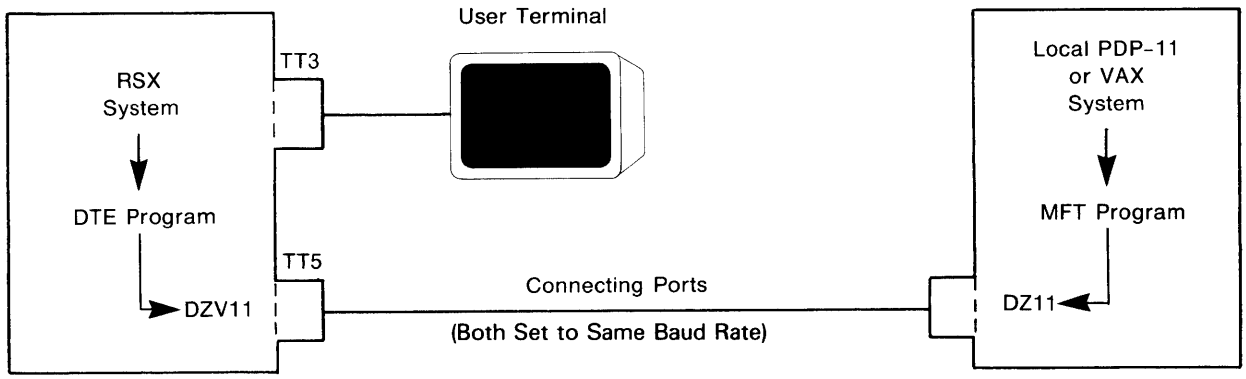
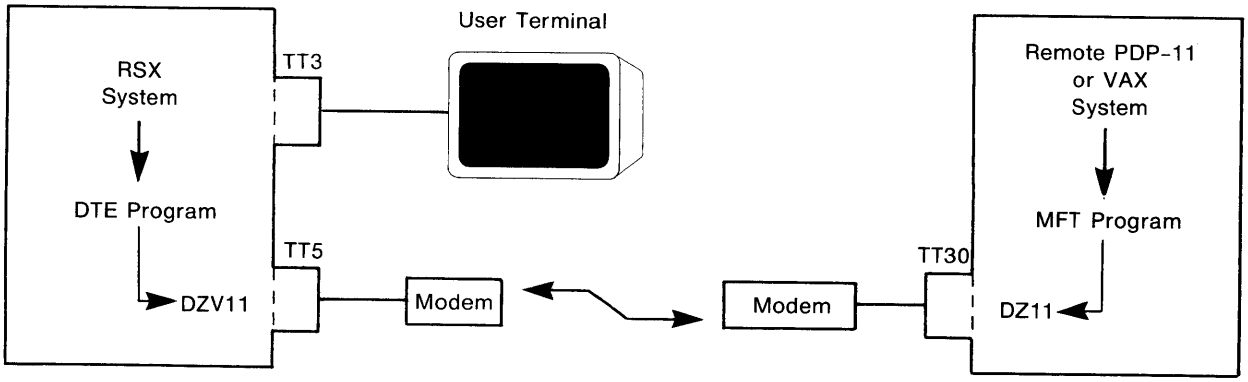


Figure C–2: Terminal Emulation with a Remote System



ZK-4126-85

In these example configurations, when the local RSX user terminal (TT3) is in terminal emulation, the following conditions exist:

- TT3 is connected to port TT5 using the DTE software.
- Port TT5 is connected to port TT30 in a PDP–11 system.
- Port TT5 is a connector on a DZV11.
- Port TT30 is a connector on a DZ11.

Port TT5 is set by using the following command line:

For the Monitor Console Routine (MCR):

```
>SET /NOECHO=TT5: RET
>SET /SLAVE=TT5: RET
```

For DIGITAL Command Language (DCL):

```
$ SET TERMINAL TT5:/SLAVE/NOECHO RET
```

C.3 Location of DTE and MFT

There are two separate programs that support terminal emulation and file transfer: one resides in the local RSX system and the other resides in the host system. DTE, which is installed on the local system, establishes the local terminal as an emulated terminal. Establishing terminal emulation means connecting a local RSX terminal to an external system so that the local terminal can be logged in as a user of the host system.

The other program is MFT, which interfaces with DTE when file operations are initiated at a local terminal. The MFT software is executed on the external computer system (the host) that is connected to the local (remote) system.

DTE and MFT tasks are separate tasks that work together, but they execute on two physically different systems. Also, note that all your input is done at a terminal connected to the local RSX system.

C.4 Establishing Terminal Emulation Using DTE

Once you have determined that the two specified computers are connected and that both DTE and MFT are installed on their respective systems, you can establish terminal emulation. You can establish a connection between a local RSX system and another system with the MCR command DTE or the DCL command SET HOST. The following sections describe the formats in detail.

C.4.1 Command Lines

The MCR command DTE and the DCL command SET HOST/DTE provide options that allow you to do the following:

- Dial out to the remote system through a modem (DF03, DF112, DF224, or DFA01).
- Specify the terminal as mute (SLAVE and NOECHO) on termination.
- Display the version of the DTE task.

The format of the command line is shown next.

Format for MCR

```
> DTE ttnn:[/option]
```

Format for DCL

```
$ SET HOST/DTE ttnn:[/option]
```

Parameters

ttnn

Specifies the RSX device and unit specification of the terminal port used for terminal emulation.

option

Specifies one of the following options:

/DIAL="..." Allows the specification of a dial command string for the modem. The string may consist of any of the digits 0 to 9, and one of the following symbols:

- Equal sign (=)
- Asterisk (*)
- Hyphen (-)
- Uppercase A,B,C,D,P,T, and W
- Lowercase a,b,c,d,p,t, and w
- A space
- Number sign (#)
- Left parenthesis
- Right parenthesis

The equal sign indicates that another dial tone is expected (for example, when you dial from an internal telephone system to an outside telephone number). The other symbols provide information for the modem being used. See the manual that accompanies your modem for more information.

Note that when you specify /DIAL for a DFA01 modem, the modem automatically changes the terminal speed, if necessary.

/MUTE Specifies whether DTE should alter certain device characteristics prior to exiting. The default is to return all of the device characteristics to their original states. However, if the device is set NOSLAVE and ECHO, it is possible that noise characters can create intersystem echo loops, which can severely affect performance. Therefore, it is recommended that either the device be set to SLAVE and NOECHO when it is idle (DTE will change these characteristics), or that the /MUTE option be specified, which will set the device to SLAVE and NOECHO when DTE exits.

/VERSION Displays the version of the DTE task for DCL only.

/ID Displays the version of the DTE task for MCR only.

After you establish terminal emulation with either the MCR or DCL command line, you are connected to the host system. You can then log in as a user of that system and use the file transfer utility (see Section C.5).

To terminate emulation, log off the host system and press CTRL/P. DTE displays the following message:

```
%DTE-S-EMUEXIT, Emulation exiting... Please wait
```

You are then returned to command level execution at the local RSX system.

Be sure that you log out from the host system before exiting terminal emulation. If you are in terminal emulation and press CTRL/P before logging out from the host system, the terminal connection is returned to the local RSX system. The host system remains logged in to your

account. Under these conditions, if another user initiates terminal emulation, that user will be logged in to the host system under your account.

To use DTE from the system's console terminal, you must first disable the console On-Line Debugging Tool (ODT). Both console ODT and DTE use CTRL/P. (For more information on ODT, refer to the *RSX-11M-PLUS and Micro/RSX Debugging Reference Manual.*) The following examples show various uses of the MCR command DTE and the DCL command SET HOST /DTE.

Examples for MCR

```
>DTE TT5:/MUTE  RET
```

Runs the DTE task, which establishes terminal emulation by connecting your terminal to port TT5. When data terminal emulation is terminated, port TT5 is set to SLAVE and NOECHO.

```
>DTE TT40:  RET
```

Invokes DTE to establish terminal emulation using terminal TT40.

```
>DTE TT50:/DIAL="5550837"  RET
```

Initiates terminal emulation using terminal TT50, after directing the modem to dial the specified number and establish a modem connection.

```
>DTE TT25:/MUTE  RET
```

Establishes terminal emulation using terminal TT25, and sets the terminal characteristics to SLAVE/NOECHO on termination.

Examples for DCL

```
$ SET HOST/DTE TT5:/MUTE  RET
```

Runs the DTE task, which establishes terminal emulation by connecting your terminal to port TT5. When data terminal emulation is terminated, port TT5 is set to SLAVE and NOECHO.

```
$ SET HOST/DTE TT40:  RET
```

Invokes DTE to establish terminal emulation using terminal TT40.

```
$ SET HOST/DTE TT50:/DIAL="5550837"  RET
```

Initiates terminal emulation using terminal TT50, after directing the modem to dial the specified number and establish a modem connection.

```
$ SET HOST/DTE TT25:/MUTE  RET
```

Establishes terminal emulation using terminal TT25, and sets the terminal characteristics to SLAVE/NOECHO on termination.

C.5 Performing File Operations Using MFT

The MFT utility deletes and transfers files by using a local (remote) RSX system, data terminal emulation (DTE), and an external (host) system. Special code in the DTE task is used to identify and perform file operations requested by the MFT task. To use MFT, you must be using the DTE task because the protocol used for the file operations is unique to the two tasks.

File transfer is performed using an error detection and correction algorithm to ensure data integrity. Your local terminal is locked out during file transfer operations to prevent interference with the data transfer between the two systems. You are returned to terminal emulation when the file operation completes.

Note

The speed of file transfer varies with the processor and interface types, system load, and software. Support for variable type-ahead buffer size allows for higher transfer rates.

The command syntax used to perform file operations with MFT depends on the command line interpreter (CLI) used by the host operating system. If your local terminal is a Professional 300 Series computer, you may want to use the Professional Forms Interface (PFT) or PFT Command Line Format (CFT) instead of MFT.

To use PFT, use the MCR command MFT /FORM or install MFT as an MCR spawnable task with the task name ...PFT. To use CFT, substitute the command MFT for the command PFT on all command lines shown in the *PRO/Communications User's Guide*. (For a complete description of PFT and CFT command lines, see the *PRO/Communications User's Guide*).

C.5.1 Command Line for Copying Files

To copy files from one system to the other, in either direction, use the command line format shown next.

Format for MCR

```
> MFT outfile[/REM]=infile[/REM]
```

Format for DCL

```
$ COPY infile[/REMOTE] outfile[/REMOTE]
```

Parameters

outfile

Specifies the output file or where the file is being copied to.

infile

Specifies the input file or where the file is being copied from.

/REM

Specifies the /REM option and indicates the file is on your local RSX system and is remote to the host system.

/REMOTE

Specifies the /REMOTE option and indicates that the file is on the local RSX system that is remote to the host system. The absence of the /REMOTE option indicates the file is on the host system.

For input files, the /REMOTE option indicates that the file resides on the DTE system. For output files, the /REMOTE option indicates that the output file should be created on the DTE system.

The following examples show how to copy files.

Examples for MCR

```
>MFT MESSAGE.TXT/REM=HOST.TXT [RET]
```

Copies the file HOST.TXT from the host system to the file MESSAGE.TXT on the RSX system.

```
>MFT MESSAGE.TXT=RSX.TXT/REM [RET]
```

Copies the file RSX.TXT from the RSX system to the file MESSAGE.TXT on the host system.

Examples for DCL

```
$ COPY INFILE.DAT OUTFILE.DAT/REMOTE [RET]
```

Copies the file INFILE.DAT from the host (MFT) system into OUTFILE.DAT on the remote (DTE) system.

```
$ COPY INFILE.DAT/REMOTE OUTFILE.DAT [RET]
```

Copies the file INFILE.DAT from the remote (DTE) system into OUTFILE.DAT on the host (MFT) system.

C.5.2 Command Line for Deleting Files

To delete files using MFT, use the format shown next.

Format for MCR

> MFT delfile/DE[/REM]

Format for DCL

$ DELETE delfile[/REMOTE]

Parameters

delfile

Specifies the name of the file to be deleted. The file name cannot include wildcards.

/DE

Specifies the file is to be deleted.

/REM

Specifies the file is on the RSX system remote to the host system. Do not specify the /REM option if the file is on the host system.

/REMOTE

Specifies the /REMOTE option and indicates that the file is on the local RSX system that is remote to the host system. The absence of the /REMOTE option indicates the file is on the host system. When using the DELETE command with the /REMOTE option, the file to be deleted should be deleted on the DTE system.

The following examples show how to delete files.

Example for MCR

```
>MFT MESSAGE.TXT/DE [RET]
```

Deletes the file MESSAGE.TXT from the host system.

Example for DCL

```
$ DELETE FILE.DAT/REMOTE [RET]
```

Deletes the file FILE.DAT on the remote (DTE) system.

C.5.3 Example of File Transfer Session

An MCR example of a file transfer session that uses DTE and MFT is as follows:

```
>DTE TT5: [RET]
>HELLO [RET]
>USERNAME: username [RET]
>PASSWORD: <password> [RET]
    .
    .
    .
>MFT MESSAGE.TXT=MICRO.TXT/REM [RET]

%DTE-S-MFTINIT, File operation initiated

%DTE-S-MFTCOMP, File operation complete

   (The cursor may be at the beginning of the last message. To
   continue the operation, press RETURN.)

[RET]

MFT>[CTRL/Z]
>BYE [RET]
    .
    .
    .
>[CTRL/P]
%DTE-S-EMUEXIT, Emulation exiting... Please wait
>
```

C.6 Messages

The following sections list and describe the messages produced by DTE and MFT.

C.6.1 DTE Messages

The following messages are produced by DTE. Some messages also include a 6-character octal field enclosed in parentheses. This field provides additional internal information regarding the cause of the error, but it is generally not useful to you unless the information is also mentioned in the message description.

?DTE-F-DEVTYPE, Illegal device type for DTE

Explanation: DTE assigned the specified emulation device and found that it was not a terminal.

User Action: No user action is required. DTE can be used only on terminal devices.

%DTE-S-DF03SEQ, Dialing... Please hold

Explanation: You specified the DTE /DIAL option and DTE is attempting to establish a connection by using a modem.

User Action: No user action is required. Your terminal is locked out until the modem responds with either success or failure.

?DTE-F-DF03TMO, Telephone connection failure due to timeout

Explanation: DTE sent the dial sequence specified by the /DIAL option, and the modem did not respond within 40 seconds.

User Action: No user action is required. DTE will attempt to hang up the line and then exit with an error status.

%DTE-S-DIALHNG, Dialup line hung up

Explanation: DTE is attempting to hang up the specified emulation terminal that is connected to a modem. It may or may not have succeeded in hanging up.

User Action: No user action is required. This is an informational message.

%DTE-I-DTETVER, Data Terminal Emulator Version x.y ([dd-mmm-yy])

Explanation: You specified the /IDENTIFICATION option in the command line, so DTE is displaying its version number.

User Action: No user action is required. This is an informational message.

%DTE-S-EMUEXIT, Emulation exiting... Please wait

Explanation: You pressed CTRL/P at your terminal, and DTE has begun exit processing. Your terminal is locked out, all DTE I/O processing is being completed, and all device characteristics are being restored.

User Action: No user action is required. This operation takes approximately 5 seconds.

?DTE-F-ERDDCMP, Transmission [DDCMP] failure (xxxxxx)

Explanation: During DTE/MFT file transfer, a DDCMP error occurred that could not be corrected after trying several times. The physical (emulation) terminal lines are producing too many errors for a file operation to be performed. Reasons for this type of error are as follows:

- Baud rates are too high.
- The systems are too far apart (the physical cable is too long).
- The system lacks the required terminal support.
- There is some electrical interference (noise) along the physical connection (because of a bad telephone connection or heavy electrical equipment).

User Action: Determine the reason for the error; correct that error, if possible; and try the operation again.

?DTE-F-FATALER, Internal error (xxxxxx)

Explanation: DTE/MFT has encountered an unexpected internal condition. This message indicates an error in the local file operations server within the DTE task.

User Action: Reenter the command. If the error persists, submit a DIGITAL Software Performance Report (SPR).

%DTE-S-MFTCOMP, File operation complete

Explanation: A DTE/MFT file operation has completed, and control has been returned to your terminal. If the operation has completed in failure, then this message is preceded by an error message showing the cause of the file operation failure.

User Action: No user action is required.

%DTE-S-MFTINIT, File operation initiated

Explanation: DTE has received the communications protocol ("handshake") sequence from the MFT task on the other system. DTE invokes the file transfer server to provide local support for the operation being initiated on the remote system.

User Action: No user action is required. Your terminal is locked out until the operation completes.

?DTE-F-MODERR, Unable to determine modem type

Explanation: DTE has either failed to receive a response from the modem or is unable to determine the type of modem (DF03, DF112, DF224, or DFA01) in use. This message may indicate faulty hardware.

User Action: Check to see that the modem is properly connected and that the modem switches and port characteristics are properly set.

?DTE-F-MSGFORM, Message format error (xxxxxx)

Explanation: The version of the MFT file server within the DTE task is not compatible with the version of the MFT task on the remote system. This may be caused by an error in the remote MFT task or by an incompatibility of releases of DTE and MFT.

User Action: No user action is required.

?DTE-F-NOTSELD, Terminal line not selected

Explanation: This is an internal error and indicates that the Logical Unit Number (LUN) has been lost.

User Action: Reenter the command. If the error persists, submit an SPR.

%DTE-S-PHONEOK, Telephone connection established

Explanation: DTE sent the dial sequence specified by the /DIAL option, and the modem responded with success (it returned the character A).

User Action: No user action is required. The connection to the remote system has been successfully established.

?DTE-F-PHONERR, Failure to establish telephone connection

Explanation: DTE sent the dial sequence specified by the /DIAL option, and the modem did not respond with success.

User Action: No user action is required. DTE will hang up the line and then exit with error status.

?DTE-F-PORTSEL, Assignment failure on specified device

Explanation: DTE was unable to assign and attach the specified emulation terminal device. The device may be attached by another user, or it may be an inappropriate device type for emulation.

User Action: Wait for the device to be detached or reenter the command with an appropriate device type specified.

?DTE-F-REJOPER, Operation rejected (xxxxxx)

Explanation: The file operations server within the DTE task cannot perform the operation requested by the MFT task executing on the remote system. Either the DTE task lacks the required support, or the version of the remote MFT task does not match the version of DTE being used on the local system.

User Action: No user action is required.

?DTE-F-RMSRERR, Remote RMS–11 error (xxxxxx)

Explanation: An error occurred during the attempt to perform the file operation. The RMS–11 error code (in octal) is displayed within the paired parentheses.

User Action: No user action is required.

?DTE-F-SYNLOST, Synchronization lost (xxxxxx)

Explanation: Synchronization has been lost between the local DTE and the remote MFT tasks. This may be related to the baud rate of the emulation terminal lines or the systems' loads.

User Action: Retry the operation.

?DTE-F-SYNXERR, Syntax error

Explanation: The syntax of the DTE command either was not appropriate or did not fit the documented syntax.

User Action: Enter the correct command.

?DTE-F-UNSPROT, Unsupported protocol operation (xxxxxx)

Explanation: The version of DDCMP being used by the local DTE file operations server and the remote MFT task are not compatible. The versions of DTE and MFT are not compatible, and the requested operation cannot be performed.

User Action: No user action is required.

C.6.2 MFT Messages

The following messages are produced by MFT. Some messages also include a "..." field. This indicates that all or part of the command line will be displayed with the message.

?MFT-F-BADOPER, Illegal option specified: ...

Explanation: A nonexistent or illegal option was specified on the command line, or the option or options selected are out of context.

User Action: No user action is required.

?MFT-F-COMFILE, Unable to open specified command file

Explanation: MFT was unable to access the specified command file. No details as to the reason for the failure are available. Typical reasons for this message are: file not found, protection violations, and so on.

User Action: No user action is required.

?MFT-F-FOPFAIL, File operation failed

Explanation: The particular file operation specified could not be performed. There are many possible causes for the failure. Typically, this message is produced when MFT is unable to synchronize with the file operations server it requests DTE to invoke. Reasons for this type of error are as follows:

- Baud rates are too high.
- The systems are too far apart (the physical cable is too long).
- The system lacks the required terminal support.

- There is some electrical interference (noise) along the physical connection (because of a bad telephone connection or heavy electrical equipment).

User Action: No user action is required.

?MFT-F-ILLNODE, Illegal node combination: ...

Explanation: The MFT command string specifies both file specifications on the same system. Either /REM appears on both file specifications or does not appear on either specification. Each file specification must specify a different system.

User Action: Specify the /REM option with one file specification only and reenter the command line.

?MFT-F-LINELEN, Command line too long

Explanation: The command line specified was too long for the command buffer. Because some of the command may have been lost, the entire command is considered invalid.

User Action: Enter a shorter command line.

?MFT-F-SPECERR, Error in file specification: ...

Explanation: One or both of the file specifications supplied are in error. File specification syntax is limited to the standard RSX–11 format, which is as follows:

```
device:[uic]filename.type;version
```

User Action: Use valid file specifications and reenter the command line.

?MFT-F-SYNXERR, Syntax error: ...

Explanation: The syntax of the command line specified is in error. Possible causes may be illegal options, missing input or output file specifications, or an error in the file specification syntax.

User Action: Use the correct syntax and reenter the command line.

Index

A

B

C

D

E

F

H

I

K

L

M

R

S

T

U

V

W

X

Y

Z

RSX-11M-PLUS
Utilities Manual
AA-JS15A-TC

READER'S COMMENTS

Your comments and suggestions are welcome and will help us in our continuous effort to improve the quality and usefulness of our documentation and software.

Remember, the system includes information that you read on your terminal: help files, error messages, prompts, and so on. Please let us know if you have comments about this information, too.

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.

What kind of user are you? ____ Programmer ____ Nonprogrammer

Years of experience as a computer programmer/user: _____

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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------