


pdp11



RSX-11M
Utilities Procedures Manual

Order No. DEC-11-OMUPA-B-D

digital

RSX-11M
Utilities Procedures Manual

Order No. DEC-11-OMUPA-B-D

RSX-11M Version 2

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

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

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright © 1974, 1975 by Digital Equipment Corporation

The HOW TO OBTAIN SOFTWARE INFORMATION pages, located at the back of this document, explain the various services available to Digital software users.

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

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECTape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM		

LIMITED RIGHTS LEGEND

Contract No. _____

Contractor or Subcontractor: Digital Equipment Corporation

All the material contained herein is considered limited rights data under such contract.

CONTENTS

		Page
PREFACE		xi
0.1	MANUAL OBJECTIVES AND READER ASSUMPTIONS	xi
0.2	STRUCTURE OF THE DOCUMENT	xi
0.3	ASSOCIATED DOCUMENTS	xiii
CHAPTER 1	INTRODUCTION	1-1
1.1	RSX-11M UTILITY PROGRAMS	1-1
1.2	INITIATING RSX-11M UTILITIES	1-2
1.2.1	Initiating of Installed Utilities	1-2
1.2.2	Initiation of Uninstalled Utilities	1-3
1.3	RSX-11M UTILITY COMMAND STRINGS	1-4
1.3.1	Command Format Conventions	1-6
1.4	INDIRECT FILES	1-7
1.5	SYSTEM-WIDE CONVENTIONS	1-8
CHAPTER 2	PERIPHERAL INTERCHANGE PROGRAM (PIP)	2-1
2.1	INTRODUCTION TO PIP	2-1
2.2	INITIATING PIP	2-1
2.3	PIP COMMAND STRING	2-1
2.3.1	Lists of File Specifiers	2-2
2.3.2	Defaults in File Specifiers	2-2
2.3.3	PIP Command Switches and Subswitches	2-3
2.3.4	Asterisk Convention - Wild Cards	2-4
2.3.4.1	Wild Cards in Output File Specifiers	2-4
2.3.4.2	Wild Cards in Input Specifiers	2-5
2.3.5	File Identification Option	2-5
2.4	PIP COMMANDS	2-6
2.4.1	APPEND Command (/AP)	2-7
2.4.2	COPY Command (No Switch) and MERGE Command (/ME)	2-8
2.4.3	DEFAULT Command (/DF)	2-12
2.4.4	DELETE Command (/DE)	2-13
2.4.5	ENTER Command (/EN)	2-15
2.4.6	FREE Command (/FR)	2-16
2.4.7	IDENTIFY Command (/ID)	2-16
2.4.8	LIST Command (/LI)	2-17
2.4.9	PROTECT Command (/PR)	2-21
2.4.10	PURGE Command (/PU[:n])	2-24
2.4.11	REMOVE Command (/RM)	2-25
2.4.12	RENAME Command (/RE)	2-26
2.4.13	SPOOL Command (/SP)	2-28
2.4.14	UNLOCK Command (/UN)	2-29
2.4.15	UPDATE Command (/UP)	2-30
2.5	PIP ERROR MESSAGES	2-31
2.6	PIP ERROR CODES	2-41

CONTENTS (Cont.)

		Page
CHAPTER 3	FILE TRANSFER PROGRAM (FLX)	3-1
3.1	INTRODUCTION TO FLX	3-1
3.2	INITIATING FLX	3-2
3.3	FLX COMMAND STRING	3-2
3.4	FILE TRANSFERS	3-2
3.5	DOS VOLUME DIRECTORY MANIPULATION	3-3
3.5.1	DOS Directory Listings	3-3
3.5.2	Deleting DOS Files	3-3
3.5.3	Initializing DOS-11 Volumes	3-3
3.6	RT VOLUME DIRECTORY MANIPULATION	3-3
3.6.1	RT Directory Listings	3-3
3.6.2	Deleting RT Files	3-4
3.6.3	Initializing RT-11 Volumes	3-4
3.7	FLX CASSETTE SUPPORT	3-5
3.7.1	Cassette File Formats	3-5
3.7.2	Multi-Volume Cassette Support	3-9
3.7.2.1	FLX Output Files	3-9
3.7.2.2	FLX Input File	3-10
3.8	FLX PAPER TAPE SUPPORT	3-10
3.9	FLX SWITCHES	3-11
3.9.1	Format Mode Switches	3-11
3.9.2	Transfer Mode Switches	3-13
3.9.3	File Control Switches	3-15
3.10	FLX ERROR MESSAGES	3-20
CHAPTER 4	FILE DUMP UTILITY (DMP)	4-1
4.1	INTRODUCTION TO DMP	4-1
4.2	INITIATING DMP	4-2
4.3	DMP COMMAND STRINGS	4-2
4.4	DMP SWITCHES	4-2
4.5	DMP ERROR MESSAGES	4-5
CHAPTER 5	LINE TEXT EDITOR (EDI)	5-1
5.1	INTRODUCTION	5-1
5.2	USING EDI	5-1
5.2.1	Preparing to Run EDI	5-1
5.2.2	Initiating EDI	5-2
5.2.2.1	Defaults in File Specifiers	5-3
5.2.3	EDI Control Modes	5-4
5.2.4	Changing Control Mode	5-4
5.2.5	Text Access Modes	5-5
5.2.5.1	Line-by-Line Mode	5-5
5.2.5.2	Block Mode	5-5
5.2.5.3	Line-by-Line Vs. Block Mode	5-6
5.2.6	Text Files	5-6
5.2.6.1	Input and Secondary Files	5-7
5.2.6.2	Output Files	5-7
5.2.7	Terminal Conventions	5-7
5.2.7.1	Carriage Return	5-7
5.2.7.2	Character Erase (RUBOUT) and Line Delete (CTRL U)	5-7

CONTENTS (Cont.)

	Page	
5.2.8	EDI Command Conventions	5-8
5.2.8.1	Use of *	5-8
5.2.8.2	Search String Constants	5-8
5.3	EDI ERROR REPORTING	5-9
5.3.1	Command Level Informational and Error Messages	5-9
5.3.2	File Access Warning Messages	5-9
5.3.3	Error Messages Requiring EDI Restart	5-9
5.3.4	Fatal Error Messages	5-10
5.4	BASIC EDI OPERATION AND COMMANDS	5-10
5.4.1	Basic EDI Operations	5-10
5.4.1.1	Creating a File	5-10
5.4.1.2	Entering Text Into a File	5-11
5.4.2	Editing a File	5-11
5.4.3	Basic EDI Commands	5-13
5.4.3.1	ADD Command	5-15
5.4.3.2	ADD AND PRINT Command	5-15
5.4.3.3	CHANGE Command	5-15
5.4.3.4	CTRL/Z Command	5-16
5.4.3.5	DELETE Command	5-16
5.4.3.6	DELETE AND PRINT Command	5-17
5.4.3.7	EXIT Command	5-18
5.4.3.8	INSERT Command	5-18
5.4.3.9	LOCATE Command	5-19
5.4.3.10	NEXT Command	5-19
5.4.3.11	NEXT PRINT Command	5-20
5.4.3.12	PRINT Command	5-20
5.4.3.13	RENEW Command	5-22
5.4.3.14	RETYPE Command	5-22
5.4.3.15	TOP OF FILE Command	5-22
5.4.4	Sample Editing Session	5-23
5.5	EXTENDED EDI COMMANDS	5-23
5.5.1	Setup Commands	5-23
5.5.1.1	BLOCK ON/OFF Command	5-25
5.5.1.2	CONCATENATION CHARACTER Command	5-25
5.5.1.3	OPENS Command	5-26
5.5.1.4	OUTPUT ON/OFF Command	5-26
5.5.1.5	SELECT PRIMARY Command	5-27
5.5.1.6	SELECT SECONDARY Command	5-28
5.5.1.7	SIZE Command	5-29
5.5.1.8	TAB ON/OFF Command	5-29
5.5.1.9	UPPER CASE ON/OFF Command	5-30
5.5.1.10	VERIFY ON/OFF Command	5-31
5.5.2	EDI Input/Output Commands	5-31
5.5.2.1	FILE Command	5-32
5.5.2.2	READ Command	5-32
5.5.2.3	WRITE Command	5-33
5.5.3	Line Pointer Control (Locative) Commands	5-34
5.5.3.1	BEGIN Command	5-36
5.5.3.2	BOTTOM Command	5-36
5.5.3.3	END Command	5-36
5.5.3.4	FIND Command	5-37
5.5.3.5	OLDPAGE Command	5-37
5.5.3.6	PAGE Command	5-38
5.5.3.7	PAGE FIND Command	5-38
5.5.3.8	PAGE LOCATE Command	5-39
5.5.3.9	SEARCH AND CHANGE Command	5-39

CONTENTS (Cont.)

		Page
5.5.3.10	TOP Command	5-40
5.5.4	Text Modification and Manipulation Commands	5-40
5.5.4.1	ERASE Command	5-42
5.5.4.2	FORM FEED Command	5-43
5.5.4.3	LINE CHANGE Command	5-43
5.5.4.4	LIST ON TERMINAL Command	5-43
5.5.4.5	LIST ON PSEUDO-DEVICE Command	5-44
5.5.4.6	MACRO Command	5-44
5.5.4.7	MACRO CALL Command	5-45
5.5.4.8	MACRO EXECUTE Command	5-46
5.5.4.9	MACRO (IMMEDIATE) Command	5-47
5.5.4.10	OVERLAY Command	5-47
5.5.4.11	PASTE Command	5-48
5.5.4.12	SAVE Command	5-48
5.5.4.13	TYPE Command	5-49
5.5.4.14	UNSAVE Command	5-50
5.5.5	EDI Close Operation Commands	5-50
5.5.5.1	CLOSE Command	5-51
5.5.5.2	CLOSES Command	5-51
5.5.5.3	CLOSE AND DELETE Command	5-51
5.5.5.4	EXIT AND DELETE Command	5-52
5.5.5.5	KILL Command	5-52
5.6	EDI ERROR MESSAGES	5-52
5.6.1	Command Level Informational and Error Messages	5-52
5.6.2	File Access Warning Messages	5-58
5.6.3	Error Messages Requiring EDI Restart	5-59
5.6.4	Fatal Error Messages	5-62
CHAPTER 6	SOURCE LANGUAGE INPUT PROGRAM (SLP)	6-1
6.1	INTRODUCTION TO SLP	6-1
6.2	PREPARING TO RUN SLP	6-1
6.2.1	Capabilities	6-1
6.2.2	Environment	6-1
6.2.3	Restrictions	6-2
6.3	INITIATING SLP	6-2
6.4	SLP STARTUP	6-2
6.4.1	Defaults in File Specifiers	6-3
6.4.2	Examples of SLP Initialization	6-3
6.5	SLP OUTPUT CONTROL SWITCHES	6-4
6.6	SLP OUTPUT FILES	6-4
6.7	SLP EDIT COMMANDS	6-6
6.7.1	SLP Edit Control Characters	6-6
6.8	INDIRECT FILES	6-8
6.8.1	Creating an Indirect File	6-8
6.8.2	Using Indirect Files	6-9
6.9	SLP EDITING EXAMPLES	6-9
6.10	SLP ERROR MESSAGES	6-12
CHAPTER 7	LIBRARIAN UTILITY PROGRAM (LBR)	7-1
7.1	INTRODUCTION TO LBR	7-1
7.1.1	Format of Library Files	7-1
7.1.2	Library Header	7-2
7.1.3	Entry Point Table	7-2
7.1.4	Module Name Table	7-4
7.1.5	Module Header	7-4

CONTENTS (Cont.)

		Page
7.2	INITIATING LBR	7-5
7.3	LBR COMMAND STRING	7-5
7.4	DEFAULTS IN LBR FILE SPECIFIERS	7-5
7.5	LBR FILE OPTION SWITCHES	7-7
7.5.1	Compress Switch (/CO)	7-7
7.5.2	Create Switch (/CR)	7-9
7.5.3	Delete Switch (/DE)	7-11
7.5.4	Default Switch (/DF)	7-12
7.5.5	Delete Global Switch (/DG)	7-13
7.5.6	Insert Switch (/IN)	7-14
7.5.7	List Switches (/LI, /LE, /FU)	7-15
7.5.8	Replace Switch (/RP)	7-17
7.5.9	Spool Switch (/SP)	7-23
7.5.10	Selective Search Switch (/SS)	7-23
7.5.11	Squeeze Switch (/SZ)	7-24
7.6	COMBINING LIBRARY FUNCTIONS	7-26
7.7	LBR CONSTRAINTS	7-27
7.8	LBR ERROR MESSAGES	7-27
7.8.1	Effect of Fatal Errors on Library Files	7-28
7.8.2	Error Messages	7-28
CHAPTER 8	FILE STRUCTURE VERIFICATION UTILITY (VFY)	8-1
8.1	INTRODUCTION TO VFY	8-1
8.2	INITIATING VFY	8-2
8.3	VFY COMMAND STRING	8-2
8.3.1	Defaults in File Specifiers	8-3
8.4	VFY COMMAND SWITCHES	8-4
8.4.1	Validity Check	8-5
8.4.1.1	File Error Reporting	8-6
8.4.1.2	Files Marked-for-Delete	8-7
8.4.1.3	Deletion of Multiply Allocated Blocks	8-8
8.4.1.4	Elimination of Free Blocks	8-8
8.4.1.5	Recovering Lost Blocks	8-9
8.4.2	DELETE Switch (/DE)	8-9
8.4.3	UPDATE Switch (/UP)	8-10
8.4.4	REBUILD Switch (/RE)	8-11
8.4.5	FREE Switch (/FR)	8-12
8.4.6	LOST Switch (/LO)	8-12
8.4.7	LIST Switch (/LI)	8-12
8.4.8	READ CHECK Switch (/RC)	8-13
8.5	VFY ERROR MESSAGES	9-14
8.6	VFY ERROR CODES	8-16
APPENDIX A	COMMANDS AND SWITCHES	A-1
A.1	INTRODUCTION	A-1
A.2	PIP COMMAND SUMMARY	A-1
A.3	FLX COMMAND SUMMARY	A-3
A.4	DMP COMMAND SUMMARY	A-5
A.5	EDI COMMAND SUMMARY	A-5
A.6	SLP COMMAND SUMMARY	A-12
A.7	LIBR COMMAND SUMMARY	A-13
A.8	VFY COMMAND SUMMARY	A-14

CONTENTS (Cont.)

		Page	
APPENDIX B	LBR, EDI AND DMP EXAMPLES	B-1	
B.1	SAMPLE LISTINGS FOR LBR LIST SWITCHES (OBJECT LIBRARY)	B-1	
B.1.1	List Module Names	B-1	
B.1.2	List Module Names and Full Module Information	B-2	
B.1.3	List Module Names, Full Module Information and Module Entry Points (Global Symbols)	B-3	
B.1.4	List Module Names and Module Entry Points (Global Symbols)	B-7	
B.2	SAMPLE LISTING FOR LBR LIST SWITCHES (MACRO LIBRARY)	B-13	
B.2.1	List Module Names	B-13	
B.2.2	List Module Names and Full Module Information	B-14	
B.3	SAMPLE EDITING OPERATIONS	B-15	
B.3.1	File Editing Sample	B-16	
B.3.2	SAVE and UNSAVE Example	B-20	
B.3.3	Use of Immediate Macro Command	B-23	
B.3.4	Use of Macro Commands	B-24	
B.4	SAMPLE DMP LISTINGS	B-26	
B.4.1	Use of /LB Switch	B-26	
B.4.2	"Standard" Command Line	B-26	
B.4.3	Dump Only the Header from SYSGEN.CMD	B-28	
B.4.4	Use of /BA Switch	B-28	
APPENDIX C	RSX-11M PRINT SPOOLER TASK	C-1	
C.1	RECEIVE QUEUE OPERATION	C-1	
C.2	TEXT REQUIREMENTS	C-1	
C.3	TASK BUILD INFORMATION	C-2	
C.4	PRT ERROR MESSAGES	C-3	
INDEX		Index-1	
FIGURES			
		Page	
Figure	2-1	Results of COPY Command With and Without /NV Specified	2-12
	2-2	Sample Directories Before and After Execution of /EN Command	2-16
	2-3	Directory Listing Examples	2-20
	2-4	Format of Protection Word	2-23
	2-5	Use of Purge Switch	2-25
	2-6	Results of Rename Command With and Without /NV Specified	2-28
	3-1	DEC Standard Cassette File Structure	3-6
	3-2	DEC Standard Cassette File Label	3-7
	3-3	DOS Directory Listings	3-18
	3-4	RT Directory Listing	3-19
	7-1	General Library File Format	7-2
	7-2	Contents of Library Header	7-3
	7-3	Format of Entry Point Table Element	7-4

CONTENTS (Cont.)

FIGURES (Cont.)

		Page
7-4	Format of Module Name Table Element	7-4
7-5	Module Header Format	7-5
7-6	Sample Files Used in LBR Examples	7-20
7-7	Output Library File After Execution of Example 1	7-21
7-8	Output Library File After Execution of Example 2	7-21
7-9	Output Library File After Execution of Example 3	7-22
7-10	MACRO Listing Before and After Running LBR with /SZ Switch	7-26
8-1	VFY Listing Sample Using the /LI Switch	8-13
C-1	PRT Send Data Buffer Format	C-2

TABLES

		Page
Table 2-1	PIP Default File Specifiers	2-2
2-2	PIP Functions and Commands	2-6
2-3	PIP COPY and MERGE Command Subswitches	2-10
2-4	LIST Command Switches	2-18
2-5	PIP Error Codes	2-41
3-1	FLX Format Mode Switches	3-12
3-2	FLX Transfer Mode Switches	3-13
3-3	FLX File Control Switches	3-15
4-1	DMP Switches	4-3
5-1	EDI Default File Specifiers	5-4
5-2	Line-by-Line vs. Block Mode	5-6
5-3	Basic EDI Commands	5-14
5-4	EDI Setup Commands	5-24
5-5	EDI Input/Output Commands	5-32
5-6	EDI Locative Commands	5-35
5-7	EDI Text Modification and Manipulation Commands	5-41
5-8	EDI Close Operation Commands	5-50
6-1	Defaults in SLP File Specifiers	6-3
6-2	SLP Output Control Switches	6-5
6-3	SLP Edit Control Characters	6-7
7-1	Defaults in LBR File Specifiers	7-6
7-2	LBR File Option Switches	7-7
8-1	VFY Default File Specifiers	8-4
8-2	VFY Functions and Switches	8-4
8-3	VFY Error Codes	8-17

PREFACE

0.1 MANUAL OBJECTIVES AND READER ASSUMPTIONS

The goal of this manual is to describe the utility programs supplied with the RSX-11M and to provide all the information necessary to use these programs. Within the RSX-11M utility program set, programs are provided to perform the following functions:

- . File transfer
- . File conversion
- . File listing
- . Interactive context editing
- . Batch-oriented editing
- . Library maintenance
- . File structure verification on Files-11 disk volumes.

It is assumed the reader is familiar with the PDP-11 computer, its peripheral devices and the software supplied with the RSX-11M system. Users of this manual should be also familiar with the RSX-11M Operator's Procedure Manual which provides supporting information necessary for the full utilization of the RSX-11M utilities.

This manual is organized and written as a reference manual and our reader class assumptions require a system programmer level of expertise; thus, the manual will not contain definitions of data processing terms and concepts familiar to this level of expertise.

0.2 STRUCTURE OF THE DOCUMENT

This document contains an introductory chapter which provides general operating information and a separate chapter for each utility program, containing specific operating information.

Chapter 1 describes the procedures required for initiating the utility programs, defines command formats and command strings, and identifies system-wide conventions.

Chapter 2 details the Peripheral Interchange Utility Program (PIP), a file transfer program.

Chapter 3 details the File Transfer Utility program (FLX), a file conversion program.

Chapter 4 describes the File Dump Utility program (DMP), a file listing program.

Chapter 5 contains detailed information on the Line Text Editor Utility program (EDI), an interactive context editing program.

Chapter 6 details the Source Language Input Utility program (SLP), a batch-oriented editing program.

Chapter 7 describes the Librarian Utility Program, a library maintenance program.

Chapter 8 details the File Structure Verification Utility program (VFY), a file structure verification program for Files-11 disk volumes.

Appendix A contains a command summary for each utility.

Appendix B contains sample LBR and DMP listings, and EDI examples.

Appendix C contains a description of the print spooler task.

Throughout this manual the following conventions are used to describe examples and command string formats:

- a. ↵ indicates a carriage return
- b. Ⓢ indicates ALTMode or ESCape key
- c. In examples, messages which are typed out by the system, are underlined.
- d. In all cases, except where [UIC] is specified, brackets signify optional parameters.
- e. Δ indicates a space.

Additional conventions and definitions are given in Chapter 1.

0.3 ASSOCIATED DOCUMENTS

Other manuals closely allied to the purposes of this document are described briefly in the RSX-11M/RSX-11S Documentation Directory, Order No. DEC-11-OMUGA-B-D. The Documentation Directory defines the intended readership of each manual in the RSX-11M/RSX-11S set and provides a brief synopsis of each manual's contents.

CHAPTER 1

INTRODUCTION

1.1 RSX-11M UTILITY PROGRAMS

RSX-11M provides the user with a comprehensive set of utility programs. The utility programs and their identifiers are as follows:

- Peripheral Interchange Utility Program (PIP)
PIP is a file transfer program that provides the user with facilities for copying, renaming, listing, deleting, and unlocking files.
- File Transfer Utility Program (FLX)
FLX is a file conversion program that provides the user with a facility for transferring DOS-11 or RT-11 files to Files-11 volumes and vice versa.
- File Dump Utility Program (DMP)
DMP is a file listing program that provides the user with a facility for obtaining a printed copy of the contents of files.
- Line Text Editor Utility Program (EDI)
EDI is an interactive context editing program that provides the user with a facility for creating and maintaining text files.
- Source Language Input Utility Program (SLP)
SLP is a batch-oriented editing program that provides the user with a facility for creating and maintaining text files on disk.
- Librarian Utility Program (LBR)
LBR is a library maintenance program that provides the user with a facility for creating, modifying, updating, listing, and maintaining library files.
- File Structure Verification Utility Program (VFY)
VFY is a disk verification program that provides the user with a facility for verifying the consistency and validity of the file structure on a Files-11 volume.

INTRODUCTION

1.2 INITIATING RSX-11M UTILITIES

There are six methods for initiating RSX-11M utility programs. These methods are described below. The first four methods apply when the utility program is installed and ready to be executed. The remaining two cause the utility program to be installed, executed, and then removed on exit.

NOTE

The RSX-11M systems provided in the distribution kit require that all utilities be initiated via methods five and six. However, using the INSTALL MCR Command (see RSX-11M Operator's Procedures Manual), the user can permanently install the utility programs in his system and thus make the first four methods of initiation available.

1.2.1 Initiation of Installed Utilities

Method 1. >utilityname command string ↵
loads, executes the specified command(s), and exits.

Method 2. >utilityname ↵
responds with the following prompt.

utilityname>

At this point, the user enters the utility command string to execute the desired function.

When the utility has completed processing a command string it again issues a prompt. The user can either enter another command string or enter a CTRL/Z* to terminate the utility.

Method 3. >RUN ...utilityname ↵
responds with the following prompt:

utilityname>

At this point, the user enters the utility command string to execute the desired function.

* CTRL/Z is entered by holding down the CTRL key while simultaneously depressing the Z key.

INTRODUCTION

When the utility has completed processing a command string, it again issues a prompt. The user can either enter another command string or enter CTRL/Z to terminate the utility.

Method 4. `>RUN ...utilityname/UIC=[group,member]`↵

The UIC under which the utility executes is explicitly specified for this run only. Normally, utility programs execute with the default UIC associated with the initiating terminal (see SET /UIC MCR command described in RSX-11M Operator's Procedures Manual). When the utility is loaded, it issues the following prompt:

utilityname>

At this point, the user enters the utility command string to execute the desired function.

When the utility has completed processing a command string, it again issues a prompt. The user can either enter another command string or enter CTRL/Z to terminate the utility.

NOTE

Methods 3 and 4 should not be used on multi-user RSX-11M systems.

1.2.2 Initiation of Uninstalled Utilities

Method 5. `>RUN $utilityname`↵
causes the utility to be installed and loaded, and to issue the following prompt:

utilityname>

At this point, the user enters the utility command string to execute the desired function.

When the utility has completed processing a command string, it again issues a prompt. The user either enters another command string to execute the desired function or enters CTRL/Z to cause the utility to exit. After exiting, the utility is removed from the system.

Method 6. `>RUN $utilityname/UIC=[group,member]`↵

INTRODUCTION

The UIC under which the utility executes is explicitly specified for this run only. Normally, utility programs execute with the default UIC associated with the initiating terminal (See SET /UIC MCR command described in the RSX-11M Operator's Procedures Manual). When the utility is installed and loaded, it issues the following prompt:

utilityname>

At this point, the user enters the utility command string to execute the desired function.

When the utility has completed processing a command string, it again issues a prompt. The user either enters another command string to execute the desired function or enters CTRL/Z to cause the utility to exit. After exiting, the utility is removed from the system.

1.3 RSX-11M UTILITY COMMAND STRINGS

Commands to RSX-11M utilities are expressed in the following format:

outflespc-1,...,outflespc-n=inflespc-1,...,inflespc-n

or

@indirect

where:

outflespc is an output file specifier.

inflespc is an input file specifier.

Any number of file specifiers is possible, the actual number being determined by the task which will use the file command string. In no case, however, can the total length of the command string exceed the maximum line length (80 characters).

Each file specifier (whether input or output) has the following format:

dev:[g,m]filename.typ;ver/sw.../sw

where:

dev: is the physical device on which the volume containing the desired file is mounted; for example, DK0: or DT1:. The name consists of 2 ASCII characters

INTRODUCTION

followed by an optional 1- or 2-digit (octal) unit number and a colon.

[g,m] is the user identification code (UIC), consisting of a group number and a member number, associated with the user file directory (UFD) containing the desired file.

filename is the name of the file. In RSX-11M, a filename can be up to nine alphanumeric characters in length. Filename and type are always separated by a period (.).

typ is a means of distinguishing among forms of one file. For example, a source FORTRAN program might be named COMP.FTN, while the object code associated with that program might be called COMP.OBJ. File type and version always are separated by a semicolon (;). File type may be up to 3 alphanumeric characters.

ver is an octal number used to differentiate among versions of a file. For example, if a file is first created using the editor, it is assigned a version number of 1. If the file is subsequently opened for editing, the editor keeps the original file for backup and creates a new file with the same filename and type, but with a version number of 2. Version is in the range 0 thru 7777(8).

/sw is a 2-character ASCII name identifying a switch option. The switch itself may have three forms. If the switch designator, for example, is SW, then:

```
    /SW      sets the switch
             action;
    /-SW,    negates      the
             switch  action,
             and
    /NOSW    also      negates
             the      switch
             action.
```

In addition, the switch identifier may be followed by any number of values. The permitted values are ASCII strings, octal numbers, and decimal numbers.

INTRODUCTION

The default for a numerical value is octal. Decimal values are terminated by a decimal point. Values preceded by a pound sign (#) are octal; the octal option is included for use as explicit documentation, since a numeric value not terminated with a decimal point is an octal value. Finally, any numeric value may be preceded by a + or - sign; plus is the default.

If explicit octal (#) is used, the sign (if the sign is used) must precede the # (pound) sign. The following are valid switch specifications:

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

The number of permissible values and the switch interpretations themselves depend entirely on the particular task to which they are directed.

@indirect is an indirect command file specifier in the following format:

```
@dev:[g,m]filename.typ;ver
```

1.3.1 Command Format Conventions

Throughout this manual, the following conventions are used in presenting command formats.

[] Entries within square brackets indicate optional items. The use of such items may be for readability, e.g., BL[OCK]; or to denote repetition, e.g., [n].

EXCEPTION

[g,m] The square brackets are REQUIRED when specifying a User's Identification Code.

Δ One or more spaces.

_____ Underlined characters denote those displayed by the system.

ABC Characters in upper case indicate constants, i.e., items the user must enter exactly as presented.

filespec A file specifier, i.e., dev:[g,m]filename.type;ver

INTRODUCTION

dev	A device specifier, i.e., device name and unit number, such as DK1:.
type	The file type, such as .CMD, .OBJ, or .OLB.
ver	The version number of the file.
[g,m]	The User's Identification Code composed of two octal numbers separated by a comma and surrounded by square brackets. The left-hand number is the user's group number; the right-hand number is the user's member number.
[uic]	Also used to indicate the User's Identification Code; [uic] and [g,m] are used interchangeably in the RSX-11M documentation.
@	Indicates an indirect file specifier follows.
/	Indicates that a command switch follows. If a minus sign (-) or the letters NO are placed between the slash (/) and the switch name, it denotes that the switch action is negated.

1.4 INDIRECT FILES

An indirect file is a sequential file containing a list of commands exclusive to, and interpretable by a single task, usually a system-supplied component of RSX-11M, such as MACRO-11, the Task Builder or a utility program.

Indirect files are initiated by replacing the file specification command string required by a task with a filename string preceded by an at sign (@).

For example, to initiate a file of MACRO-11 commands, the user would input:

```
>MAC @INPT.CMD
```

After MACRO-11 is initiated, it accesses the file INPT.CMD for all its commands.

An indirect file may contain any command interpretable by the task to which it is directed, but no others.

Indirect files may not contain indirect file references (i.e., only one level of command file indirection is permitted).

A complete description of indirect files for use with MCR is contained in the RSX-11M Operator's Procedures Manual.

INTRODUCTION

NOTES

1. The default file type for indirect command files is .CMD.
2. Other default values for indirect command file specifiers are:

dev = SY0:
[uic] = The UIC under which the specified utility is running.
filename = No default.
type = .CMD
ver = Latest version

1.5 SYSTEM-WIDE CONVENTIONS

There are a number of system-wide conventions of which the user should be aware.

1. The use of NO is the equivalent of a minus sign (-) in specifying the negation of a switch, e.g.,
/NOSP is equivalent to /-SP
2. The alphanumeric file names and types are composed of the letters A through Z and the numbers 0 through 9.
3. All numbers followed by a period (.) denote decimal numbers; others are interpreted as octal numbers.

CHAPTER 2
PERIPHERAL INTERCHANGE PROGRAM (PIP)

2.1 INTRODUCTION TO PIP

The Peripheral Interchange Program (PIP) is an RSX-11M file utility program that transfers data files from one standard Files-11 device to another. PIP also performs simple control functions. The major functions performed by PIP are:

- Copy files from one device to another
- Delete files
- Rename files
- List file directories
- Set the default device and UIC for PIP
- Unlock files

2.2 INITIATING PIP

All RSX-11M utility programs can be initiated in several ways. These methods are described in Section 1.2. The methods for PIP are:

```
>PIP↵  
>PIP command string↵  
>RUN ...PIP↵  
>RUN ...PIP/UIC=[group,member]↵  
>RUN $PIP↵  
>RUN $PIP/UIC=[group,member]↵
```

2.3 PIP COMMAND STRING

All commands to PIP are issued by entering PIP command strings through the initiating terminal. The format of the elements which comprise PIP command strings differs for each command. Therefore, the command string formats will be individually described in their respective sections.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

2.3.1 Lists of File Specifiers

All of the commands, except /IDENTIFY and /DEFAULT, accept a list of file specifiers on which to operate. In all cases, the lists have the property that the device, directory, filename, and type are propagated down the list to provide defaults for missing fields in subsequent file specifiers.

2.3.2 Defaults in File Specifiers

If any of the elements in the file specifier, except the filename and type, are omitted, PIP uses a default. These default values are listed in Table 2-1.

Table 2-1
PIP Default File Specifiers

Element	Default Value
dev:	SY0: -- For first file specifier, the unit on which the system disk is mounted, or the default specified by the PIP /DF switch; otherwise, the same as specified or assumed for previous file specifier.
[uic]	For first file specifier, the default UIC under which PIP is running (usually [200,200]), the UIC specified by the MCR SET command, or the default specified by the PIP /DF switch; otherwise, the same as specified or assumed for previous file specifier.
filename	No default for the first file specifier. For the second through n file specifiers, the last previously specified filename. An asterisk (wild card) specification is valid (see Section 2.3.4).
.typ	No default for the first file specifier. For the second through n file specifiers, the last previously specified .typ. An asterisk (wild card) specification is valid (see Section 2.3.4).
;version	The default for input files is the most recent version number. The default for output files is the next higher version number, or, version one if the file doesn't already exist in the output directory. An exception is the PIP file delete function that requires an explicit version number.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

NOTE

A version number of ;-1 may be used to specify the oldest version of a file. An explicit version of ;0 or ; may be specified to signify the most recent version.

2.3.3 PIP Command Switches and Subswitches

A switch specification consists of a slash (/) followed by a 2-character switch name, and is optionally followed by a subswitch name separated from the switch code by a slash. The subswitch itself can have arguments which are separated from the subswitch by a colon (:). If more than one subswitch is used, each is preceded by a slash.

When present, a switch must follow any file or UIC specification; that is, a switch cannot appear before the filename, type, version, or UIC of the file on which the switch is to operate. However, some switches may be specified without any file specification at all.

Command switches are global, that is, they may be specified once for an entire list of file specifiers, and may appear on either side of the equal sign. For example:

```
string1,string2,string3/DE
```

The /DE switch applies to all of the strings; the files described by the file specifiers would be deleted.

Switch arguments are specified as octal, decimal, or alphabetic characters, depending on the switch. These values are discussed in detail in the sections which discuss the individual PIP commands.

Command subswitches are local, that is, they only apply to the file specifier which immediately precedes them. In the following example, the NEW VERSION subswitch is applied to a particular file (ASDG.MAC). (The NEW VERSION subswitch is used with the RENAME command in this example):

```
*.SMP=PRTX.QRT,ASDG.MAC/NV,KG.BAC/RE
```

In this example, files PRTX.QRT and KG.BAC are renamed, but they maintain their associated version numbers. File ASDG.MAC is also renamed, but the version number is forced to one greater than the latest version of file ASDG.SMP.

NOTE

If a subswitch is applied to the first file specifier in a collection of file specifiers, and no command switch has been specified, PIP assumes that the command with which the subswitch is associated is the one requested, and the entire list of files is treated as though the command were actually

PERIPHERAL INTERCHANGE PROGRAM (PIP)

specified.

Example:

```
PIP>FILE1/GR:R/WO,FILE2/GR:RW ↵
```

This command is equivalent to:

```
PIP>FILE1/GR:R/WO,FILE2/GR:RW/PR ↵
```

This example would result in the following file protection:

- | | | | |
|----------|--------|---|-------------------|
| a. FILE1 | SYSTEM | - | Unchanged |
| | MEMBER | - | Unchanged |
| | GROUP | - | Read access |
| | WORLD | - | No access |
| b. FILE2 | SYSTEM | - | Unchanged |
| | MEMBER | - | Unchanged |
| | GROUP | - | Read/write access |
| | WORLD | - | Unchanged |

2.3.4 Asterisk Convention - Wild Cards

PIP allows wild cards to be specified by means of an asterisk character in the file specifier. The * character in one or more fields of a file specifier stands for "all"; e.g., all files, types or all versions. Wild card use, however, is restricted in some cases. Sections 2.3.4.1 and 2.3.4.2 describe the allowable uses and restrictions on wild cards for input and output files.

2.3.4.1 Wild Cards in Output File Specifiers - Use of wild cards in the output file specifiers is very restricted. In the following types of command actions, the output file specifier may not have any wild cards:

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

When a list of files is to be copied, the output specifier must be *.*;* or default.

For the RENAME and ENTER commands, the output specifier may have wild cards mixed with specified fields. Furthermore, a wild card field may optionally be left null. In either case, the equivalent field of the input file specifier is used.

In all cases where wild cards are allowed in the output file

PERIPHERAL INTERCHANGE PROGRAM (PIP)

specifier, the wild card UIC form [*,*] (but not [n,*] or [*,n]) may be used to indicate that the output UIC is to be the same as the input UIC.

2.3.4.2 Wild Cards in Input Specifiers - The following wild card features are provided for input file specifiers:

- *.*;* means all versions of all files.
- *.DAT;* means all versions of all files of type DAT.
- TEST.*;* means all versions and all types of files named TEST.
- TEST.DAT;* means all versions of file TEST.DAT.
- *.* means the most recent version of all files.
- *.DAT means the most recent version of all files with type DAT.
- TEST.* means the most recent version of all types of files named TEST.
- TEST.DAT means the most recent version of TEST.DAT.

The following wild card UIC features are also provided:

- [*,*] means all group, member number combinations (each from 1 to 377 octal);
- [n1,*] means all member numbers under group n1; and
- [*,n2] means all group numbers for member n2.

2.3.5 File Identification Option

Wherever a file specifier is used to describe an already existing file, the /FI (file identification) option can be used. This option allows the user to specify the file he wants to access by specifying the device, unit, and its unique file identification number. This file identification is assigned to the file at file creation time by the RSX-11M system. Although the file identification number of a file is normally invisible to the user, it can be obtained by using the PIP FULL LIST command /FU. (Section 2.4.8 contains a description of the LIST command). This will give the user a full or complete directory listing of his files. The full list of his directory contains, among other pertinent information about the file, its unique file identification number. The user need only specify this number using the /FI option to access his files. The /FI option is specified in the following format:

/FI:n1:n2

where n1 and n2 are the file number and file sequence number of the file, respectively.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

2.4 PIP COMMANDS

PIP commands are in the form of switches appended to optional file specifiers. Command switches and functions are summarized in Table 2-2.

NOTE

Only one of the PIP command switches listed in Table 2-2 can appear on a command line. More than one subswitch on a line is legal, as described in the following command descriptions.

Table 2-2
PIP Functions and Commands

Command	Switch	Function
Append	/AP	Add files to the end of an existing file.
Copy	No switch	Copy a file.
Default	/DF	Change PIP's default device or UIC.
Delete	/DE	Delete one or more files.
Enter	/EN	Enter a synonym for a file in a directory file.
Free	/FR	Print out available space on specified volume.
Identify	/ID	Identify the version of PIP being used.
List	/LI	List a directory file.
Merge	/ME	Concatenate two or more files into one file.
Protect	/PR	Change the protection of a file.
Purge	/PU:n	Delete obsolete version(s) of a file.
Remove	/RM	Remove a file entry from a directory.
Rename	/RE	Change the name of a file.
Spool	/SP	Specify a list of files to be printed and deleted.
Unlock	/UN	Unlock a file.
Update	/UP	Rewrite an existing file.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

2.4.1 APPEND Command (/AP)

FUNCTION

The APPEND command opens an existing file and appends the input file(s) onto the end of it.

FORMAT

outfile=infile-1[,infile-2,...,infile-n]/AP[/FO]

where:

outfile is the output file specifier in the format:

dev:[uic]filename.typ;ver[/AP] [/FO]

NOTES

1. No wild card specifiers are allowed in the output file specifier.
2. The file type and record attributes are taken from the existing file.
3. No defaults are allowed for the file name or file type.

infile is the input file specifier in the format:

dev:[uic]filename.typ;ver/AP

NOTE

If filename, file type, and version are null, then *.*;* is the default.

/AP is the APPEND switch

/FO is the Set File Ownership subswitch which specifies that the owning UIC of the output file corresponds to the directory into which the file was entered. If the /FO subswitch is not specified, the owning UIC of the new file is the UIC under which PIP is running, regardless of the directory into which the files were entered (see COPY command for examples using /FO).

EXAMPLE

PIP>DK1:FILE1.DAT;l=FILE2.DAT;l,FILE3.DAT;l,FILE4.DAT;l/AP↵

FILE1.DAT;l on DK1: will be opened, and the contents of FILE2.DAT;l, FILE3.DAT;l and FILE4.DAT;l will be appended to it.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

2.4.2 COPY Command (No Switch) and MERGE Command (/ME)

FUNCTION

The COPY command is used to create a copy of a file on the same or another device. The COPY command is the PIP default command, if an implicit output file specifier is used and no command switch is specified. When an explicit output file specifier and only one input file specifier are contained in a command line, the PIP default command is also COPY. The MERGE command is used to create a new file from two or more existing files. If an explicit output file specifier is used and more than one input file is named without an appended switch, the MERGE command becomes the PIP default command.

FORMAT

outfile = infile-1[,infile-2,...,infile-n]

where:

outfile is the output file specifier in the format:
 dev:[uic]filename.type;ver/subswitch

NOTE

If the output filename, file type, and version are either null or *.*;*, the input filename, file type, and version are preserved. See /NV and /SU subswitches. If any of the output file name, file type, or version fields is present, none may be wild and there may be only one input file specifier for a COPY command.

infile is the input file specifier in the format:
 dev:[uic]filename.type;ver/subswitch

NOTE

If the filename, file type, and version fields are all null, then *.*;* is the default.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

EXAMPLES

1. PIP>DK1:SAMP.DAT=DK2:TEST.DAT↵
Copy the latest version of file TEST.DAT from DK2: to DK1:
as SAMP.DAT.
2. PIP>DK1:[*,*]=DK0:[11,*]↵
Copy all files from all members in group number 11 of DK0:
to DK1:, preserving the UIC.
3. PIP>LP:=*.LST↵
Copy the latest version of all files with a type of .LST to
the line printer.
4. PIP>DK1:SAMP.DAT=DK2:TEST.DAT;1,NEW.DAT;2/ME↵
Concatenate version 1 of file TEST.DAT and version 2 of file
NEW.DAT from DK2: generating file SAMP.DAT on DK1.
5. PIP>DK1:=TESTPROG.MAC, .OBJ↵
Copy the latest versions of TESTPROG.MAC and TESTPROG.OBJ
from the system device to DK1:.
6. PIP>DK1:=DK0:*.DAT;*↵
Copy all versions of all of the files of the type DAT from
DK0: to DK1:.
7. PIP>DT0:=[200,10]*.*;*↵
Copy all files under [200,10] from system device to DT0:.
8. PIP>DP0:[200,10]=DT0:*.*↵
Copy all of user's files from DT0: to DP0:[200,10].

The optional subswitches described in Table 2-3 are used with both the COPY and MERGE commands.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

Table 2-3
PIP COPY and MERGE Command Subswitches

Subswitch	Description
	<p style="text-align: center;">NOTE</p> <p>All of the subswitches below can appear on either the output or input file specifier. If the subswitch is placed on an input file specifier, it pertains only to that file. If the subswitch is placed on the output file specifier, it pertains to the entire list of input specifiers.</p>
/BL:n[.]	<p>Blocks Allocated - This switch specifies the number of contiguous blocks to allocate to the output file, where n is an octal or decimal value (decimal values must be followed by a decimal point). The /BL:n switch is useful for copying a contiguous file and changing its size.</p>
/CO	<p>Contiguous Output - This switch causes the output file to be contiguous.</p>
/-CO	<p>Noncontiguous Output - This switch causes the output file to be noncontiguous.</p>
	<p style="text-align: center;">NOTE</p> <p>If none of the above subswitches is specified, PIP defaults to the size and attributes of the input file.</p>
/FO	<p>Set File Ownership - This subswitch specifies that the owning UIC of the output file corresponds to the directory into which the file was entered. If the /FO switch is not specified, the owning UIC of all new files is the UIC under which PIP is running, regardless of the directory in which the files were entered. This subswitch can be used with both COPY and MERGE commands.</p> <p>If PIP is running under the UIC [1,1], the command:</p> <p style="text-align: center;">DK0:[200,200]=DK1:[200,220]TEST.DAT</p> <p>results in a new file being created in the [200,200] directory on DK0: and the file being owned by UIC [1,1].</p> <p>However, the command:</p> <p style="text-align: center;">DK0:[200,200]=DK1:[200,220]TEST.DAT/FO</p>

PERIPHERAL INTERCHANGE PROGRAM (PIP)

Table 2-3 (Cont.)
PIP COPY and MERGE Command Subswitches

Subswitch	Description
/SU	<p>results in the output file being owned by UIC [200,200].</p> <p>Supersede - This switch allows the user to copy a file of which the name, type, and version of the file already exists in the specified output directory file. The old file is deleted and replaced with the specified input file.</p>
/NV	<p>New Version - This switch allows the user to force the output version number of the file being copied to the latest version plus one of the file already in the output directory. If the file does not already exist in the output directory, a version of one is assigned. The results which occur when the /NV switch is specified are depicted in Figure 2-1.</p>

Examples Using /FO Subswitch

NOTE

When using the /FO subswitch, PIP must be running under a UIC that has write access to all output directories.

1. PIP>DK1:[*,*]/FO=DP0:[13,10],[32,10],[34,10] ↵

Copy all the files from the specified input directories to the corresponding directories of DK1:, making the file owners agree with the output directories.

2. PIP>DK1:[*,*]=DK0:[*,10]*.MAC/FO ↵

Copy all the .MAC files from all group numbers with member number 10 to DK1:, preserving the directory UIC and setting the file owner for each file to the directory UIC.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

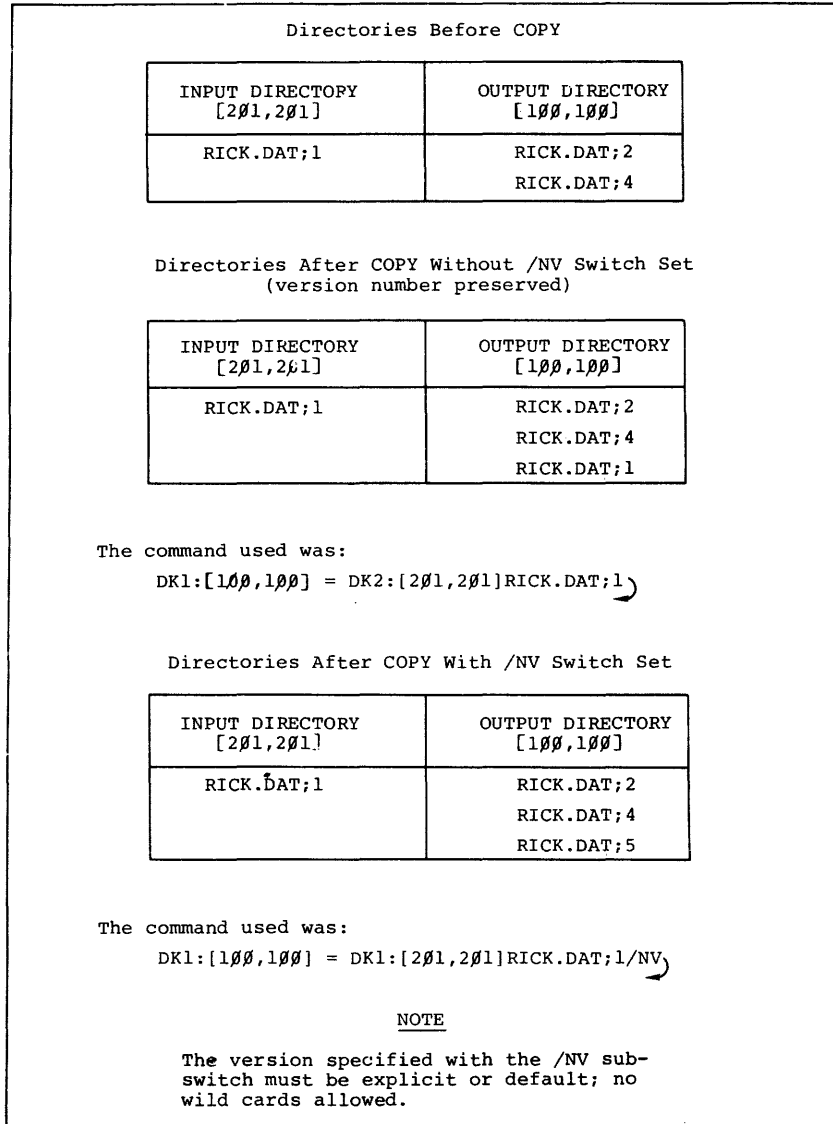


Figure 2-1
Results of COPY Command With and Without /NV Specified

2.4.3 DEFAULT Command (/DF)

FUNCTION

The DEFAULT command provides the user with a facility to change the default device or UIC.

The normal default UIC is the UIC under which PIP is currently running; that is, the UIC specified in the last MCR SET/UIC command, or that specified with the /UIC switch in a RUN ...PIP command.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

NOTE

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

The normal default device for PIP is SY0:.

FORMAT

dev:[group,member]/DF

where:

dev:	if specified, is the new default device to be applied to subsequent PIP command strings.
[group,member]	if specified, is the new default UIC to be applied to subsequent PIP command strings.
/DF	is the DEFAULT command switch.

EXAMPLES

1. PIP>[27,27]/DF ↵
Set the default UIC to [27,27].
2. PIP>DK1:/DF ↵
Set the default device to DK1:.
3. PIP>DK1:[27,27]/DF ↵
Set the default device to DK1:, and the default UIC to [27,27].

2.4.4 DELETE Command (/DE)

FUNCTION

The DELETE command provides the user with a facility to delete files.

FORMAT

infile-1[,infile-2,...,infile-n]/DE

where:

infile is the file specifier for the file to be deleted in the format:

dev:[uic]filename.type;version/switch

PERIPHERAL INTERCHANGE PROGRAM (PIP)

NOTES

1. A version number must always be specified when using the DELETE command switch.
2. When deleting files, a version number of ;-1 may be used to specify the oldest version of a file. An explicit version of ;0 or ; may be specified to signify the most recent version.

Examples

a. PIP>TEST.DAT;-1/DE ↵

Delete the oldest version
of file TEST.DAT

b. PIP>TEST1.DAT;0,TEST2.DAT;/DE ↵

Delete the latest version of
files TEST1.DAT and TEST2.DAT.

Wild cards in the filename or file type fields are illegal when a version of ;-1, ;0, or ; is specified.

3. The file specifier must be issued because a null filename, file type, and version does not default to *.*;*.
4. The input file specifier can take all the usual forms, including wild cards, even in the group, member number [UIC]. The only special requirement is that the version number field must always be explicit or "*". It cannot be defaulted to the most recent version if wild cards are used.

/DE is the DELETE command switch.

EXAMPLES

1. PIP>TEST.DAT;5/DE ↵

Delete version 5 of the file TEST.DAT in the default directory on the default device.

2. PIP>TEST.DAT;1,;2/DE ↵

Delete versions 1 and 2 of file TEST.DAT in the default directory on the default device.

3. PIP>*.OBJ;*;*.TMP;*/DE ↵

Delete all versions of all files of the type OBJ and TMP from the current default directory on the default device.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

2.4.5 ENTER Command (/EN)

FUNCTION

The ENTER command provides the facility to enter a synonym for a file in a directory or directories, thus allowing the file to be accessed by more than one name. Also provided is a subswitch, New Version (/NV), which allows the user to force the version number of the file being entered into the directory to a number one greater than the latest version for the file.

FORMAT

outfile=infile-1[,infile-2,...,infile-n]/EN[/NV]

where:

outfile is the file specifier to be given to the new directory entry. The output file specifier has a special property in that the filename, type, and version are individually allowed to be explicit, wild card (*) or defaulted (null). A name, type, or version field that is either wild card (*) or defaulted (null) means that the corresponding field of the input file is to be used.

infile is the file specifier for the input file in the format:

dev:[uic]filename.type;ver/sw[/subsw]

If no device is specified, in either the input or output file specifier, then the current default device is assumed to be the default device. If a device is specified on either the input or output side, that device is defaulted for the other side. If both the input side and the output side explicitly reference different devices, PIP will flag this as an error and request that the line be reentered.

The default input file specifier is *.*.*.

/EN is the ENTER command switch.

/NV is the New Version subswitch.

NOTE

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 will be 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 will be forced to a number one greater than the latest version.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

EXAMPLE

PIP> [101,101]TWIG/EN=[200,200]RICK.DAT;1 ↵

<u>Before</u>	
DIRECTORY [200,200]	DIRECTORY [101,101]
RICK.DAT;1	JEN.OBJ;2
	LAU.OBJ;3

<u>After</u>	
DIRECTORY [200,200]	DIRECTORY [101,101]
RICK.DAT;1	JEN.OBJ;2
	LAU.OBJ;3
	TWIG.DAT;1

NOTE

The directory items for RICK.DAT;1 and TWIG.DAT;1 both reference the same file.

Figure 2-2
Sample Directories Before and After Execution of /EN Command

2.4.6 FREE Command (/FR)

FUNCTION

The FREE command provides the user with the ability to print out the available space on a specified volume.

FORMAT

dev:/FR

The output from the /FR command is shown below.

dev: HAS nnnn. BLOCKS FREE, nnnn. BLOCKS USED OUT OF nnnn.

2.4.7 IDENTIFY Command (/ID)

FUNCTION

The IDENTIFY command allows the user to identify the version of PIP being used.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

FORMAT

/ID

NOTE

When this command is invoked, the version number is listed on the input terminal as follows:

PIP VERSION Mvvee

where:

vv is the version number.

ee is the edit number.

EXAMPLE

PIP>/ID ↵

PIP VERSION M1301

2.4.8 LIST Command (/LI)

FUNCTION

The LIST command provides the user with the facility to list one or more directories. Also provided are three alternate mode switches (/BR, /FU and /TB) which allow the user to specify a choice of directory listing formats. These switches are described in Table 2-4.

FORMAT

[listfile]=infile-1[,infile-2,...,infile-n]/LI [/switch]

where:

listfile is the listing file specifier in the format:

dev:[uic]filename.type;ver

NOTE

If listfile is not specified, it defaults to TI:.

infile is the input file specifier in the format:

dev:[uic]filename.typ;ver/switch

NOTE

The default for this file is *.*;*

PERIPHERAL INTERCHANGE PROGRAM (PIP)

/LI is the LIST command switch. This switch causes the following information to be listed.

1. filename.type;version
2. number of blocks used (decimal)
3. file code:
 - (null) = non-contiguous
 - C = contiguous
 - L = locked
4. creation date and time

/switch are the alternate mode switches of the LIST command described in Table 2-4.

Table 2-4
LIST Command Switches

Switch	Description
/BR	This switch specifies the brief form of directory listing. This switch will cause only the filename, type, and version to be listed.
/FU[:n]	<p>This switch specifies the full directory format. This switch has an optional modifier which allows the specification of the number of characters to be printed on a line.</p> <p>If specified, n is the number of characters per line. If not specified, the number is defaulted to (80.). This switch causes the following information to be listed:</p> <ol style="list-style-type: none"> 1. filename. type; version 2. file identification number in the format: <ul style="list-style-type: none"> (file number, file sequence number) 3. number of blocks used/allocated (base 10) 4. file code <ul style="list-style-type: none"> (null) = non-contiguous C = contiguous L = locked 5. creation date and time

PERIPHERAL INTERCHANGE PROGRAM (PIP)

Table 2-4 (Cont.)
LIST Command Switches

Switch	Description
/TB	<p>6. owner UIC and file protection in the format:</p> <p>[group,member]</p> <p>[system,owner,group,world]</p> <p style="text-align: center;">NOTE</p> <p style="text-align: center;">These protection fields can contain the values R,W,E,D.</p> <p>where:</p> <p style="padding-left: 40px;">R = Read access permitted W = Write access permitted E = Extend privilege permitted D = Delete privilege permitted</p> <p>7. date and time of the last update plus the number of revisions.</p> <p>8. summary line:</p> <p style="padding-left: 40px;">The number of blocks used, the number of blocks allocated, and the number of files are printed.</p> <p style="padding-left: 40px;">The summary line is not printed when the /BR switch is specified.</p> <p>This switch specifies that the user wants only the summary line in the following format.</p> <p style="padding-left: 40px;">TOTAL OF nnnn./mmmm. BLOCKS IN xxxx. FILES</p> <p>where:</p> <p style="padding-left: 40px;">nnnn = blocks used</p> <p style="padding-left: 40px;">mmmm = blocks allocated</p> <p style="padding-left: 40px;">xxxx = number of files</p> <p style="text-align: center;">NOTE</p> <p>Figure 2-3 contains sample directory listings in the various formats.</p>

PERIPHERAL INTERCHANGE PROGRAM (PIP)

Total Blocks (/TB) Format

STORAGE USED/ALLOCATED FOR DIRECTORY DK2:[200,270]
15-JUL-75 15:46

TOTAL OF 145./150. BLOCKS IN 5. FILES

Brief (/BR) Format

DIRECTORY DK2:[200,270]

CKTST.MAC#6
IOTST.MAC#4
IOTST.TSK#1
CKTST.TSK#1
CKTST.MAC#7

Standard (/LI) Format

DIRECTORY DK2:[200,270]
15-JUL-75 15:46

CKTST.MAC#6	3.		15-JUL-75 15:39
IOTST.MAC#4	4.		15-JUL-75 15:39
IOTST.TSK#1	69.	C	15-JUL-75 15:39
CKTST.TSK#1	69.	C	15-JUL-75 15:40
CKTST.MAC#7	0.	L	15-JUL-75 15:40

TOTAL OF 145. BLOCKS IN 5. FILES

Full (/FU) Format

DIRECTORY DK2:[200,270]
15-JUL-75 15:46

CKTST.MAC#6	(10,10)	3./3.	15-JUL-75 15:39
[200,270][RWED,RWED,RWED,R]			
IOTST.MAC#4	(11,11)	4./4.	15-JUL-75 15:39
[200,270][RWED,RWED,RWED,R]			
IOTST.TSK#1	(7,12)	69./69.	C 15-JUL-75 15:39
[200,270][RWED,RWED,RWED,R]			
CKTST.TSK#1	(12,13)	69./69.	C 15-JUL-75 15:40
[200,270][RWED,RWED,RWED,R]			
CKTST.MAC#7	(13,14)	0./5.	L 15-JUL-75 15:40
[200,270][RWED,RWED,RWED,R]			

TOTAL OF 145./150. BLOCKS IN 5. FILES

Figure 2-3
Directory Listing Examples

PERIPHERAL INTERCHANGE PROGRAM (PIP)

EXAMPLES

1. PIP>/LI ↵

Equivalent to TI:=/LI where the directory of the current default device and UIC is listed.

2. PIP>LP:=[*,*]/FU:132. ↵

List, on the line printer, in full format (132-column listing), all of the directories on the current default device.

3. PIP>TI:=TEST.DAT/FU ↵

List on TI: the full directory listing (80-column) for the latest version of TEST.DAT on the current default device and directory.

4. PIP>JUL13.DIR=[200,200]*.*/LI ↵

List the latest version of all files in directory [200,200] on the current default device to file JUL13.DIR in the default directory on the default device.

5. PIP>LP:=[11,*]*.CMD;*/LI ↵

List, on the line printer, all versions of all files of type .CMD in all directories in group 11.

6. PIP>LP:/BR=[11,11]*.CMD;*,*.DAT;*,*.MAC;1 ↵

List, on the line printer, in brief format, all versions of all files with a type of .CMD; all versions of all files with a type of .DAT; and all files of type MAC with a version number of 1. These files all reside in the directory [11,11] on the current default device.

2.4.9 PROTECT Command (/PR)

FUNCTION

The PROTECT command provides the facility to alter the protection of a file. File protection is provided for four categories as follows:

1. System - Specifies what categories of access the system UICs are allowed to the file (all group numbers less than or equal to 10 octal).
2. Owner - Specifies what categories of access the owner has allowed himself.
3. Group - Specifies what categories of access other members in the same group have.
4. World - Specifies what categories of access have been given all UICs not covered above.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

For each category, the user can specify whether that category can Read, Write, Extend, or Delete the file.

NOTE

Only the owner or a system UIC can alter the protection of a file.

FORMAT

```
infile-l/PR[/SY[:RWED]] [/OW[:RWED]] [/GR[:RWED]] [/WO[:RWED]] [/FO]
```

where:

infile is the file specifier for the file whose protection is being changed, in the format:

```
dev:[uic]filename.type;ver/switch
```

NOTE

File specifier must be issued because a null filename, file type, and version does not default to *.*;*.

/PR is the PROTECT command switch.

/SY,/OW,/GR, AND /WO are the PROTECT command subswitches which allow the user to specify the protection he wishes to assign to a file. These subswitches allow the user to specify which protection is to be altered (others are left intact). The values which follow the switch are any of the four letters R, W, E, D (for read, write, extend, delete) in any order. They specify which privileges the respective categories can have. If the subswitch is present and no value is given, then no privileges are granted for that category.

The subswitches are identified as follows:

```
/SY is the system subswitch.  
/OW is the owner subswitch.  
/GR is the group subswitch.  
/WO is the world subswitch.
```

NOTES

1. Protection can also be specified by an optional octal value on the /PR switch itself, in the format:

```
/PR:n
```


PERIPHERAL INTERCHANGE PROGRAM (PIP)

where n is the octal representation of the protection to be assigned to the file. This octal number is taken as the new protection word. The format of the protection word is shown in Figure 2-4.

2. A new protection value may be set at the same time as file ownership, or file ownership alone may be changed.

/FO

is the Set File Ownership subswitch which provides the facility to set the ownership of a file to that of the UIC of the directory in which it is entered. A new protection value can be set at the same time the file ownership is changed. If there are files in the [200,200] directory which are owned by another UIC, the following command:

```
PIP>[200,200]*.*;*/PR/FO ↵
```

would cause all files to be owned by [200,200].

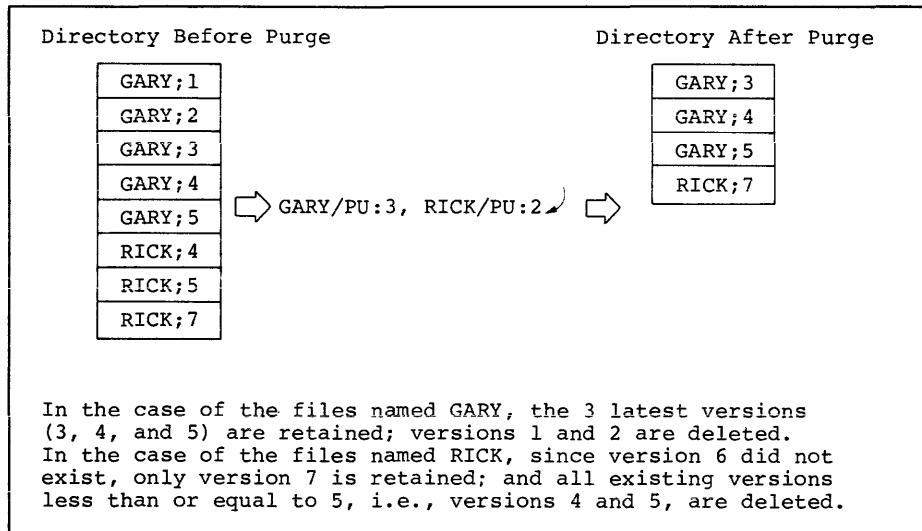


Figure 2-4
Format of Protection Word

EXAMPLES

1. PIP>TEST.DAT;5/PR/OW:RWE/GR:RWE:/WO ↵

Sets the protection so owner and group have RWE privileges (not delete), world has no access privileges, and system privileges are unchanged.

2. PIP>[*,*]*.*;*/PR:0 ↵

Sets the protection of all files so all categories are granted all access privileges.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

3. PIP>DK0:[*,*]*.*;*/PR/FO ↵

Sets all file owners to correspond with the directories in which they are entered.

2.4.10 PURGE Command (/PU[:n])

FUNCTION

The PURGE command provides the user with a facility to delete a specified range of obsolete versions of a file.

FORMAT

infile-1[,infile-2,...,infile-n]/PU[:n]

where:

infile is the file specifier for the file to be deleted in the format:

dev:[uic]filename.type/switch

/PU:n is the PURGE switch. The PURGE switch provides the user with a convenient way to delete old versions of files. If the optional value n is specified and the latest version of the file is m, then all existing versions greater than m-n are retained and all existing versions less than or equal to m-n are deleted (see Figure 2-5). Although it is useful to think of this command as deleting all but the n most recent versions, it is important to understand that if any versions are already deleted between m-n and m, then fewer than n versions will be retained.

If the value n is omitted, 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 file, no files are deleted.

NOTE

A version number is not required when using the PURGE switch. When specified, the version number field is ignored.

EXAMPLE

PIP>*.OBJ/PU,*.MAC/PU:2 ↵

Delete all but the highest version of all files with a type of .OBJ, and delete all but the two highest versions of all files with a type of .MAC.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

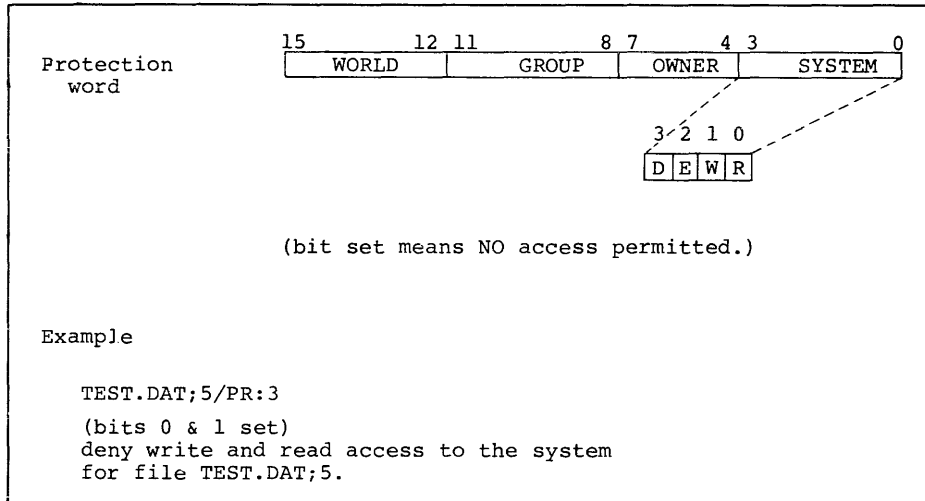


Figure 2-5
Use of Purge Switch

2.4.11 REMOVE Command (/RM)

FUNCTION

The REMOVE command allows the user to remove an entry from a directory file. Unlike the DELETE command, the REMOVE command does not delete the associated file; only the directory entry is removed. REMOVE is particularly useful for getting rid of directory entries which, for whatever reason, point to nonexistent files. It is also used to delete synonyms generated using the ENTER command. If an entry to an existing file is removed, that file can only be located using the VFY /LO switch (see Section 8.4.6).

FORMAT

infile-1[,infile-2,...,infile-n]/RM

where:

infile is the file specifier for the directory file entry to be removed in the format:

dev:[uic]filename.type;ver

NOTES

1. The file specifier must be issued because a null filename, file type, and version does not default to *.*;*.
2. The input file specifier allows full wild card facilities, but has the restriction that the version number must be specified explicitly or as a wild card.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

/RM is the REMOVE command switch.

EXAMPLE

```
PIP>DK1:[10,10]RICKSFILE.DAT;1/RM ↵
```

Removes the file entry RICKSFILE.DAT;1 from the directory [10,10] on DK1:.

2.4.12 RENAME Command (/RE)

FUNCTION

The RENAME command provides the user with the facility to change the name of a file. Also provided is a subswitch (/NV) which allows the user to force the renamed file to be a version number one greater than the latest version of the previously-existing file with the same name.

FORMAT

```
outfile=infile-1[,infile-2,...,infile-n]/RE[/NV]
```

where:

outfile is the file specifier to be given to the new file. The output file specifier has a special property in that the filename, type, and version are individually allowed to be explicit, wild card (*) or defaulted (null). A UIC, filename, type, or version field that is either wild card (*) or defaulted (null) means that the corresponding field of the input file is to be used. Thus, the rename command provides the facility to change one or more fields while preserving the others. The format of the output specifier is as follows:

```
dev:[uic]filename.type;ver/switch
```

infile is the file specifier of the file to be renamed. The input file specifiers are standard and allow wild cards in all fields, including UIC. This specifier is entered in the following format:

```
dev:[uic]filename.type;ver/switch
```

NOTES

1. A null filename, file type, and version defaults to *.*.*.
2. Renaming files across devices is not allowed. However, renaming across directories on the same device is allowed. Thus, it is possible to move files out of one directory into another, preserving the name, type, and version, or changing them if desired. This is permitted only if

PERIPHERAL INTERCHANGE PROGRAM (PIP)

PIP is running under a UIC which has write privileges on each of the directories involved.

3. If no device is specified in either the input or output file specifier, then the current default is assumed to be the default device. If a device is specified on either the input or output side, that device is defaulted for the other side. If both the input side and the output side explicitly reference different devices, PIP will flag this as an error and request that the line be reentered.

/RE is the RENAME command switch.

/NV is the New Version subswitch.

NOTES

1. The /NV subswitch allows the user to force the version number of the renamed file to a number one greater than the latest version for the file.
2. 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 will be 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 will have its version number forced to a number one greater than the latest version for the file.

EXAMPLES

1. PIP>TESTFILE.DAT;1=TEST.DAT;5/RE ↵

File TEST.DAT;5 is renamed TESTFILE.DAT;1.

2. PIP>BACKUP.*;*=TEST1.*;*,TEST2.*;*,TEST3.*;*/RE ↵

Rename all versions of all files with the names TEST1, TEST2, and TEST3 to BACKUP, preserving the type and version of each file.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

3. PIP>*. *;1=*. *;*/RE ↵

Rename all of the latest copies of files to version 1.

CAUTION

There should only be one version of each of these files.

4. PIP>[200,220]=[200,200]/RE ↵

Rename all files from [200,200] to [200,220], preserving the filename, type, and version of each file.

5. PIP>EXAMPLE.*;*=TEST.*;*/RE ↵

Rename all versions of all files with the name TEST to the name EXAMPLE, preserving the type and version of each file.

6. PIP>SAVE.DAT/RE/NV=OUTPUT.DAT;1 ↵

Rename OUTPUT.DAT;1 and force the version number to one greater than the latest version of SAVE.DAT. Figure 2-6 illustrates the results both with and without the /NV switch set.

<u>Directory Before Rename</u>
SAVE.DAT;2 SAVE.DAT;3 SAVE.DAT;4 OUTPUT.DAT;1 OUTPUT.DAT;2
<u>Directory After Rename Without /NV Switch Set</u>
SAVE.DAT;2 SAVE.DAT;3 SAVE.DAT;4 SAVE.DAT;1 OUTPUT.DAT;2
<u>Directory After Rename With /NV Switch Set</u>
SAVE.DAT;2 SAVE.DAT;3 SAVE.DAT;4 SAVE.DAT;5 OUTPUT.DAT;2

Figure 2-6
Results of Rename Command With and Without /NV Specified

2.4.13 SPOOL Command (/SP)

FUNCTION

The SPOOL command allows the user to specify a list of files to be printed asynchronously.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

FORMAT

infile-1[,infile-2,...,infile-n]/SP

where:

infile is the file specifier of the file to be spooled for printing in the format:

dev:[uic]filename.typ;ver/SP

NOTES

1. File specifier must be issued because a null filename, file type, and version does not default to *.*;*.
2. If the user specifies a file by its file identification number, the file will be printed. File identification numbers (/FI) are discussed in Section 2.3.5.
3. The line printer symbiont task (PRT...) must be installed in the system (see Appendix C for a description of the print spooler).

/SP is the SPOOL command switch.

EXAMPLE

PIP>RICK1.LST;1,KATHY.LST;1,/FI:12:22/SP ↵

Spool the files RICK1.LST;1, KATHY.LST;1, and the file whose file identification number is 12:22 for asynchronous printing.

2.4.14 UNLOCK Command (/UN)

FUNCTION

The UNLOCK command allows the user to unlock a file that was locked as a result of being improperly closed. If a program using File Control Services (FCS) has a file open with write access and exits without first closing the file, the file will be locked against further access as a warning that it may not contain proper information. Typically the following information would not have been written to the file:

1. The current block buffer being altered.
2. The record attributes which contain the end-of-file information.

By using the UNLOCK command, the user can access the file and determine the extent of the damage, perhaps taking appropriate corrective action.

FORMAT

infile-1[,infile-2,...,infile-n]/UN

where:

infile is the file specifier for the file to be unlocked,
in the format:

dev:[uic]filename.typ;ver/switch

NOTES

1. The file specifier must be given because a null filename, file type, and version does not default to *.*;*.
2. PIP must be running under the UIC of the file owner or a system UIC.

/UN is the UNLOCK switch

EXAMPLE

PIP>DK1:[100,100]RICK1.OBJ;3/UN ↵

File RICK1.OBJ;3 in directory [100,100] of device DK1: is unlocked.

2.4.15 UPDATE Command (/UP)

FUNCTION

The UPDATE command is similar to a COPY or MERGE command, except that an existing file is opened and new data is written into it from the beginning.

FORMAT

outfile=infile-1[,infile-2,...,infile-n]/UP[/FO]

where:

outfile is the file specifier for the file to be rewritten
in the format:

dev:[uic]filename.type;ver

As in the MERGE and the APPEND commands, the output file specifier must be explicit, i.e., no wild cards are allowed.

NOTE

The characteristics and record attributes of the output file are taken from the first input file.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

infile is the file specifier for the file to be copied into the file being rewritten in the format:

dev:[uic]filename.type;ver/switch

NOTE

A null filename, file type, and version defaults to *.*;*.

/UP is the UPDATE command switch.

/FO is the Set File Ownership subswitch which specifies that the owning UIC of the output file corresponds to the directory into which the file was entered. If the /FO switch is not specified, the owning UIC of all new files is the UIC under which PIP is running, regardless of the directory into which the file was entered (see the COPY command for examples using /FO).

EXAMPLE

PIP>DK1:SAMPLE.DAT;l=TEST1.DAT;l,TEST2.DAT;l,TEST3.DAT;l/UP ↵

The file SAMPLE.DAT;l on DK1: will be opened, and the contents of files TEST1.DAT;l, TEST2.DAT;l and TEST3.DAT;l will replace the data which already exists in the file.

2.5 PIP ERROR MESSAGES

Errors encountered by PIP during processing are reported to the user in the following format:

PIP -- <main error message>

<filename or filespec> - <secondary error message>

The filename or filespec, if present, identifies the file or set of files being processed when the error occurred. If the error was detected by the operating system, file system, or device driver, the secondary error message is included to explain the cause of the error.

PIP error messages are contained in message files on the system device. If PIP cannot access the message files, errors are reported in the following format:

PIP -- ERROR CODE nn.

<filename or filespec> - <Driver Code -mm.>

or

<QIO Error Code -qq.>

PERIPHERAL INTERCHANGE PROGRAM (PIP)

where:

nn is one of the PIP error codes contained in Table 2-5.

-mm is one of the standard system, file primitive, or file control service codes listed in Appendix I of the RSX-11 I/O Operations Reference Manual.

-qq is one of the directive error codes listed in Appendix I of the RSX-11 I/O Operations Reference Manual.

The PIP error messages, their descriptions and suggested user actions are as follows.

PIP -- ALLOCATION FAILURE - NO CONTIGUOUS SPACE

Description

Contiguous space available on the output volume is insufficient for the file being copied.

Suggested User Action

Delete all files that are no longer required on the output volume, and reenter the command line.

PIP -- ALLOCATION FAILURE ON OUTPUT FILE

or

PIP -- ALLOCATION FAILURE - NO SPACE AVAILABLE

Description

Space available on the output volume is insufficient for the file being copied.

Suggested User Action

Delete all files that are no longer required on the output volume, and reenter the command line.

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

Description

The user has specified a wild card "*" for an output filename where use of a wild card is explicitly disallowed.

Suggested User Action

Reenter the command line with the proper output file explicitly specified.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

PIP -- CANNOT FIND DIRECTORY FILE

Description

The user has specified a UFD that does not exist on the specific volume.

Suggested User Action

Reenter the command line, specifying the correct UIC.

PIP -- CANNOT FIND FILE(S)

Description

The file(s) specified in the command were not found in the designated directory.

Suggested User Action

Check the file specifier and reenter the command line.

PIP -- CANNOT RENAME FROM ONE DEVICE TO ANOTHER

Description

The user has attempted to rename a file across devices.

Suggested User Action

Reenter the command line, renaming the file on the input volume; then, enter another command to transfer the file to the originally intended volume.

PIP -- CLOSE FAILURE ON INPUT FILE

or

PIP -- CLOSE FAILURE ON OUTPUT FILE

Description

For some reason, the input or output file cannot be properly closed. The file will be locked to indicate possible corruption.

Suggested User Action

Reenter the command line. If the error recurs, run the validity check of the file structure verification utility (VFY) against the volume in question to determine if it is corrupted. VFY is described in Chapter 8.

PIP -- COMMAND SYNTAX ERROR

Description

The user has entered a command in a format that does not conform to syntax rules.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

Suggested User Action

Reenter the command line with the correct syntax.

PIP -- DEVICE NOT MOUNTED

Description

The message is self-explanatory.

Suggested User Action

Mount the device, and reenter the command line.

PIP -- DIRECTORY WRITE PROTECTED

Description

PIP could not remove an entry from a directory because the device was write-protected, or because of privilege violation.

Suggested User Action

Write enable the unit, or have the owner of the directory change its protection.

PIP -- ERROR FROM PARSE

Description

The specified directory file does not exist.

Suggested User Action

Reenter the command line with the correct UIC specified.

PIP -- FAILED TO ATTACH OUTPUT DEVICE

or

PIP -- FAILED TO DETACH OUTPUT DEVICE

Description

An attempt to attach/detach a record-oriented output device has failed. This is usually caused by the device being off-line or not resident.

Suggested User Action

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

PIP -- FAILED TO DELETE FILE

or

PERIPHERAL INTERCHANGE PROGRAM (PIP)

PIP -- FAILED TO MARK FILE FOR DELETE

Description

The user has attempted to delete a protected file.

Suggested User Action

Request PIP under the correct UIC and reenter the command line.

PIP -- FAILED TO ENTER NEW FILE NAME

Description

The user has specified a file that already exists in the directory file, or the user does not have the necessary privileges to make entries in the specified directory file.

Suggested User Action

Reenter the command line, ensuring that the filename and UIC are specified correctly, or request PIP under the correct UIC and reenter the command line.

PIP -- FAILED TO FIND FILE(S)

Description

The file(s) specified in the command line were not found in the designated directory.

Suggested User Action

Check the file specifier and reenter the command line.

PIP -- FAILED TO GET TIME PARAMETERS

Description

An internal system failure occurred while PIP was trying to obtain the current date and time.

Suggested User Action

Reenter the command line. If the problem persists, consult software support representative.

PIP -- FAILED TO OPEN STORAGE BITMAP FILE

Description

PIP could not read the specified volume's storage bit map, usually because of a privilege violation.

Suggested User Action

Retry by running PIP under a system UIC, or have the system manager change the protection on the storage bit map.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

PIP -- FAILED TO READ ATTRIBUTES

Description

The user's volume is corrupted or the user does not have the necessary privileges to access the file.

Suggested User Action

Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check of the file structure verification utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 8.

PIP -- FAILED TO REMOVE DIRECTORY ENTRY

Description

PIP could not remove an entry from a directory because the unit was write-protected, or a privilege violation was detected.

Suggested User Action

Write enable the unit, or have the owner of the directory change its protection.

PIP -- FILE IS LOST

Description

PIP has removed a file from its directory, failed to delete it, and failed to restore the directory entry.

Suggested User Action

Run the lost check of the file structure verification utility (VFY) to recover the filename. VFY is described in Chapter 8.

PIP -- FAILED TO SPOOL FILE FOR PRINTING

Description

Insufficient system dynamic memory is available, or the spooler task is not installed.

Suggested User Action

Wait for spooler queue to empty or install the spooler task and reenter the command line.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

PIP -- FAILED TO WRITE ATTRIBUTES

Description

The user volume is corrupted or the user does not have the necessary privileges to write the file attributes.

Suggested User Action

Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check of the file structure verification utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted. VFY is described in Chapter 8.

PIP -- FILE NOT LOCKED

Description

The user issued an unlock command for a file that was not locked.

Suggested User Action

Reenter the command line, specifying the correct file.

PIP -- ILLEGAL COMMAND

Description

The user has entered a command that is not recognized by PIP.

Suggested User Action

Reenter the command line with the PIP command correctly specified.

PIP -- ILLEGAL SWITCH

Description

The user has specified a switch that is not a legal PIP switch or has used a legal switch in an illegal manner.

Suggested User Action

Reenter the command line with the correct switch specification.

PIP -- ILLEGAL "*" COPY TO SAME DEVICE AND DIRECTORY

Description

The user has attempted to copy all versions of a file into the same directory that is being scanned for input files. This results in an infinite number of copies of the same file.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

Suggested User Action

Reenter the command line, renaming the files or copying them into a different directory.

PIP -- ILLEGAL USE OF WILD CARD VERSION

Description

The use of a wild card version number in the attempted operation results in inconsistent or unpredictable output.

Suggested User Action

Reenter the command line with different options or with explicit or default version number.

PIP -- I/O ERROR ON INPUT FILE

or

PIP -- I/O ERROR ON OUTPUT FILE

Description

One of the following conditions may exist:

1. The device is not on-line,
2. The device is not mounted.
3. The hardware has failed.
4. The volume is full (output only).
5. Input file is corrupted.

Suggested User Action

1. Determine which condition exists.
2. Rectify the condition.
3. Reenter the command line.

PIP -- EXPLICIT OUTPUT FILENAME REQUIRED

Description

This message is self explanatory.

Suggested User Action

Reenter the command line with the output filename explicitly specified.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

PIP -- NOT A DIRECTORY DEVICE

Description

The user has issued a directory-oriented command to a device (such as a printer) that does not have directories.

Suggested User Action

Reenter the command line without specifying a UIC.

PIP -- NOT ENOUGH BUFFER SPACE AVAILABLE

Description

PIP has insufficient I/O buffer space to perform the requested command.

Suggested User Action

Have the system manager install PIP in a larger partition.

PIP -- NO SUCH FILE(S)

Description

The file(s) specified in the command were not found in the designated directory.

Suggested User Action

Check the file specifier and reenter the command line.

PIP -- ONLY[*,*] IS LEGAL AS DESTINATION UIC

Description

The user has specified a UIC other than [*,*] as the output file UIC for a copy.

Suggested 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

Description

The specified file could not be opened. One of the following conditions may exist:

1. The file is protected against access.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

2. A problem exists on the physical device (e.g., device cycled down).
3. The volume is not mounted
4. The specified file directory does not exist.
5. The named file does not exist in the specified directory.

Suggested User Action

1. Determine which condition exists.
2. Rectify the condition.
3. Reenter the command line.

PIP -- OUTPUT FILE ALREADY EXISTS-NOT SUPERSEDED

Description

An output file of the same name, type, and version as the file already exists.

Suggested User Action

Retry the copy with the /NV switch to assign a new version number or the /SU switch to supersede the output file.

PIP -- TOO MANY COMMAND SWITCHES - AMBIGUOUS

Description

The user has specified too many switches, or the switches conflict.

Suggested User Action

Reenter the command line, specifying the correct set of switches.

PIP -- VERSION MUST BE EXPLICIT OR "*"

Description

The version number of the specified file must be expressed explicitly or as a wild card "*".

Suggested User Action

Reenter the command line with the version number correctly expressed.

PERIPHERAL INTERCHANGE PROGRAM (PIP)

2.6 PIP ERROR CODES

Table 2-5 identifies the error codes PIP issues when it doesn't have access to the message files. Message descriptions and suggested user actions are identical to the information contained in Section 2.5.

Table 2-5
PIP Error Codes

Error Code	PIP Error Message is:
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 WILD CARDS IN DESTINATION FILE NAME
8.	EXPLICIT OUTPUT FILE NAME REQUIRED
9.	ALLOCATION FAILURE - NO CONTIGUOUS SPACE
10.	ALLOCATION FAILURE - NO SPACE AVAILABLE
11.	ALLOCATION FAILURE ON OUTPUT FILE
12.	I/O ERROR ON INPUT FILE
13.	I/O ERROR ON OUTPUT FILE
14.	ILLEGAL USE OF WILD CARD VERSION
15.	OPEN FAILURE ON INPUT FILE
16.	OPEN FAILURE ON OUTPUT FILE
17.	CLOSE FAILURE ON INPUT FILE
18.	CLOSE FAILURE ON OUTPUT FILE
19.	FAILED TO DETACH OUTPUT DEVICE
20.	DEVICE NOT MOUNTED
21.	OUTPUT FILE ALREADY EXISTS - NOT SUPERSEDED
22.	FAILED TO MARK FILE FOR DELETE
23.	FILE IS LOST
24.	VERSION MUST BE EXPLICIT OR "*"
25.	ERROR FROM PARSE
26.	FAILED TO DELETE FILE
27.	CANNOT FIND DIRECTORY FILE
28.	FAILED TO ATTACH OUTPUT DEVICE
29.	FAILED TO GET TIME PARAMETERS
30.	NOT A DIRECTORY DEVICE
31.	FAILED TO WRITE ATTRIBUTES
32.	FAILED TO READ ATTRIBUTES
33.	FILE NOT LOCKED
34.	FAILED TO ENTER NEW FILE NAME
35.	FAILED TO RESTORE ORIGINAL DIRECTORY ENTRY - FILE IS LOST
36.	CANNOT RENAME FROM ONE DEVICE TO ANOTHER.
37.	FAILED TO SPOOL FILE FOR PRINTING
38.	(Not used in RSX-11M)
39.	FAILED TO OPEN STORAGE BITMAP FILE
40.	FAILED TO FIND FILE(S)
41.	CANNOT FIND FILE(S)
42.	NO SUCH FILE(S)
43.	FAILED TO REMOVE DIRECTORY ENTRY
44.	DIRECTORY WRITE PROTECTED
45.	NOT ENOUGH BUFFER SPACE AVAILABLE

CHAPTER 3
FILE TRANSFER PROGRAM (FLX)

3.1 INTRODUCTION TO FLX

FLX is a file utility program that performs file conversion between DOS-11 or RT-11 and Files-11 formats. FLX is designed to perform the following conversions:

- From DOS-11 to Files-11 format,
- From Files-11 to DOS-11 format,
- From DOS-11 to DOS-11 format,
- From Files-11 to Files-11 format,
- From RT-11 to Files-11 format,
- From Files-11 to RT-11 format,
- From RT-11 to RT-11 format.

FLX also allows the user to:

1. List directories of cassettes, RT-11, or DOS-11 volumes,
2. Delete files from DOS-11 and RT-11 volumes,
3. Initialize cassettes, RT-11, or DOS-11 volumes.

Valid DOS-11 devices are:

DK, DT, MT, MM, CT, PR, and PP

Valid RT-11 devices are:

DK, DT, and DX

All valid Files-11 devices are supported, including RSX-format cassette.

FILE TRANSFER PROGRAM (FLX)

3.2 INITIATING FLX

All RSX-11M utilities can be initiated in several ways. The various methods are explained in Section 1.2. The methods for FLX are:

```
>FLX ↵
>FLX command string ↵
>RUN ...FLX ↵
>RUN ...FLX/UIC=[group,member] ↵
>RUN $FLX ↵
>RUN $FLX/UIC=[group,member] ↵
```

3.3 FLX COMMAND STRING

The command string issued to FLX consists of an optional output file specifiers, and one or more input file specifier in the following format:

```
outfile=infile-1 [, infile-2, ...,infile-n]
```

For a complete description of file specifiers, see Section 1.3.

Wild cards are only valid for "infile" specifiers.

Version numbers are only valid for Files-11 files and may not be specified as wild. The standard rules for updating version numbers apply.

3.4 FILE TRANSFERS

File transfers are specified by a command containing an output and an input specifier together with a Format Mode Switch to denote the format to be used. For example:

```
FLX>DT0:/DO=DK1:SYS1.MAC/RS ↵
```

transfers SYS1.MAC from Files-11 DK1: to the DOS-11 DT0:.

```
FLX>DT1:/RT=DK0:SYS1.MAC/RS ↵
```

transfers SYS1.MAC from Files-11 DK0: to RT-11 DT1:.

If no /RS, /RT, or /DO is specified, FLX assumes /DO for input specifiers and /RS for output. This "DOS-to-RSX" default transfer direction can be dynamically modified -- see /DO and /RS switch descriptions.

FILE TRANSFER PROGRAM (FLX)

3.5 DOS VOLUME DIRECTORY MANIPULATION

3.5.1 DOS Directory Listings

The /LI or /DI switch instructs FLX to issue the directory of the cassette or DOS-11 volume specified in the input specifiers to the Files-11 file specified in the output specifier. If no output specifier is present, then the directory will be issued to TI:. For example:

```
FLX>LP:=DT0:[100,100]*.MAC/LI ↵
```

lists on the line printer the directory of all .MAC files under UIC [100,100] on DOS-11 DT0:.

3.5.2 Deleting DOS Files

Files may be deleted from DOS-11 disks or DECTape by using the /DE switch. The delete command string uses no output specifier. For example:

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

deletes SYS1.MAC under UIC [100,100] from the DOS-11 DK1:.

3.5.3 Initializing DOS-11 Volumes

Cassettes and DOS-11 volumes are initialized by using the /ZE switch. The initialize command has no output specifier. For example:

```
FLX>DT1:/ZE ↵
```

Initializes DOS-11 DT1:.

3.6 RT VOLUME DIRECTORY MANIPULATION

3.6.1 RT Directory Listings

The /LI or /DI switch, when combined with the /RT switch, instructs FLX to issue the directory of the RT-11 volume specified in the input specifiers to the Files-11 file specified in the output specifier. If no output specifier is present, then the directory will be issued to TI: For example:

```
FLX>LP:=DT0:*.MAC/LI/RT ↵
```

lists on the line printer the directory of all .MAC files on RT-11 DT0:.

FILE TRANSFER PROGRAM (FLX)

3.6.2 Deleting RT Files

Files may be deleted from RT-11 disks or DECTape by using the /DE switch in conjunction with the /RT switch. The delete command string uses no output specifier. For example:

```
FLX>DK1:SYS1.MAC/DE/RT ↵
```

deletes SYS1.MAC from RT-11 DK1:.

3.6.3 Initializing RT-11 Volumes

RT-11 volumes are initialized by using the /ZE switch in conjunction with the /RT switch. The initialize command uses no output specifier. For example:

```
FLX>DT1:/ZE/RT ↵
```

initializes RT-11 DT1:.

When initializing RT-11 volumes, the /ZE switch takes an optional argument in the form:

```
/ZE:n
```

where n specifies the number of extra words per directory entry. A directory segment consists of two disk blocks with a total of 512 words. The directory header uses five words, leaving 507 words for directory entries.

Normally, each directory entry is 7-words long and 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) are determined as follows:

$$\begin{aligned} \text{Directory Entries} &= (507 \div 7) - 2 \\ &= 72 - 2 = 70 \text{ entries} \end{aligned}$$

When extra words are specified (via /ZE:n) for directory entries, the number of directory entries are determined as follows:

$$\text{Directory Entries} = [507 \div (n+7)] - 2$$

For example, 61 entries can be made per directory segment if the switch /ZE:1 is used.

The /NU switch is used with the /ZE and /RT switches to specify the number of directory segments to allocate to the RT-11 volume. The /NU switch has the following form:

```
/NU:n
```

where n specifies the number of directory segments to allocate. If the /NU switch is not specified, or if n is not specified, four directory segments are allocated. The maximum number of segments which can be allocated is 37(8) or 31(10). For example:

```
FLX>DT0:/ZE:2/NU:6/RT ↵
```

FILE TRANSFER PROGRAM (FLX)

This command initializes RT-11 DT0:, allocates two extra words per directory entry, and allocates six directory segments.

3.7 FLX CASSETTE SUPPORT

FLX supports the DEC standard cassette file structure. Files may be transferred to and from cassettes in either RSX (/RS) or DOS (/DO) transfer mode. The transfer mode selected depends on the user's file format requirements.

3.7.1 Cassette File Formats

The file format for RSX or DOS cassette files is essentially the same. That is, they both conform to the DEC standard cassette file format. The DEC standard cassette file structure is described in Figure 3-1. The DEC standard cassette file label is described in Figure 3-2.

The difference between the RSX and DOS cassette file formats are as follows:

<u>RSX Format</u>	<u>DOS Format</u>
• Standard level 2	• Standard level 0
• 12-character file name (9+3)	• 9-character file name (6+3)
• Blocks of any size up to 512 bytes (128 bytes default)	• 128-byte blocks
• Version numbers	• No version numbers

RSX cassette file mode (level 2) is a superset of the DOS cassette file mode (level 0). Therefore, any cassette written in DOS mode can be read in RSX mode. The reverse of this, however, is only true when:

1. The RSX mode file was written with 128-byte blocks, and
2. The extra file header data (version number, etc.), which does not appear in DOS files, can be ignored.

RSX mode files and DOS mode files can be mixed on a given cassette, as long as a proper retrieval mode is utilized when the files are being accessed. 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.

FILE TRANSFER PROGRAM (FLX)

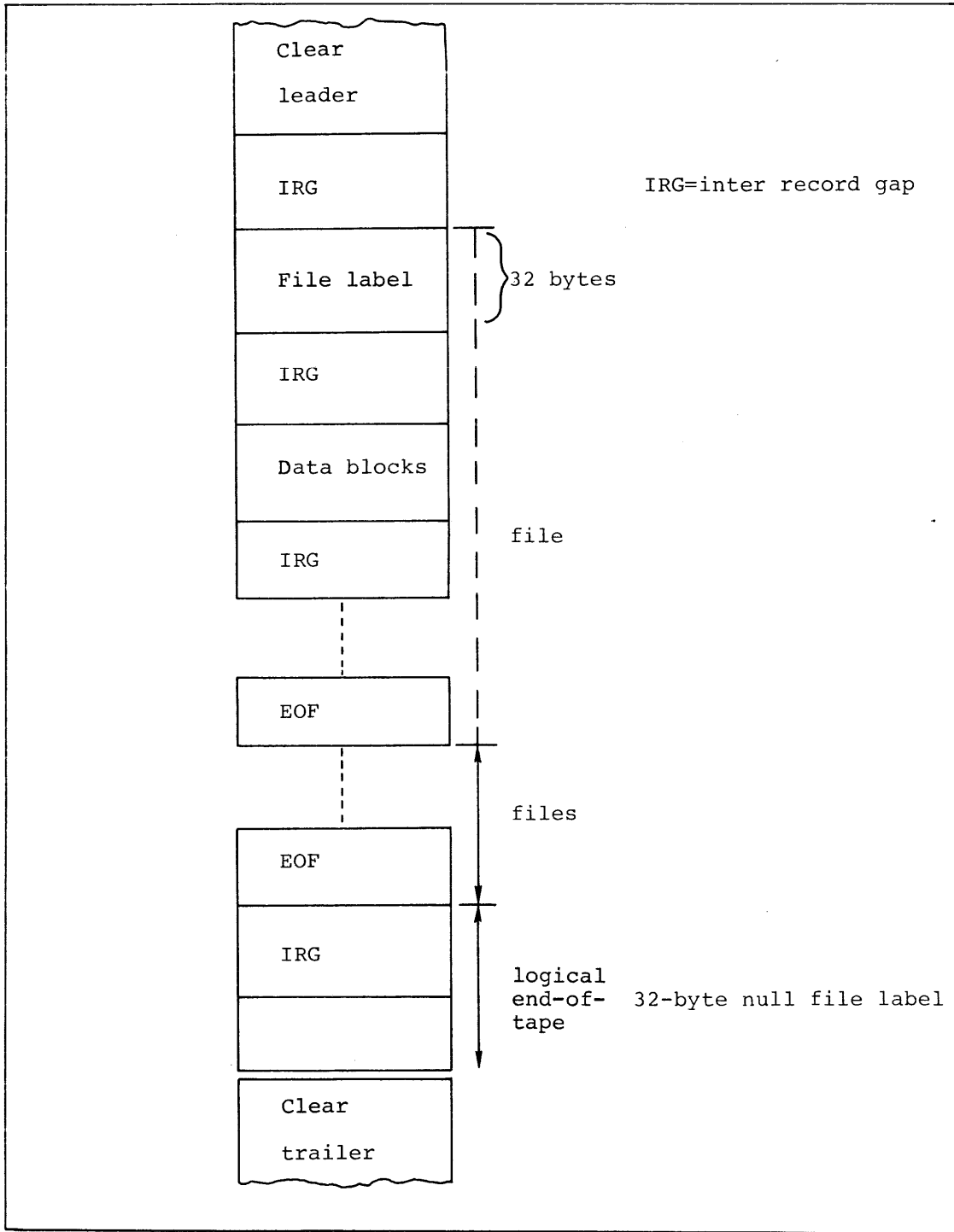


Figure 3-1
DEC Standard Cassette File Structure

FILE TRANSFER PROGRAM (FLX)

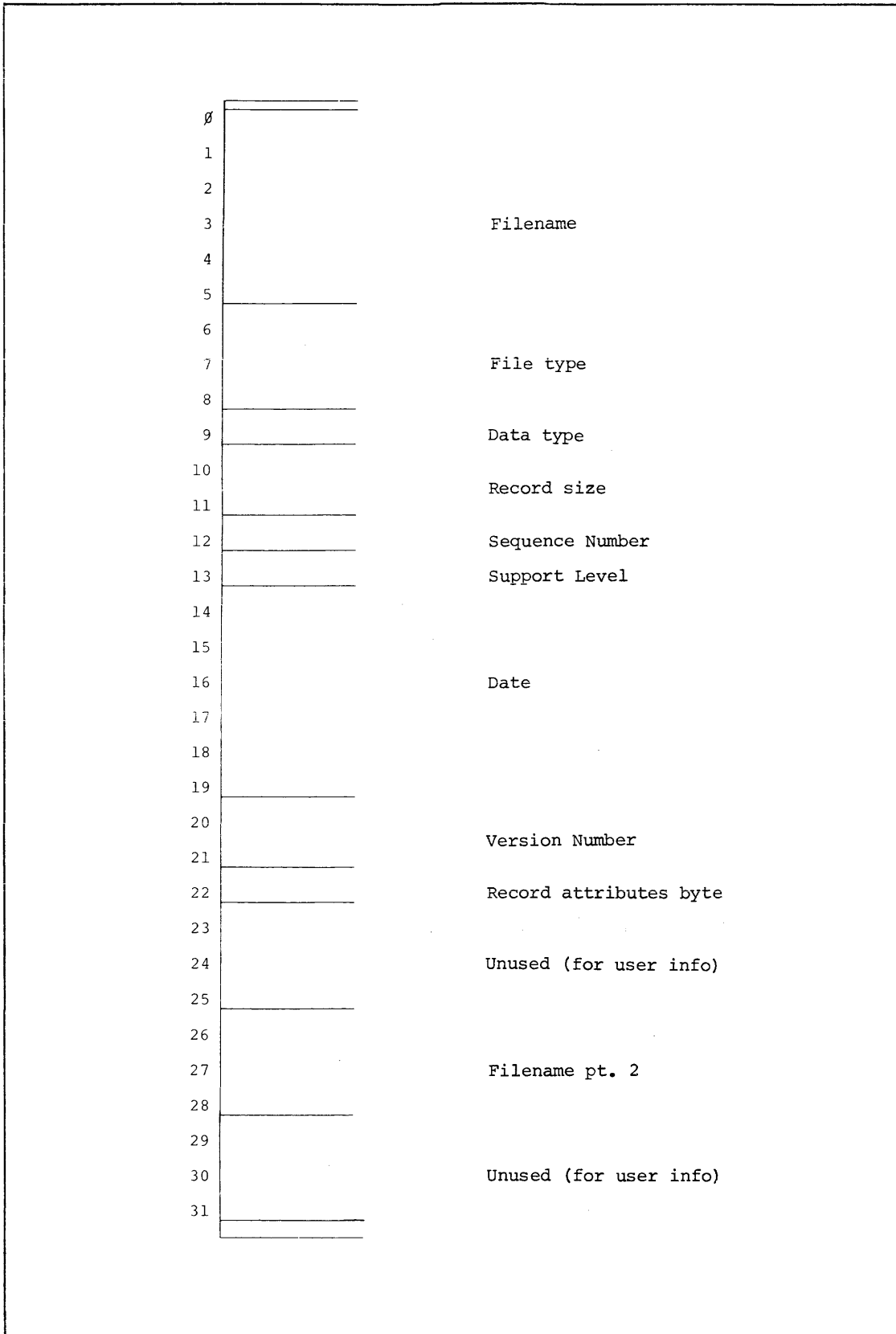


Figure 3-2
DEC Standard Cassette File Label

FILE TRANSFER PROGRAM (FLX)

Filename	6 ASCII Characters
File type	3 ASCII Characters
Data type	Describes the type of file to follow. Ø -- Unknown type 1 -- ASCII (formatted) 2Ø -- Formatted binary (DOS) 22 -- DOS LDA format 26 -- Unformatted binary TSK format
Record size	16-bit binary record size, where Ø < size < 512. and even.
Sequence number	Used for multi-volume files, Ø for sequence one.
Support level	Ø for DOS format 2 for RSX format
Date	6 ASCII characters in ddmmyy form
Version number	16-bit version number used in RSX mode. Undefined in DOS mode. The Files-11 standard rules for version numbers apply to file creation.
Record attributes	1 = FORTRAN carriage control present 2 = standard carriage control
Filename pt. 2	3 ASCII characters used in RSX mode. Undefined in DOS mode.

Figure 3-2 (Cont.)
DEC Standard Cassette File Label

FILE TRANSFER PROGRAM (FLX)

3.7.2 Multi-Volume Cassette Support

FLX supports multi-volume cassettes in both RSX and DOS formats. No special switches are required to notify FLX that a multi-volume file is being accessed.

3.7.2.1 FLX Output Files - When FLX detects the physical end-of-tape for an output cassette, the following sequence of events occurs.

1. FLX issues the following message:

FLX -- END OF VOLUME ON CASSETTE
CTn:[g,m]

2. The cassette is rewound.
3. FLX issues an additional message.

MOUNT NEW CASSETTE? (Y, Z (OUTPUT ONLY) OR CR)
FLX>

4. At this point, the user has three alternatives:
 - a. The user can mount the next output cassette volume and type Y, followed by a carriage return. If the user selects this alternative, the new cassette is rewound, FLX searches for the logical end-of-tape (end of the last file), and then continues transferring data onto the tape. If a file with the same name as the current input file is encountered on the new output cassette while searching for the logical end of tape, FLX prints the message:

FLX -- FILE ALREADY EXISTS

and then returns to step 3 above.

- b. The user can mount the next output cassette volume and type Z, followed by a carriage return. The new cassette is rewound, and FLX continues by transferring data onto it. Thus, the tape is effectively zeroed before data is transferred to it.
 - c. The user can enter a carriage return to terminate the transfer.

If the user selects alternative c, FLX assumes that EOF is desired, and issues the following message:

FLX -- REQUEST TERMINATED -- LAST BLOCK NOT WRITTEN

The last input file block processed was not written onto the tape.

FILE TRANSFER PROGRAM (FLX)

3.7.2.2 FLX Input File - 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 specifier on which the end-of-tape was detected:

FLX -- END OF VOLUME ON CASSETTE
CTn:[g,m] filename.type

2. The cassette is rewound.
3. FLX issues an additional message:

MOUNT NEW CASSETTE: (Y, Z (OUTPUT ONLY) OR CR)
FLX>

4. At this point the user has two alternatives:
 - a. The user can mount the next input cassette volume and type Y, followed by a carriage return to continue; or
 - b. The user can type a carriage return to terminate the transfer.

If the user selects alternative a, the new input cassette is rewound, and a validity check is performed on the file label and sequence number. If the file label and sequence number are correct, FLX begins processing data from the volume. If, however, the file label and sequence number are not correct, FLX issues the following message:

FLX -- FILE NOT FOUND

the process then returns to step 3 above.

If the user selects alternative b, FLX assumes that EOF is desired, and the transfer is terminated.

NOTE

If the input file is being processed as a formatted binary or an ASCII file, a format error may occur.

If the user types Z, FLX prints the message:

FLX -- BAD RESPONSE

the process then returns to step 3 above.

3.8 FLX PAPER TAPE SUPPORT

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

FLX provides the ability to delimit records on paper tape for files that are in formatted binary mode or in formatted ASCII mode.

FILE TRANSFER PROGRAM (FLX)

Formatted binary records are delimited by standard DOS 4-byte headers and a trailing checksum. Formatted ASCII records which do not already end with line feeds or form feeds are delimited by carriage return-line feed pairs.

Special treatment is given to files which normally default to image mode transfers, i.e., TSK, OLB, MLB, and SYS files. On output to paper tape, these files are written, by default, in formatted binary. When read back from paper tape to a Files-11 volume, the file is written with fixed-length, 512-byte records as the default.

These defaults insure, when these files are read back from paper tape, that they will be in exactly the same format as they were before they were punched. However, the new files will not be contiguous unless the user specifies /CO/BL:n with the output file specifier. An appropriate value for n (the number of contiguous blocks to allocate), must be known by the user before issuing the command.

NOTE

The use of explicit transfer mode switches when transferring TSK, OLB, MLB, and SYS files between paper tape and Files-11 volumes can cause files read back in from paper tape to be different from the files that were originally written out.

3.9 FLX SWITCHES

FLX provides three types of switches for file transfers: format mode switches, which specify the format of the file; transfer mode switches which control the mode of transfer (e.g., formatted ASCII mode, formatted binary mode, or image mode); and file control switches, which control such things as the number of blocks to be allocated to the output file, or the output file's UIC, etc. Switch specifications consist of a slash (/), followed by a 2-character switch name, and is optionally followed by a value separated from the switch specifier by a colon (:).

3.9.1 Format Mode Switches

FLX has three format mode switches: /DO (DOS format), /RT (RT-11 format); and /RS (Files-11 format). When specified, these switches describe the format of the specified files. These switches are described in Table 3-1.

FILE TRANSFER PROGRAM (FLX)

Table 3-1
FLX Format Mode Switches

Switch	Description
/DO	<p>Identifies the file as a DOS-11 formatted file.</p> <p style="text-align: center;">NOTE</p> <p>If no /DO, /RT, or /RS switch is specified in a command string, FLX initially assumes /RS for output files and /DO for input files. However, this default transfer direction can be dynamically changed by specifying a command consisting of only the switch desired for the input side (/DO or /RS only). See the note following /RT for the /RT default operation.</p> <p>Example:</p> <p>To specify the default transfer direction from RSX to DOS, type:</p> <p style="padding-left: 40px;"><u>FLX</u>>/RS ↵</p> <p>To specify the default transfer direction from DOS to RSX, type:</p> <p style="padding-left: 40px;"><u>FLX</u>>/DO ↵</p>
/RS	<p>Identifies the file as a Files-11 formatted file.</p> <p style="text-align: center;">NOTE</p> <p>See notes for /DO and /RT switches.</p>
/RT	<p>Identifies the file as an RT-11 formatted file.</p> <p style="text-align: center;">NOTE</p> <p>If the /RT switch is specified on one side of a command string, the default entry for the other side is /RS.</p> <p>Examples:</p> <ol style="list-style-type: none"> 1. <u>FLX</u>>DK0:=DT0:SYS1.MAC/RT ↵ The output is defaulted to /RS. 2. <u>FLX</u>>DK0:/RT=DK0:SYS1.MAC ↵ The input is defaulted to /RS.

FILE TRANSFER PROGRAM (FLX)

3.9.2 Transfer Mode Switches

FLX has three modes of file transfer for conversion in either direction between DOS-11 and Files-11, or between RT-11 and Files-11. These modes are: formatted binary, formatted ASCII, and image mode. When a switch is specified, it determines the transfer mode to be applied during translation. The switch formats and descriptions are listed in Table 3-2 below.

Table 3-2
FLX Transfer Mode Switches

Switch	Description
/FA:n	<p>Formatted ASCII</p> <p>The DOS-11 or RT-11 file is to be formatted ASCII. Formatted ASCII is defined as ASCII data records terminated by carriage return/form feed (CR-FF), form feed (FF), or vertical tab (VT). In transfers from DOS-11 or RT-11 files to Files-11 files, CR-LF pairs are removed from the end of records. In transfers from Files-11 files to DOS-11 or RT-11 files, CR-LF pairs are added to the end of each record which does not already end with LF or FF. In both directions all nulls, rubouts, and vertical tabs (VT) are removed from input records.</p> <p>If n is specified with Files-11 output, fixed-length records of size n are generated. Output records will be padded with nulls, if necessary.</p> <p>If n is not specified with Files-11 output, then variable-length records are generated. The output record size will equal the input record size.</p> <p style="text-align: center;">NOTE</p> <p>ASCII data is transferred as 7-bit values. The eighth bit of each byte is masked off before transfer. CTRL/Z (ASCII 032 octal) is treated as the logical end of input file for formatted ASCII transfers from DOS-11 cassette or paper tape to Files-11.</p>

FILE TRANSFER PROGRAM (FLX)

Table 3-2 (Cont.)
FLX Transfer Mode Switches

Switch	Description								
/FB:n	<p>Formatted Binary</p> <p>The DOS-11 or RT-11 file is to be formatted binary. In formatted binary mode, formatted binary headers and checksums are added to records output to DOS-11 or RT-11 files, and they are removed when transferred to Files-11 files.</p> <p>If n is specified with Files-11 output, then fixed-length records of size n will be output (512 bytes is the maximum). FLX pads records with nulls to create the specified length. If n is not specified with Files-11 output, then variable-length records are produced. The output record size is equal to the input record size.</p>								
/IM:n	<p>Image Mode</p> <p>The transfer is to be in image mode. Image mode forces fixed-length records. The value n can be used to indicate the desired record length for Files-11 output (512 bytes is maximum). If the value n is not specified, a record length of 512 bytes is assumed.</p> <p style="text-align: center;">NOTES</p> <p>1. The following default transfer modes are assumed for these file types (with the exception of paper tape transfers -- see Section 3.8).</p> <table data-bbox="617 1239 1055 1428"> <thead> <tr> <th><u>Mode</u></th> <th><u>File Type</u></th> </tr> </thead> <tbody> <tr> <td>/IM</td> <td>TSK, OLB, MLB, SYS</td> </tr> <tr> <td>/FB</td> <td>OBJ, STB, BIN, LDA</td> </tr> <tr> <td>/FA</td> <td>All others</td> </tr> </tbody> </table> <p>2. If the value n is specified in conjunction with /FA, /FB, or /IM when the output file is not a Files-11 file, it is ignored.</p>	<u>Mode</u>	<u>File Type</u>	/IM	TSK, OLB, MLB, SYS	/FB	OBJ, STB, BIN, LDA	/FA	All others
<u>Mode</u>	<u>File Type</u>								
/IM	TSK, OLB, MLB, SYS								
/FB	OBJ, STB, BIN, LDA								
/FA	All others								

FILE TRANSFER PROGRAM (FLX)

3.9.3 File Control Switches

In addition to the switches associated with the transfer modes and directions, FLX provides switches to control file processing. These switches are described in Table 3-3.

Table 3-3
FLX File Control Switches

Switch	Description
/BL:n	<p>Indicates the number of contiguous blocks to be allocated to the output file where n specifies the number of blocks.</p> <p>This switch is used normally in conjunction with /CO.</p> <p>If /BL is not specified, the input file size is used as the output file size.</p> <p style="text-align: center;">NOTE</p> <p>The file allocation scheme used for RT-11 volumes normally allocates the largest available space on the volume for a new file. Using /BL:n with the /RT switch for the output file causes the output file to be allocated the first <UNUSED> space of size $\geq n$. However, when the RT-11 file is closed, the input file size is used as the output file size. If the input file size is not $\leq n$, an error results. Since all RT-11 files are contiguous, the /CO switch is not required with the /BL:n switch for RT-11 output.</p>
/BS:n	<p>Specifies the block size for cassette tape output, where:</p> <p style="padding-left: 40px;">n specifies the block size in bytes.</p> <p>If /BS is not specified, a block size of 128 is assumed. /BS is only valid in a cassette tape (CT) output file specifier.</p>
/CO	<p>Indicates that the output file is to be contiguous.</p> <p>The /CO switch is meaningful only to disks and DECTape.</p> <p>If the input file is paper tape, cassette or DOS-11 magtape, /BL is also required.</p>

FILE TRANSFER PROGRAM (FLX)

Table 3-3 (Cont.)
FLX File Control Switches

Switch	Description
/DE	<p style="text-align: center;">NOTE</p> <p>The file types .TSK, .SYS, and .OLB are transferred to Files-11 volumes with /CO implied when the input is a FILES-11 volume or a DOS-11 DECTape or disk.</p> <p>Deletes files from a DOS-11 DECTape or disk. It is used also in conjunction with /RT to delete files from an RT-11 DECTape or disk.</p>
	<p style="text-align: center;">NOTE</p> <p>When /DE is specified, the FLX command string has no output specifier.</p>
/DI	<p>Causes a directory listing of cassettes or DOS volumes to be listed on a specified output file. It is used also in conjunction with /RT to generate a directory listing of RT-11 volumes in a specified output file.</p> <p style="text-align: center;">NOTES</p> <ol style="list-style-type: none"> 1. Files-11 volume directories can not be listed using FLX. 2. If no output specifier is present, then the directory is issued to TI:. 3. If no filename is specified, *.* is assumed. <p>Figure 3-3 contains sample DOS volume directory listings, along with a description of the information contained in each field. Figure 3-4 contains sample RT volume directory listings, along with a description of the information contained in each field.</p>

FILE TRANSFER PROGRAM (FLX)

Table 3-3 (Cont.)
FLX File Control Switches

Switch	Description
/FC	Indicates that FORTRAN carriage control conventions are to be used, i.e., FD.FTN is set in the file data block. The default is a carriage return and line feed between records, i.e., FD.CR is set. The /FC switch applies only to Files-11 output files. Refer to the <u>RSX-11 I/O Operations Reference Manual</u> for a discussion of the file data block and record attributes.
/ID	Requests the current version number of FLX. The switch can be part of an output or input specifier or it can be typed in response to the FLX prompt message (FLX>).
/LI	Same as /DI
/NU:n	Used in conjunction with the /ZE and /RT switches to specify the number of directory blocks (n) to allocate when initializing an RT-11 disk or DECTape. If /NU:n is not specified, four directory blocks are allocated. The maximum number of blocks which can be allocated is 37 octal (31 decimal).
/SP	Indicates that the converted file is to be spooled via the print spooler. The /SP switch applies only to Files-11 output files. The print spooler is described in Appendix C.
/UI	Indicates that the output file is to have the same UIC as the input file. The /UI switch is ignored if the output specifier contains an explicit UIC.
/VE	Causes each record written to the cassette to be read and verified. The /VE switch is only valid with a CT output file specifier.
/ZE	Initializes cassettes or DOS-11 volumes. It is used also in conjunction with /RT (and /NU) to initialize RT-11 volumes.
NOTES	
1. For DOS-11 DECTape, /ZE creates an entry for the current UIC.	
2. The /ZE command uses no output specifier.	

FILE TRANSFER PROGRAM (FLX)

DEctape Directory Listing

```

DIRECTORY      DT:[200,200]
19-SEP-74

FLX.TSK        104.      19-SEP-74 <233>
UFD.TSK        8.       19-SEP-74 <233>
TKN.TSK        6.       19-SEP-74 <233>
MOU.TSK        14.      19-SEP-74 <233>
    
```

TOTAL OF 132. BLOCKS IN 4. FILES

Casette Directory Listing

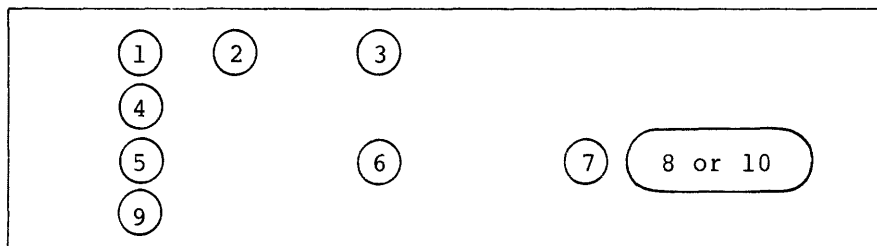
```

DIRECTORY      CT1:[200,200]
19-SEP-74

UFD.TSK;1-0   28.      19-SEP-74 128.
TKN.TSK;1-0   20.      19-SEP-74 128.
MOU.TSK;1-0   52.      19-SEP-74 128.
    
```

TOTAL OF 100. BLOCKS IN 3. FILES

These directories contain similar information. The following key explains what that information is and where it is located.



1. Identifies this as a directory listing.
2. Specifies the device name and unit number.
3. Is the User Identification Code.
4. Is the date the directory was listed.
5. Is the filename, file type, version number (RSX cassettes only), and sequence number (cassettes only).
6. Is the file size in blocks.
7. Is the file creation date.
8. Is the record size in bytes for the file (cassettes only).
9. Is a total of the actual file sizes, and the total number of files in the directoty.
10. Protection code (disk and DEctape only).

Figure 3-3
DOS Directory Listings

FILE TRANSFER PROGRAM (FLX)

DEctape Directory Listing

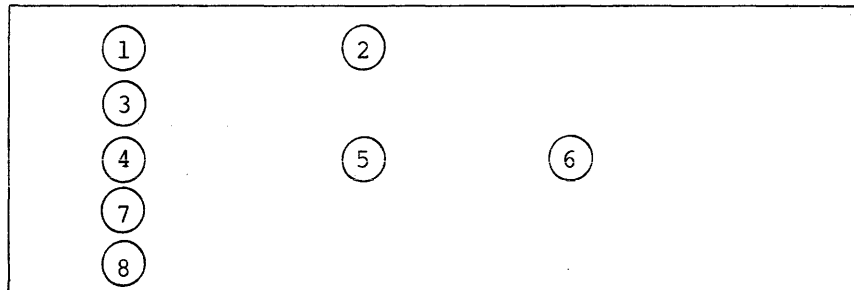
```
DIRECTORY      DK:
 4-JUN-75

SIPBOO.MAC    49.   4-JUN-75
< UNUSED >    6.
SIP  .MAC     10.   4-JUN-75
SIPCD .MAC     7.   4-JUN-75
< UNUSED >    21.
SIPQIO.MAC    7.   4-JUN-75
< UNUSED > 4686.
```

4713. FREE BLOCKS

TOTAL OF 73. BLOCKS IN 4. FILES

The following key explains what the information is, and where it is located.



1. Identifies this as a directory listing.
2. Specifies the device name and unit number.
3. Is the date the directory was listed.
4. Is the filename and file type; or <UNUSED> indicates free (unused) space.
5. Is the number of blocks in the file or free space.
6. Is the file creation date, or blank for free space.
7. Is the total number of free blocks on the volume.
8. Is the total number of blocks allocated to files on the volume.

Figure 3-4
RT Directory Listing

FILE TRANSFER PROGRAM (FLX)

3.10 FLX ERROR MESSAGES

Errors encountered by FLX during processing are reported on the initiating terminal.

The FLX error messages, their descriptions and suggested user actions are described below.

FLX -- BAD LIST FILE SPEC

Description

The user has specified one of the following:

1. More than one output file for an /LI or /DI operation.
2. Wild cards in the output file for an /LI or /DI operation.

Suggested User Action

Reenter the command line correctly.

FLX -- BAD RESPONSE

Description

The user has specified the Z response to the message:

MOUNT NEW CASSETTE (Y, Z (OUTPUT ONLY) OR CR)
FLX>

and the cassette in question is an input volume.

Suggested User Action

Respond with Y or CR after the message has been redisplayed.

FLX -- CAN'T OPEN @ FILE

Description

The specified indirect command file could not be opened for one of the following reasons:

1. The file is protected against access.
2. A problem exists on the physical device (e.g., device cycled down).
3. Volume is not mounted.
4. The specified file directory does not exist.
5. The named file does not exist in the specified directory.
6. The volume is not on-line.

Suggested User Action

Correct the condition and reenter the command line.

FILE TRANSFER PROGRAM (FLX)

FLX -- CO FILES TO OUTPUT DEVICE NOT ALLOWED

Description

The user has used the /CO switch with an illegal output device (e.g., MT, CT, or PP).

Suggested User Action

Reenter the command line without the /CO switch specified.

FLX -- CASSETTE ERROR I/O TERMINATED

Description

An unexpected hardware error has occurred during the end-of-volume sequence on a cassette volume. The transfer is aborted.

Suggested User Action

Reenter the command line using a new cassette.

FLX -- COMMAND SYNTAX ERROR

Description

The user has entered a command in a format that does not conform to syntax rules.

Suggested User Action

Reenter the command line with the correct syntax.

FLX -- CONFLICTING TRANSFER MODES SPECIFIED

Description

The user has specified conflicting transfer mode switches.

Example:

SY:=DT:FOO.OBJ/IM/FB ↙

Suggested User Action

Reenter the command line with only one transfer mode switch specified.

FLX -- DOS OR RT-11 DEVICE NOT VALID FORMAT

Description

The device specified with the /DO switch has an incorrect DOS file structure, or the device specified with the /RT switch has an incorrect RT file structure.

Suggested User Action

Correct the problem, and reenter the command line.

FILE TRANSFER PROGRAM (FLX)

FLX -- DT: UFD FULL

Description

The DECTape directory is full.

Suggested User Action

Clean up the directory by deleting all unnecessary files.

FLX -- END OF VOLUME ON CASSETTE

MOUNT NEW CASSETTE? (Y, Z (OUTPUT ONLY) OR CR)

Description

Physical end-of-tape has been encountered during a cassette transfer. The tape rewinds, and the user is asked to mount the next cassette.

Suggested User Action

See Section 3.7.2.1 if an output transfer is being performed or Section 3.7.2.2 if an input transfer is being performed.

FLX -- ERROR DURING DIRECTORY I/O

Description

One of the following conditions may exist:

1. The volume is not write-enabled.
2. The /DO, /RT or /RS switches were incorrectly specified.
3. The volume is not of the proper format.
4. A hardware error occurred during a directory I/O operation (i.e., bad tape).

Suggested User Action

The following user actions correspond (by number) to the conditions listed above.

1. Write-enable the volume.
2. Respecify the /DO, /DT or /RS switches correctly.
3. No recovery is possible with the volume currently mounted. Mount a volume which is in the proper format, and retry the operation.
4. Retry the operation.

FLX -- FILE ALREADY EXISTS

Description

The user specified an output file which already exists on the device specified.

FILE TRANSFER PROGRAM (FLX)

Suggested User Action

Reenter the file specifier using a new or corrected filename.

FLX -- @ FILE NESTING EXCEEDED

Description

More than one level of indirect files was specified.

Suggested User Action

Retry the operation with only one level of indirect file specified.

FLX -- FILE NOT FOUND

Description

The named file does not appear, as specified, in the requested directory.

Suggested User Action

Retry the operation with the filename and directory correctly specified.

FLX -- WARNING -- INPUT FILE OUT OF SEQUENCE

Description

A cassette multi-volume file is being accessed out of sequence.

Suggested User Action

This is a warning message. The transfer will continue unless terminated by the user.

FLX -- @ FILE SYNTAX ERROR

Description

Syntax error in the indirect file specifier.

Suggested User Action

1. Edit the indirect command file using either EDI or SLP.
2. Rerun FLX using the corrected command file.

FLX -- FMTD ASCII RECORD FORMAT BAD

or

FLX -- FMTD BINARY RECORD FORMAT BAD

Description

Either the file is corrupted, or the file is not of the specified type.

FILE TRANSFER PROGRAM (FLX)

Suggested User Action

If the file is corrupted, there is no recovery possible. If the file type is incorrect, retry the operation with the correct type.

FLX -- ILLEGAL /BS SIZE -- USE 0<N<=512. AND EVEN

Description

An illegal block size was specified with the /BS switch on cassette output.

Suggested User Action

Reenter the command line with a legal block size.

FLX -- INCORRECT # IN/OUT SPECS

Description

The user specified more than one input or output specifier in a command where only one is allowed.

Suggested User Action

Reenter the command line with the proper syntax.

FLX -- INVALID DEVICE

Description

The user specified a device that cannot be utilized as an input or output device, e.g., trying to read from a line printer.

Suggested User Action

Reenter the command line with a legal device specified.

FLX -- INVALID DOS OR RT-11 FILE SPEC

or

FLX -- INVALID RSX FILE SPEC

Description

The file specifier does not conform to proper syntax, or the specified operation could not be performed on the specified device.

Suggested User Action

Reenter the file specifier with the proper syntax.

FLX -- INVALID SWITCH

Description

The user has specified a switch that is not a valid FLX switch or does not conform to proper syntax.

FILE TRANSFER PROGRAM (FLX)

Suggested User Action

Reenter the command line with a correct switch specification.

FLX -- I/O ERROR

Description

One of the following conditions may exist:

1. The specified device is not on-line.
2. A Files-11 volume is not mounted.
3. A hardware error has occurred (e.g., bad tape).

Suggested User Action

1. Ensure that the device is on-line and that the volume is mounted (if it is an Files-11 volume).
2. Reenter the command line.

FLX -- I/O ERROR DELETING LINKED FILE

Description

An uncorrectable error occurred while a DOS linked file was being deleted.

Suggested User Action

No action required; the file is effectively deleted, but the volume may be corrupted.

FLX -- I/O ERROR INITIALIZING DIRECTORY

Description

One of the following conditions may exist:

1. The specified device is not on-line.
2. The specified volume is not mounted.
3. A hardware error has occurred (e.g., bad tape).

Suggested User Action

1. Ensure that the device is on-line and in operable condition.
2. Reenter the command line with the required switch specified.

FLX -- I/O ERROR ON COMMAND INPUT

Description

An unexpected error in command input was encountered from either an indirect command file, or TI;; FLX exits.

Suggested User Action

Restart FLX.

FILE TRANSFER PROGRAM (FLX)

FLX -- I/O ERROR ON FLX TEMPORARY FILE

Description

FLX encountered an error condition with its temporary file. FLX creates a temporary file on SY: for operations involving DOS-11 CT, DT, or MT. This error occurs when:

1. SY: is not on-line and mounted.
2. SY: is write-locked.
3. A protection violation occurred.
4. An I/O error was encountered.

Suggested User Action

Correct the error condition and reenter the command line.

FLX -- I/O ERROR ON LIST FILE

Description

An error occurred on the output device during a /DI or /LI sequence. There is a hardware problem with the output device (e.g., device powered down).

Suggested User Action

1. Rectify the condition.
2. Reenter the command line.

FLX -- OUTPUT DEVICE FULL

Description

The DOS or RT-11 output volume does not contain enough space for the output file.

Suggested User Action

Delete all unnecessary files and reenter the command line.

FLX -- OUTPUT FILE SPEC NOT ALLOWED

Description

The user supplied an output file specifier for a command that does not allow one.

Suggested User Action

Reenter the command without an output file specifier.

FILE TRANSFER PROGRAM (FLX)

FLX -- RECORD TOO LARGE

Description

FLX has detected an input record in a Files-11 transfer that is larger than the specified or implied record size for the file, i.e., the file is corrupted.

Suggested User Action

The file in question is unusable.

FLX -- REQUEST TERMINATED -- LAST BLOCK NOT WRITTEN

Description

The <CR> reply was given by the user to indicate that no new volume would be mounted when an end-of-volume was encountered on cassette output. The block which FLX was attempting to write when it encountered the end of the cassette has not been written.

Suggested User Action

No action is required; the message is purely informational.

FLX -- SPECIFIED RECORD SIZE BAD, 512. USED

Description

The record size specified with the /FA, /FB, or /IM switch is not acceptable. A record size of 512(10) bytes is assumed.

Suggested User Action

This is a warning message; no action is required.

FLX -- UNABLE TO ALLOCATE FILE

Description

There is no available space on the DOS or Files-11 volume for the specified file; the volume is full.

Suggested User Action

Delete all unnecessary files and reenter the command line.

FLX -- UNABLE TO OPEN FILE

Description

A specified input or output Files-11 file could not be opened. Possible reasons are:

1. Input file does not exist.
2. Volume is not mounted.
3. Protection violation occurred.

Suggested User Action

Correct the condition and reenter the command line.

FILE TRANSFER PROGRAM (FLX)

FLX -- UNABLE TO OPEN LIST FILE

Description

The list file cannot be opened under the specified filename and UIC; the specified device may not be a valid Files-11 volume.

Suggested User Action

Reenter the command line specifying the correct filename and UIC.

FLX -- UNDIAGNOSABLE REQUEST

Description

FLX does not recognize the command line syntax.

Suggested User Action

Reenter the command line with the proper syntax.

FLX -- /CO FILES FROM INPUT DEVICE NOT ALLOWED UNLESS BL: SPEC

Description

When transferring files from MT, PR, or CT, the /CO switch can be only specified when the /BL switch is also specified.

Suggested User Action

Reenter the command line, specifying the /BL switch.

FLX -- * IN VERSION NUMBER NOT ALLOWED

Description

A wild card was detected in the version number field of a file specifier.

Suggested User Action

Reenter the command line with all version numbers explicitly specified.

CHAPTER 4
FILE DUMP UTILITY (DMP)

4.1 INTRODUCTION TO DMP

The File Dump utility (DMP) program produces a printed listing of the contents of a file. The listing can be directed to any suitable output device: line printer, terminal, DECTape or disk. DMP runs in either one of two modes:

1. File Mode

In file mode, one input file is specified and all, or a specified range (see /BL:n:m) of virtual blocks, of the named file is dumped.

NOTES

- a. A virtual block refers to a relative block of data in a file.
- b. Virtual blocks are numbered sequentially from 1 through n, where n is the total number of virtual blocks of the file.
- c. The input device must be a Files-11 structured volume and must be mounted via the MCR MOUNT command.

2. Device Mode

In device mode, only the device is specified, and a specified range (/BL:n:m) of logical blocks is dumped.

NOTES

- a. /BL:n:m switch is a required parameter.
- b. A logical block refers to the actual 512-byte block on disk and DECTape, and

FILE DUMP UTILITY (DMP)

physical records on magtape and cassette. DMP will handle physical records up to 2048 bytes in length.

- c. Logical blocks are numbered from 0 to n-1, where n is the total number of logical blocks on the device.
- d. The volume to be dumped must not be mounted.

4.2 INITIATING DMP

All RSX-11M utility programs can be initiated in several ways. These methods are described in Section 1.2. The methods for DMP are:

```
>DMP ↵  
>DMP command string ↵  
>RUN ...DMP ↵  
>RUN ...DMP/UIC=[group,member] ↵  
>RUN $DMP ↵  
>RUN $DMP/UIC=[group,member] ↵
```

4.3 DMP COMMAND STRINGS

Commands to DMP are expressed in the following format:

outfile=infile/switch

For a complete description of file specifiers, see Section 1.3.

4.4 DMP SWITCHES

DMP switch specifications consist of a slash (/) followed by a 2-character switch name, optionally followed by a value, which is separated from the switch by a colon (:). Eight switches are recognized by DMP. These switches are described in Table 4-1.

FILE DUMP UTILITY (DMP)

Table 4-1
DMP Switches

Switch	Description
Default	Word mode octal dump
/AS	<p>The /AS switch specifies that the data should be dumped in ASCII mode. The control characters (0-37) are printed as ↑, followed by the alphabetic character corresponding to the character code +100. For example, bell (code 7) is printed as ↑G (code 107). Lower case characters (140-177) are printed as %, followed by the corresponding upper case character (character code -40).</p>
/BA:n:m	<p>This switch allows the user to specify a 2-word base block address, where n = high-order base block address (octal), and m = low-order base block address (octal). When specified, all future block numbers will be added to this value to obtain an effective block number. This switch is useful to specify block numbers that exceed 16 bits. For example:</p> <pre>DMP>/BA:1:0 ↵</pre> <p>Specifies that all future block numbers will be relative to 65536(10) (200000(8)).</p> <pre>DMP>/BA:0:0 ↵</pre> <p>Clears the base address.</p>
/BL:n:m	<p>Specifies the range of blocks to be dumped, where n is the first block and m is the last block.</p> <p style="text-align: center;">NOTES</p> <ol style="list-style-type: none"> 1. If the /BL:n:m switch is specified in file mode, it specifies the range of virtual blocks to be dumped. 2. If the /BL:n:m switch is specified as /BL:0 in file mode, no virtual blocks are dumped. This is useful when the user wishes to dump only the header portion of the file (see /HD). 3. The /BL:n:m switch is a required parameter in device mode. When used in device mode, it specifies the range of logical blocks to be dumped.

FILE DUMP UTILITY (DMP)

Table 4-1 (Cont.)
DMP Switches

Switch	Description
/BY	The /BY switch specifies that the data should be dumped in byte octal format.
/HD	<p>This switch is an optional parameter to be used in file mode. If specified, /HD causes the file header as well as the specified portion of the file to be dumped.</p> <p style="text-align: center;">NOTE</p> <p>If just the header portion of the file is desired, the user can specify /HD/BL:0. The file header is described in Appendix F of the <u>RSX-11 I/O Operations Reference Manual</u>.</p>
/ID	<p>Causes DMP's version to be identified. This switch may be specified on a line by itself at any time.</p> <p>Example:</p> <pre style="margin-left: 40px;">>DMP /ID ↵</pre>
/LB	<p>Logical block. This switch gives the user only the starting block number and a contiguous or noncontiguous indication for the file.</p> <p>Example:</p> <pre style="margin-left: 40px;">DMP>TI:=DK0:RICKSFILE.DAT;3/LB ↵ STARTING BLOCK NUMBER = 0,135163 C</pre> <p>File RICKSFILE.DAT, version 3 is a contiguous file starting at block number 0,135163. (See /BA:n:m for block number description.)</p>
/MD[:n]	<p>Memory dump. This switch allows control of line numbers. Line numbers are normally reset to zero whenever a block boundary is crossed. The /MD[:n] switch allows lines to be numbered sequentially for the full extent of the file, i.e., the line numbers are not reset when block boundaries are crossed. The optional value (:n) allows the user to specify the value of the first line number. The default is 0.</p> <p style="text-align: center;">NOTE</p> <p>Sample listings are presented in Appendix B.4.</p>

FILE DUMP UTILITY (DMP)

4.5 DMP ERROR MESSAGES

DMP -- BAD DEVICE NAME

Description

The user has specified an invalid device name in a file specifier.

Suggested User Action

Reenter the command line specifying the correct device.

DMP -- BLOCK SWITCH REQUIRED IN LOGICAL BLOCK MODE

Description

Self-explanatory -- /BL switch must be specified.

Suggested User Action

Reenter the command line with the /BL switch specified.

DMP -- CANNOT FIND INPUT FILE

Description

The requested file cannot be located in the specified directory.

Suggested User Action

Reenter the command with the correct filename and UIC specified.

DMP -- COMMAND SYNTAX ERROR

Description

The user has entered a command in a format that does not conform to syntax rules.

Suggested User Action

Reenter the command line with the correct syntax.

DMP -- FAILED TO ASSIGN LUN

Description

The user has specified an illegal device in a file specifier.

Suggested User Action

Reenter the command line with the correct device specified.

FILE DUMP UTILITY (DMP)

DMP -- FAILED TO READ ATTRIBUTES

Description

The user has attempted to access a file for which he does not have read access privileges.

Suggested User Action

Rerun DMP using a UIC which has read access privileges to the file.

DMP -- ILLEGAL SWITCH

Description

The user has specified a switch that is not a valid DMP switch or used a legal switch in an invalid manner.

Suggested User Action

Reenter the command line with the correct switch specified.

DMP -- I/O ERROR ON INPUT FILE

or

DMP -- I/O ERROR ON OUTPUT FILE

Description

One of the following conditions exists:

1. A problem exists on the physical device (e.g., device cycled down).
2. File is corrupted or the format is incorrect.
3. Output volume is full.

Suggested User Action

1. Determine which of the above conditions may exist.
2. Rectify the condition.
3. Reenter the command line.

FILE DUMP UTILITY (DMP)

DMP -- NO INPUT FILE SPECIFIED

Description

The user has terminated a command without entering an input file specifier.

Suggested User Action

Reenter the command line with an input file specified.

DMP -- NO LISTS OR WILD CARDS ALLOWED

Description

The user either entered a command with more than one input or output filename or entered a wild card in a file specifier.

Suggested User Action

Reenter the command line with only one input file specifier and one output file specifier. No wild card specifiers allowed.

DMP -- OPEN FAILURE ON INDIRECT FILE

Description

The requested indirect command file does not exist as specified. One of the following conditions may exist:

1. The file is protected against access.
2. A problem exists on the physical device (e.g., device cycled down).
3. The volume is not mounted.
4. The specified file directory does not exist.
5. The named file does not exist in the specified directory.

Suggested User Action

1. Determine which of the above conditions may exist.
2. Rectify the condition.
3. Reenter the command line.

FILE DUMP UTILITY (DMP)

DMP -- OPEN FAILURE ON INPUT FILE

or

DMP -- OPEN FAILURE ON OUTPUT FILE

Description

One of the following conditions may exist:

1. The file is protected against access.
2. The named file does not exist in the specified directory.
3. The volume is not mounted.
4. The specified file directory does not exist.
5. A problem exists on the physical device (e.g., device cycled down).

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

CHAPTER 5

LINE TEXT EDITOR (EDI)

5.1 INTRODUCTION

The Line Text Editor (EDI) is an interactive context-editing program that provides the capability to create and modify source programs and other ASCII text material. EDI may be directed to read a line, or group of lines, from the input file into an internal buffer by means of terminal commands. The user can then, by means of additional commands, examine, delete and change text, and insert new text at any point in the buffer. When the line or block of lines has been edited, the user can issue a command to write the data into a new file.

EDI is most frequently used to modify MACRO and FORTRAN source programs, but it can also be used to edit any ASCII text material.

The EDI commands are described in sections 5.4 and 5.5.

5.2 USING EDI

This section is designed to give the user a step-by-step approach to using the RSX-11M Line Text Editor.

5.2.1 Preparing to Run EDI

Before initiating EDI, the user must consider the following:

1. EDI can operate only on Files-11 format files. All other file formats are rejected.
2. 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.

Example:

If a user has a file on DECTape and he wants to edit that file and store the resulting file on disk, he must:

- a. Transfer the file to disk and perform the editing there.
- b. Edit the file on DECTape and then transfer the file to disk using PIP.

LINE TEXT EDITOR (EDI)

3. If a device other than SY0: is to be utilized, it must be mounted via the MCR MOUNT command.
4. If other than the latest version of a file is to be edited, the desired version number can be explicitly stated in the file specifier. This file will be opened as the input file and the version number of the output file will be one greater than the latest version of the file. If a command issued during the editing session closes the explicit version of the file without terminating the session (e.g., a TOF command), both output and input files are closed, and the latest version of the file is reopened as the input file.

5.2.2 Initiating EDI

All RSX-11M utility programs can be initiated in several ways. These methods are described in Section 1.2. The methods for EDI are:

```
>EDI ↵  
>EDI command string ↵  
>RUN ...EDI ↵  
>RUN ...EDI/UIC=[group,member] ↵  
>RUN $EDI ↵  
>RUN $EDI/UIC=[group,member] ↵
```

If any format except ">EDI command string" is used, EDI issues the following prompting message:

```
EDI>
```

At this point, the user must enter a file specifier for the file to be edited. The file specifier is in the following format:

```
dev:[uic]filename.typ
```

If the file specifier is a new file (i.e., the file specified cannot be found on the specified device), the assumption is that the user wishes to create a new file with the given filename. EDI then prints the following comment lines:

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

and enters Input mode.

NOTES

1. Edit control modes are described in Section 5.2.3.
2. If the message "FILE DOES NOT EXIST" is displayed, it means that the specified user file directory is nonexistent.

LINE TEXT EDITOR (EDI)

3. EDI does not accept indirect command file specifiers.
4. The abbreviation "filespec" is used in the command formats to indicate file specifier.

If an existing filename is specified, EDI prints:

[PAGE 1]

*

and waits in Edit mode for the first command to be issued.

If the ">EDI command string" format is used, the prompt message (EDI>) is not issued, and EDI starts up in either Input or Edit mode, depending on the filename specified: Input mode if the filename is new; Edit mode if the filename already exists.

NOTE

At program startup, after the input file has been identified and the output file has been created, the program is ready for commands. The first line available to the user for editing is always one line above the top of the input file or the block buffer. This allows for inserting text at the beginning of the input file or the block buffer. If, however, the user wishes to manipulate the first line of text, he must perform a NEXT operation to make that line available.

5.2.2.1 Defaults in File Specifiers - If any of the elements of the file specifier, except filename and type for input file, are omitted, EDI uses a default. The default values for both the input and the output files are listed in Table 5-1.

LINE TEXT EDITOR (EDI)

Table 5-1
EDI Default File Specifiers

Element	Default Value for Input File	Default Value for Output File
dev:	SY0:	Same as input device
[uic]	UIC under which EDI is currently running	Same as input [uic]
filename	No default--must be specified	Same as input filename
.typ	No default--must be specified	Same as input file type
;version	Latest version	Latest version+1

5.2.3 EDI Control Modes

EDI is capable of operating in two control modes:

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

Edit mode is invoked automatically at program startup, if an existing file is being edited.

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

Input mode is invoked automatically at program startup if a non-existent file is specified. When Input mode is active, lines entered at the terminal are treated as text to be inserted into the output file.

5.2.4 Changing Control Mode

If EDI is in Edit mode and the user wishes to enter Input mode, the INSERT command is issued, followed by a carriage return. This will place EDI in Input mode, and all lines entered from this point will be added into the file as new text, following the current line.

NOTE

The INSERT command is described in section 5.4.3.8.

LINE TEXT EDITOR (EDI)

If EDI is in Input mode and the user wishes to switch to Edit mode, a carriage return is entered as the first character in a line. EDI will then issue the prompting character * , which signifies that the Edit mode is active.

5.2.5 Text Access Modes

EDI provides the user with two modes of accessing and manipulating lines of text in the input file:

- Line-by-line Mode - Allows the user to access lines of text one line at a time (a line is a string of characters terminated by a carriage return).
- Block Mode - Allows the user to access a block of lines, on a line-by-line basis.

NOTE

Block mode is the default text access mode.

5.2.5.1 Line-by-Line Mode - In this mode, a single line is the unit of the input file available to the user for modification at any point. Line-by-line mode is entered by issuing a BLOCK OFF command, and is terminated by issuing a BLOCK ON command. The BLOCK ON/OFF command is described in section 5.5.1.1.

The line currently available is specified by a pointer, which can be thought of as moving sequentially through the file, starting just before the first line in the file. The user can manipulate the line pointer by using the editing commands which are described in sections 5.4 and 5.5.

When a file is opened at the beginning of an editing session, the first line of that file can be brought into memory and made available for modification. This line remains in memory until the user requests that a new line be brought in. The pointer then moves down the file until the line 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 old, or previous, line is written into the output file, and is no longer accessible unless the user issues a TOP command (the TOP command is described in Section 5.5.3.10). The TOF command can be used to move the line pointer to the top of the file; however TOF always causes EDI to re-enter the block mode (see Section 5.4.3.15).

5.2.5.2 Block Mode - In this mode, a user-specified portion (80 lines is the default) of the input file is held in the block buffer for editing until the user requests that the contents of the buffer be added to the output file.

LINE TEXT EDITOR (EDI)

When the user is operating in block mode, EDI executes commands only with respect to that portion of the input file currently in the buffer. The lines of text in the buffer can be addressed backward as well as forward within the buffer, thus allowing the user to back up to a previously edited line without having to process the entire block or file all over again, or having to issue a TOF command.

5.2.5.3 Line-by-Line Vs. Block Mode - Table 5-2 provides a brief summary of the differences between line-by-line and block mode.

Table 5-2
Line-by-Line vs. Block Mode

Line-by-Line Mode	Block Mode
<p>One line available for modification at a time (see third statement below for exception).</p> <p>Lines can only be accessed forward through the file.</p> <p>Locative commands, those which allow the user to locate a string of text for modification, can be applied to search the entire file.</p>	<p>Entire block of lines available for modification at a time, on a line-by-line basis.</p> <p>Lines can be accessed forward and backward within a block.</p> <p>Locative commands search only the block that is in memory. To search more data, another block must be read in.</p>
<p style="text-align: center;">NOTE</p> <p style="text-align: center;">The line pointer, regardless of editing mode utilized, always points to the first character in the line.</p>	

5.2.6 Text Files

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

LINE TEXT EDITOR (EDI)

5.2.6.1 Input and Secondary Files - EDI accepts input from:

1. The input terminal (i.e., commands and text entries).
2. Files-11 volumes which contain
 - a. The file to be edited; or
 - b. A secondary file; or
 - c. A save file; or
 - d. A macro file.

The input file is always preserved. To delete the input file, the user can use the CLOSE-AND-DELETE command or the EXIT-AND-DELETE command, or PIP can be utilized. 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. The output file is not completely destroyed; it is a truncated version of the input file containing all of the edits to the point of failure.

5.2.6.2 Output Files - The output device is defaulted to the input device, as well as the same directory and filename, with the version number incremented by one. If the user wishes to change any of the file parameters (except device and directory), he can specify a completely new file specifier when closing a file or exiting at the end of an EDI session.

5.2.7 Terminal Conventions

5.2.7.1 Carriage Return - The carriage return has the following effects, depending on how it is used:

1. When issued in place of an input file specifier, carriage return causes EDI to terminate.
2. When issued in Edit mode, carriage return causes the next line to be printed and that line to be the current line.
3. When issued in Input mode as the first character in the line, carriage return causes a return to Edit mode.
4. When issued alone after an INSERT command, carriage return puts the user in Input mode.

5.2.7.2 Character Erase (RUBOUT) and Line Delete (CTRL U) - Two terminal commands are provided which permit the deletion (erasure) of individual characters in a line or the deletion of an entire line. RUBOUT deletes individual characters. CTRL/U deletes the current input line. For a complete description of these commands, see the RSX-11M Operator's Procedures Manual.

During editorial operations, neither the RUBOUT nor CTRL/U command affects previously prepared text.

LINE TEXT EDITOR (EDI)

5.2.8 EDI Command Conventions

5.2.8.1 Use of * - The asterisk character, *, can be used in place of a numeric argument and is equal to 32767(10).

Example

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

```
PRINT *
```

5.2.8.2 Search String Constants - In a number of the EDI commands, the user must identify a string(s) of characters to be located and/or changed. To reduce the necessary terminal entries, the more advanced user can utilize the following special string constructs. In these special cases, the three periods (...) are used to represent any number of intervening characters.

- Case 1. string-1... string-2 - Any string which starts with string-1, continues with any number of intervening characters, and ends with string-2.
- Case 2. ..."string" - Any string which starts at the beginning of the current line and ends with "string".
- Case 3. "string"... - Any string which starts with "string" and ends at the end of the current line.
- Case 4. ... - A string which is the current line.

Examples

THIS IS A SAMPLE OF SPECIAL STRING CONSTRUCTS. Using the preceding sentence and the commands specified in each case, observe the results:

- Case 1. C /S A...E O/S AN EXAMPLE O
results in
THIS IS AN EXAMPLE OF SPECIAL STRING CONSTRUCTS.
- Case 2. C /...SPEC/THIS IS AN EXAMPLE OF SPEC
results in
THIS IS AN EXAMPLE OF SPECIAL STRING CONSTRUCTS.
- Case 3. C /STRING.../EDI STRING CONSTRUCTS.
results in
THIS IS A SAMPLE OF SPECIAL EDI STRING CONSTRUCTS.
- Case 4. C /.../EXAMPLES OF SPECIAL EDI CONSTRUCTS.
results in
EXAMPLES OF SPECIAL EDI CONSTRUCTS.

5.3 EDI ERROR REPORTING

Errors encountered by EDI are reported to the user by appropriate error messages displayed on the input terminal. For the purpose of clarification, these messages (and their descriptions and suggested user actions) are described as four separate classes of errors. These classes are:

- Command level informational error messages
- File access warning messages
- Error messages requiring EDI restart
- Fatal error messages

Input and output files are affected differently by the occurrence of errors. Input files are always preserved. The resulting output file is a truncated version of the input file, containing all edits completed up to the time of the error.

5.3.1 Command Level Informational and Error Messages

Messages in this class indicate information that is designed to be helpful to the user or identify errors that were encountered in the previous command. All messages in this class are enclosed within square brackets and followed by a prompt for a new command. These messages are described in Section 5.6.1.

5.3.2 File Access Warning Messages

File access warning messages represent attempts on the part of the user to access directories, files, or devices that are not present in the host system or which are protected against access. Each message is prefixed with:

EDI --

and, after the message is displayed, EDI returns to command level and requests input by issuing an asterisk. These messages are described in Section 5.6.2.

5.3.3 Error Messages Requiring EDI Restart

The error messages that result in restarting the editing session are caused by conditions encountered by EDI that make it impossible to continue the current editing session. EDI closes all open files (with the exception of the secondary input file), reinitializes, and then prompts for the next file to be edited. Each message is prefixed with:

EDI --

These messages are described in Section 5.6.3.

5.3.4 Fatal Error Messages

Fatal error messages represent system and/or hardware error conditions which make it impossible for EDI to continue execution. All files are closed and EDI terminates its execution. Each error message is prefixed with:

EDI --

and followed by the exit message:

[EXIT]

on the next line. These messages are described in Section 5.6.4.

5.4 BASIC EDI OPERATION AND COMMANDS

EDI can be used to create new files, enter new text into existing files, and edit new and existing files. These operations are described in this section, along with the basic EDI commands which are required by the new user.

5.4.1 Basic EDI Operations

5.4.1.1 Creating a File - To create a file using EDI, a nonexistent filename is used as the file specifier. This causes EDI to print the following lines on the user terminal:

[CREATING NEW FILE]
INPUT

and to enter Input mode. The user then types the desired text on the terminal, observing the required spacing within the line. When the typing of the line is complete, the line is terminated by a carriage return. The next line is entered in an identical manner.

During file creation, input errors on lines which have been terminated by a carriage return are corrected by switching to Edit mode. This is accomplished by typing a carriage return as the first character in a line. Once Edit mode is entered, EDI issues a prompt (*) to indicate that it is ready to receive an edit command. If additional text is to be inserted after the corrections have been completed, the user must locate the text line preceding the desired entry point and return EDI to the Input mode by typing an I, followed by a carriage return. The user can switch between the Input and Edit modes, as required, by following the procedures described in this paragraph.

When the file creation and necessary corrections have been completed, the user must switch to Edit mode to exit from EDI. The output file is stored with the file specifier used when EDI was initiated, or it can be renamed with the EXIT command.

LINE TEXT EDITOR (EDI)

5.4.1.2 Entering Text Into a File - Text can be entered into a file in either Input or Edit mode. Since there are differences between the two modes of text entry, each is described separately:

- Entering Text in Input Mode - When EDI is in Input mode, the information typed on the terminal is inserted in the line following the current line. As each line is terminated by a carriage return, the line pointer is moved down one line and the line which was just entered becomes the current line. EDI does not recognize any command and remains in Input mode until a carriage return is typed as the first character of a line. Then EDI switches to Edit mode.
- Entering Text in Edit Mode - When EDI is in Edit mode, text can be appended to the current line (via an ADD or ADD & PRINT command), inserted as a line following the current line (via an INSERT command), or used to replace the current line (via a RETYPE command). Each of the commands inserts only a single line of text; the command must be reissued to insert another line. Other EDI commands can be used to modify a single line or a complete file, but the ADD, ADD & PRINT, INSERT, and RETYPE commands are mainly the commands used to insert text in Edit mode.

5.4.2 Editing a File

Editing operations are performed only when in Edit mode. The commands available with EDI are categorized as follows:

- Setup commands select data modes, select and open files, select operating conditions, etc.
- Input/Output commands transfer text from input files and to output files.
- Locative commands control the positioning of the current line pointer.
- Text modification and manipulation commands display, change and modify the text.
- Close operation commands terminate editing operations.

The various commands are described in Sections 5.4.3 and 5.5.

Two text access modes are available with EDI, and a command is available that allows the user to select either mode. The line-by-line mode allows the user to locate a line anywhere within a file, but the current line pointer must always be moved down through the file being edited. Once a line is passed, the pointer must be moved to the beginning of the file to access that line again.

LINE TEXT EDITOR (EDI)

The second text access mode, which is the more frequently used of the two modes, is block mode. In block mode, a block of data is stored in the block buffer, and the line pointer can be moved up or down through the data block. The default size of the data block is 80 lines. Since data is packed in the block buffer, the user can increase or decrease the data block size, as required, or he can read additional data blocks into the block buffer as long as the buffer capacity is not exceeded. Once the contents of the block buffer are established, only the contents of the block buffer are available for editing operations. If another block is to be edited, the block buffer must be renewed before new data can be edited. Because of this limitation in block mode, commands have been included in EDI that allow the user to specify a string or page number not in the present contents of the block buffer. Then, EDI performs the necessary search and renew operations to locate the desired data and load it into the block buffer. After selecting edit mode, the user moves the line pointer to locate the line to be changed. The user then can perform editing operations, which can consist of adding, changing, modifying or deleting data, with a wide variety of commands. If a common change is required more than once in a line or to more than one line, commands are available to perform these operations without requiring the user to locate each occurrence and enter the required command. Storage for up to three EDI macro definitions is available so that frequently-used EDI commands or string of EDI commands can be stored, called, and executed, when required.

A single EDI command can insert the contents of a save file into the file or data block being edited. Another command can delete a number of lines and place EDI in Input mode so new lines can be entered via the terminal in place of the original lines. Storage capacity of the block buffer must never be exceeded; surpassing buffer capacity during an operation is considered an EDI error condition. The data which caused the overflow will be deleted.

Commands are included in EDI to perform the following input/output operations:

- Write the contents of the block buffer into the output file and renew the contents of the block buffer with the next input data block.
- Write the contents of the block buffer into the output file and erase the present contents of the block buffer.
- Read the next block or group of blocks into the block buffer.
- Write edited text into the output file and return to top of file.

In all cases, the input/output commands have no effect on the input file. The input file can be deleted only during closing operations. In most cases, an editing session is terminated by transferring the contents of the block buffer and the remaining lines in the input file, in that order, to the output file, and then closing the files. Whether EDI is terminated is dependent upon which closing command is used. The available closing commands are as follows:

LINE TEXT EDITOR (EDI)

- Close the current editing session and remain in EDI.
- Close the current editing session, delete the input file and remain in EDI.
- Exit EDI without deleting the input file.
- Exit EDI and delete the input file.
- Remain in EDI but delete both output and input files.

EDI is a versatile editing tool which enables the user to edit all types of text files. In the following descriptions of the commands, examples are included using text lines and files to illustrate the function of the various commands. Text has been used in these examples so that the intent of the command is not lost in the syntax of a language which the new user may not understand.

5.4.3 Basic EDI Commands

The basic EDI commands listed in Table 5-3 allow the user to create a file, modify a file by adding, deleting, or changing its contents, and exit after the desired operations have been completed. These commands are the most important commands to the new user; therefore, they have been presented as a group near the beginning of this section to provide the new user with an understanding of some of the more important EDI capabilities. As the user becomes familiar with EDI operations, the additional commands described in Section 5.5 will allow utilization of the full EDI capabilities.

LINE TEXT EDITOR (EDI)

Table 5-3
Basic EDI Commands

Command	Command Format	Description
ADD	A[DD] (string)	Append (string) to current line.
ADD & PRINT	AP (string)	Append (string) to current line and print resultant line.
CHANGE	[n]C[HANGE] /string-1/ string-2/	Replace string-1 with string-2 n times in the current line.
CTRL Z	↑Z	Close files and terminate editing session.
DELETE	D[ELETE] [n] or D[ELETE] [-n]	Delete current line and n-1 lines if n is (+); delete n lines preceding current line if n is (-). [-n] operates in block mode only.
DELETE & PRINT	DP [n] or DP [-n]	Same as DELETE except new current line is printed.
EXIT	EX[IT] [filespec]	Close files, name output file and terminate editing session.
INSERT	I[NSERT] (string)	Enter (string) following current line or enter input mode if (string) is not specified.
LOCATE	[n]L[OCATE] (string)	Locate "nth" occurrence of string.
NEXT	N[EXT] [n] or N[EXT] [-n]	Establish new current line n lines away from current line.
NEXT & PRINT	NP [n] or NP [-n]	Establish and print new current line.
PRINT	P[RINT] [n]	Print current line and the next n-1 lines. The last printed line is the new current line.
RENEW	REN[EW] [n]	Write current block to output file and read new block from input file (block mode only).
RETYPE	R[ETYPE] (string)	Replace current line with string; or delete current line if (string) is null.
TOP OF FILE	TOF	Return to top of input file and save all pages previously edited.

LINE TEXT EDITOR (EDI)

5.4.3.1 ADD Command

Function

This command causes the specified string to be appended to the current line.

Format

A[DD] (string)

Example

The following command completes the line HAPPY DAYS ARE HERE

```
*A AGAIN.↵
```

5.4.3.2 ADD AND PRINT Command

Function

This command performs the same function as the ADD command, except that the resultant line is printed.

Format

AP (string)

Example

Using the same line as the ADD command, the following command causes the new line to be printed as follows:

```
*AP AGAIN.↵
```

```
HAPPY DAYS ARE HERE AGAIN.
```

5.4.3.3 CHANGE Command

Function

This command searches for string-1 in the current line and, if found, replaces it with string-2. If string-1 is given, but cannot be located in the current line, EDI prints [NO MATCH] and returns an * prompt. If string-1 is null (not given), string-2 is inserted at the beginning of the line. If string-2 is null, string-1 is deleted from the current line. The search for string-1 begins at the beginning of the current line and proceeds across the line until a match is found. The delimiters may be any matching characters which are not contained in the specified string. Slashes are shown in the example. The first character following the command is considered the beginning delimiter and the next matching character ends the string. Thus, characters used as delimiters must not appear in the string itself. The closing delimiter is optional.

A numeric value "n" preceding the command results in the first "n" occurrences of string-1 being changed to string-2. For each

LINE TEXT EDITOR (EDI)

replacement of string-1 with string-2, the entire line is rescanned beginning at the first character in the line. This allows the user to generate a string of `n` characters as shown in the example below.

If no match occurs, a [NO MATCH] message is displayed.

Format

```
[n]C[HANGE] /string-1/string-2[/]
```

Example

If a line contained A;B;C;D, then the command `4C/;/;` generates A; ; ; ; B;C;D.

5.4.3.4 CTRL/Z Command

Function

Typing CTRL/Z (holding the CTRL key down while typing the letter Z) terminates the editing session. If an output file is open when CTRL/Z is typed, then 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 if EDI is prompting for command input with an asterisk or is in Input mode. If EDI is prompting for another file specifier when CTRL Z is entered, all files are closed (including any open secondary input file), and EDI exits.

5.4.3.5 DELETE Command

Function

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

1. If `n` is given as `+n`, the current line and `n-1` lines following the current line are deleted. The new line available for modification (the new current line) is the line following the last deleted line.
2. If `n` is given as `-n`, the current line is not deleted, but the specified number of lines that precede it are deleted. The line pointer remains unchanged.
3. If `n` is null, the current line is deleted, and the next line becomes the new current line.

NOTES

1. A negative value for `n` can be used only in block mode.
2. If `n` is not specified, a value of `+1` is assumed.

LINE TEXT EDITOR (EDI)

Format

D[ELETE] [n]

or

D[ELETE] [-n]

Example

To delete the previous five lines in the block buffer, the following command is typed:

```
*D -5 ↵
```

5.4.3.6 DELETE AND PRINT Command

Function

This command performs the same function as the DELETE command, except that the new current line is printed when the deletion of all lines has been completed.

Format

DP [n]

or

DP [-n]

NOTES

1. If n is not specified, +1 is assumed.
2. A negative value for n can be used only in block mode.

Example

If the following lines are contained in a file:

THIS IS LINE 1

THIS IS LINE 2

THIS IS LINE 3

THIS IS LINE 4

and the line pointer is at the first line, the following command obtains the results shown below:

```
*DP 2 ↵
```

THIS IS LINE 3

LINE TEXT EDITOR (EDI)

5.4.3.7 EXIT Command

Function

This command transfers all remaining lines in the block buffer and input file (in that order) into the output file, closes the files, and terminates the editing session. If file specifier is used, the output file is renamed to the specified filename.

Format

EX[IT] [filespec]

Example

The following command terminates the editing session, without renaming the output file, and causes the following printout:

```
*EX ↵  
[EXIT]  
Σ
```

5.4.3.8 INSERT Command

Function

This command inserts "string" immediately following the current line. The string becomes the new current line. If "string" is not specified, the user enters Input mode.

Format

I(NSERT) (string)

Example

*I TEXT INSERT IN EDIT MODE ↵	Inserts a line of text immediately after the current line.
*L ABC ↵	Locates a line containing ABC.
<u>ABC IS THE START OF THE ALPHABET</u>	This is the line found.
*I ↵	An I followed by a carriage return causes EDI to switch
TEXT INSERT 1 IN INPUT MODE ↵	to Input mode and a series
TEXT INSERT 2 IN INPUT MODE ↵	of new lines can be input
ETC. ↵	following the current line.
↵	
* -	A carriage return as the first character in a line causes EDI to return to Edit mode and prompt for a new command.

LINE TEXT EDITOR (EDI)

5.4.3.9 LOCATE Command

Function

This command causes a search of the block buffer or input file, beginning at the line following the current line for "string", which may occur anywhere in the line sought. If "string" is not specified, the line following the current line is considered a match. A numeric value "n" preceding the command results in locating the "nth" occurrence of "string". The line pointer is positioned to the line containing the located string. LOCATE applies to the block buffer if EDI is in block mode and to the input file if in line-by-line mode. When the line is located, it is printed, unless a VERIFY OFF command is in effect.

Format

[n]L[OCATE] (string)

Example

The following command can be used to locate the line HAPPY DAYS ARE HERE AGAIN.

```
*L PPY ↵
```

The file or block buffer is checked, and the line is printed when it is located, if the VERIFY ON command is in effect.

5.4.3.10 NEXT Command

Function

This command establishes a new current line at n lines, plus or minus, from the current line.

Format

N[EXT] [n]

or

N[EXT] [-n]

NOTES

1. If n is not specified, a value of +1 is assumed.
2. A negative n can be used only in the block mode.

Example

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

```
*N -5 ↵
```

LINE TEXT EDITOR (EDI)

5.4.3.11 NEXT PRINT Command

Function

Same as NEXT command, except the new current line is printed.

Format

NP [n]

or

NP [-n]

NOTES

The following conventions can be used in place of issuing a complete NP command.

1. Pressing the <CR> key is the same as an NP+1 command.
2. Pressing the <ALTMODE> (or ESCape) key while in the block mode is the same as an NP-1 command.
3. If n is not specified, then a value of +1 is assumed.

Example

Assume the following four lines are contained in the file and the line pointer is at the first line.

LINE 1 OF THE FILE

LINE 2 OF THE FILE

LINE 3 OF THE FILE

LINE 4 OF THE FILE

If the following command is issued, EDI would return the following printout:

*NP 2 ↵

LINE 3 OF THE FILE

5.4.3.12 PRINT Command

Function

This command prints out the current line and the next n-1 lines on the terminal; the last line printed becomes the new current line. (Compare with the TYPE command, Section 5.5.4.13.)

LINE TEXT EDITOR (EDI)

Format

P[RINT] [n]

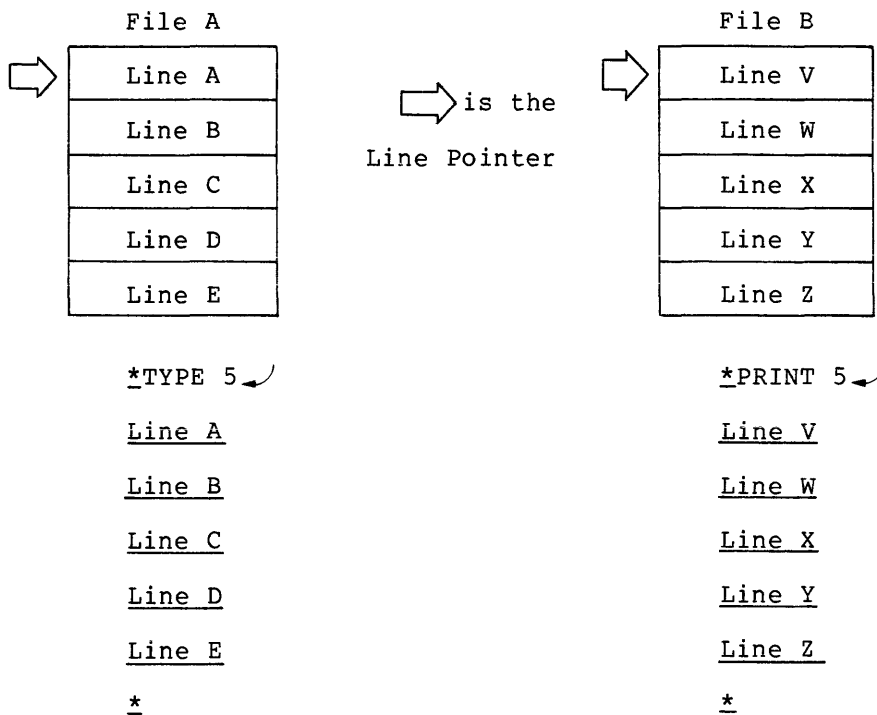
NOTE

If n is not specified, a value of +1 is assumed.

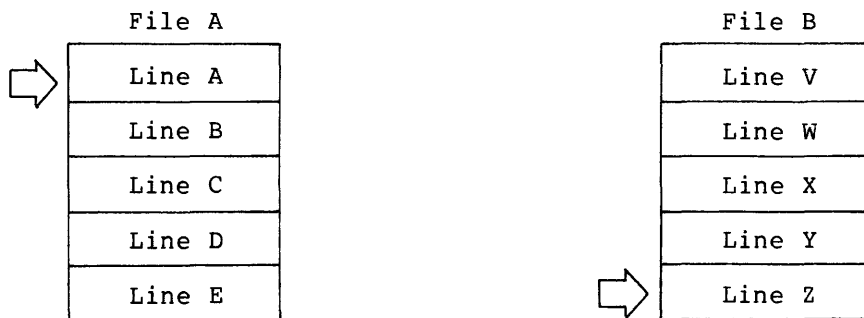
Example

The following example illustrates both the PRINT and the TYPE commands:

Before



After



5.4.3.13 RENEW CommandFunction

This command writes the current block buffer into the output file and reads a new block from the input file. The value n, which is optional, specifies how many times the RENEW command is to be repeated. If n is specified, the process is repeated n times, the inter-blocks are written into the output file, and the last block is left in the block buffer. If n is not specified, it is assumed that the process is to be performed only once. This command can be used only in the block mode.

Format

```
REN[EW] [n]
```

Example

```
*RENEW 10 ↵
```

In this example, ten consecutive blocks are transferred from the input file to the block buffer. The initial contents of the block buffer and the next nine blocks are transferred to the output file. The current line pointer is pointing to the first line in the tenth block, which is currently in the block buffer.

5.4.3.14 RETYPE CommandFunction

This command causes the current line to be replaced by "string". If "string" is not specified, the line will be deleted.

Format

```
R[ETYPE] [string]
```

Example

```
*R THIS IS A NEW LINE ↵
```

In this example, the string "THIS IS A NEW LINE" replaces the current line.

5.4.3.15 TOP OF FILE CommandFunction

This command causes a return to the top of the input file and saves previously edited text. If this command is issued when in line-by-line mode, EDI will switch to block mode after saving the edited data. The first block is read into the block buffer.

LINE TEXT EDITOR (EDI)

Format

TOF

Example

*TOF ↵

This command causes the previously edited pages to be written into the output file and the current line pointer to be reset to the top of the input file. The first block is read into the block buffer.

5.4.4 Sample Editing Session

Section B.2.1 contains a sample EDI editing session that illustrates how EDI commands can be used to edit a file.

5.5 EXTENDED EDI COMMANDS

5.5.1 Setup Commands

The setup commands required at the start of an editing session allow the user to provide parameters for EDI features. Table 5-4 contains a list of these commands.

LINE TEXT EDITOR (EDI)

Table 5-4
EDI Setup Commands

Command	Command Format	Description
BLOCK ON/OFF	BL[OCK] ON or BL[OCK] OFF	Switch text access modes.
CONCATENATION CHARACTER	CC [letter]	Change concatenation character to specified character (default is &).
OPENS	OP[ENS] filespec	Open specified secondary file.
OUTPUT ON/OFF	OU[TPUT] ON or OU[TPUT] OFF	Continue/Discontinue transfer to output file (line-by-line mode).
SELECT PRIMARY	SP	Reestablish primary file as input file.
SELECT SECONDARY	SS	Select opened secondary file as input file.
SIZE	SIZE n	Specify maximum number of lines to be read into block buffer (default is 80 lines).
TAB	TA[B] ON or TA[B] OFF	Turn automatic tabbing on or off. If TAB ON is selected, all text lines are moved over 8 spaces, unless the line has a label followed by a colon, or the line starts with a semicolon in column 1.
UPPER CASE ON/OFF	UC ON or UC OFF	Convert lower-case characters entered from terminal to upper-case characters.
VERIFY ON/OFF	V[ERIFY] ON or V[ERIFY] OFF	Select whether locative and change commands are verified.

LINE TEXT EDITOR (EDI)

5.5.1.1 BLOCK ON/OFF Command

Function

This command allows the user to switch between block mode and line-by-line mode. When BLOCK ON is issued, block mode becomes active, and the next block of text is brought into the block buffer. When BLOCK OFF is issued, the current block being processed is written to the output file, and line-by-line mode becomes active, with the first line from the next sequential block in the input file as the current line.

NOTES

1. If a conflicting BLOCK command is issued (e.g., BLOCK ON is issued when EDI is already in BLOCK ON mode), the command is ignored.
2. BLOCK ON is the default text access mode. It is assumed when neither ON nor OFF is specified.

Format

BL[OCK] ON

or

BL[OCK] OFF

Example

*BLOCK ON ↵

This command causes EDI to switch to block mode and the next block of text to be read into the block buffer.

5.5.1.2 CONCATENATION CHARACTER Command

Function

This command allows the user to change the command concatenation character to the specified character. If none is specified, the ampersand (&) is assumed. The concatenation character links two EDI commands on the same command line.

Format

CC [letter]

Example

```
*CC : ↵
*L A&B:C /A&B/ABC/ ↵
*CC & ↵
```

In this example, the string to be located contains an ampersand. Therefore, the default concatenation character must be changed to something else before the line can be located.

The first command line changes the default concatenation character from & to :.

The second command line instructs EDI to locate the string A&B and change string A&B to ABC. (Note: this line contains two commands which are concatenated by the new concatenation character, ":".)

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

5.5.1.3 OPENS CommandFunction

This command opens the specified secondary input file. The primary input file, if any, remains open and subsequent text is read from primary input file until the secondary input file is selected for input (see Section 5.5.1.6 for a description of SELECT SECONDARY command).

Format

OP[ENS] filespec

Example

```
*OPENS RICKS.MAC ↵
*SS ↵
*READ 1 ↵
```

In this example, the file RICKS.MAC is opened as a secondary input file, selected for input, and the first block is read in.

5.5.1.4 OUTPUT ON/OFF CommandFunction

This command allows the user to selectively continue/discontinue the transfer of text to the output file. OUTPUT ON is the default

LINE TEXT EDITOR (EDI)

condition; it is automatically established each time a CLOSE command is issued. This command can be used only in the line-by-line mode.

Format

OU[TPUT] ON

or

OU[TPUT] OFF

NOTE

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

Example

*OUTPUT OFF ↵

*NP 5 ↵

*OUTPUT ON ↵

In this example, the user wishes to bypass five lines of text in the input file without having these lines written into the output file.

The first command line causes the transfer of text to the output file to be disabled.

The second command line causes five consecutive lines of text from the input file to be bypassed.

The third command causes the transfer of text to the output file to be reenabled.

5.5.1.5 SELECT PRIMARY Command

Function

This command selects the primary file for input. It allows the user to reestablish the primary input file as the file from which text is to be read.

Format

SP

Example

```
*OPENS SECOND.MAC ↵  
*SS ↵  
*RENEW 10 ↵  
*CLOSES ↵  
*SP
```

In this example, the user

1. Opens the secondary file SECOND.MAC.
2. Selects SECOND.MAC as the secondary input file.
3. Issues a RENEW command, which reads ten consecutive blocks from the secondary input file into the block buffer. The first nine blocks are automatically transferred to the output file.
4. Closes the secondary input file SECOND.MAC.

NOTE

The secondary file need not be closed before the primary file is reselected as input.

5. Reselects the primary input file for input.

5.5.1.6 SELECT SECONDARY Command

Function

This command allows the user to select the secondary file as the input file.

Format

SS

Example

To add text to the output file from a secondary input file, the user must first open the secondary input file and select it for input. The use of the SS command is illustrated in the example presented in Section 5.5.1.5.

LINE TEXT EDITOR (EDI)

5.5.1.7 SIZE Command

Function

This command allows the user to specify the maximum number of lines to be read into the block buffer on a single READ. The default value for SIZE is 80 lines.

Format

SIZE n

Example

```
*SIZE 50↵
```

This command would condition EDI to read a default of 50 lines into the block buffer during a single READ command.

5.5.1.8 TAB ON/OFF Command

Function

Turn automatic tabbing on or off. TAB OFF is the default at the start of an editing session. TAB ON results in a tab (equivalent to eight spaces) being inserted automatically at the beginning of each input line unless the line contains a label followed by a colon, or the first column in the line contains a semicolon.

Format

TA[B] ON

or

TA[B] OFF

NOTE

If neither ON nor OFF is specified when a TAB command is issued, ON is assumed.

Example

```
*TAB ON↵  
*I↵  
; HI THERE. THIS IS A SAMPLE OF TABBING.↵  
THIS LINE GETS A TAB↵  
l: THIS ONE DOESN'T↵  
END↵  
*TAB OFF↵  
*N -3↵  
*P 4↵  
; HI THERE. THIS IS A SAMPLE OF TABBING.  
THIS LINE GETS A TAB  
l: THIS ONE DOESN'T  
END
```

LINE TEXT EDITOR (EDI)

5.5.1.9 UPPER CASE ON/OFF Command

Function

This command allows the user to enter lower-case characters from a terminal and have them converted to upper-case characters. If UPPER CASE OFF is issued, all input characters are accepted as they are entered (no conversion is performed), except that all EDI commands are converted to upper-case characters.

Format

UC ON

or

UC OFF

NOTE

If neither ON nor OFF is specified, then ON is assumed. UC ON is default at startup.

Example

```
*UC OFF ↵  
*I this line is entered in lower case ↵  
*UC ON ↵  
*I this line is converted to upper case ↵
```

Assuming that the input terminal is capable of generating lower case input, the above example would create the following lines in the output file.

```
this line is entered in lower case  
THIS LINE IS CONVERTED TO UPPER CASE
```

LINE TEXT EDITOR (EDI)

5.5.1.10 VERIFY ON/OFF Command

Function

This command allows the user to select whether locative and change commands are to be verified. Specifying VERIFY ON allows the user to determine whether the desired change has been correctly done. EDI is in the VERIFY ON mode at the start of editing.

Format

V[ERIFY] [ON]

or

V[ERIFY] [OFF]

NOTE

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

Example

```
*V OFF ↵
*L VERIFY ↵
*P ↵
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
*N -2 ↵
*V ON ↵
*L VERIFY ↵
LINE IS PRINTED AUTOMATICALLY IF VERIFY IS ON
```

In this example, the PRINT command is issued to verify that the desired line has been located when VERIFY is OFF, but when the LOCATE command is reissued with VERIFY ON, EDI prints the line automatically.

5.5.2 EDI Input/Output Commands

Input/Output commands are used to transfer text to and from input and output files. Table 5-5 contains a list of these commands.

LINE TEXT EDITOR (EDI)

Table 5-5
EDI Input/Output Commands

Command	Command Format	Description
FILE	FI[LE] filespec	Transfer lines from input file to both the output file and the specified file until a form-feed or end-of-file is encountered.
READ	REA[D] [n]	Read next n blocks of text into block buffer. If buffer contains text, new text is appended to it.
WRITE	W[RITE]	Write contents of block buffer to output file and erase block buffer.

5.5.2.1 FILE Command

Function

This command 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 end-of-file is reached. At that time the specified file is closed, and the form-feed character is not included in the specified file. During the transfer, the original file remains intact (i.e., all lines written to the specified file are also written to the normal output file, including the form-feed). When the command is complete, the current line in the input file is one line beyond the form-feed.

If the specified file is not an existing file, a new file is created. When the specified file does exist, the contents of the specified file are overwritten with the new data.

Format

FI[LE] filespec

Example

*FI SEC.DAT ↵

In this example, the contents of the input file from the current line to end-of-file (assuming that no form-feed character is encountered) is written into the output file and file SEC.DAT.

5.5.2.2 READ Command

Function

This command allows the user to read the next n blocks of text into the block buffer.

LINE TEXT EDITOR (EDI)

If a block is already in the buffer, the new blocks are appended to it.

NOTE

EDI must be running in block mode before this command can be executed.

Format

REA[D] [n]

If n is not specified, a value of 1 is assumed. The value of n must be positive.

Example

*READ 4 ↵

In this example, four blocks of the input file are read into the block buffer.

NOTE

The number of blocks specified should not exceed buffer capacity. If the blocks being read exceed block buffer capacity, EDI will fill the block buffer and then issue a diagnostic message explaining that the buffer capacity has been exceeded. Following the message, EDI will issue an * prompt and wait for another command. At this point, the user should issue a TOF command, locate the line where the first READ command was issued, and issue a READ command with a smaller number of blocks specified.

5.5.2.3 WRITE Command

Function

This command causes the entire contents of the block buffer to be written into the output file. The contents of the block buffer is then erased.

NOTE

EDI must be running in block mode before this command can be executed.

Format

W[RITE]

Example

*W ↵

*REA 2 ↵

In this example, the contents of the block buffer is written into the output file and the block buffer is erased. Then, the next two blocks are read into the block buffer.

5.5.3 Line Pointer Control (Locative) Commands

During editing operations, EDI maintains a pointer that identifies the current line (i.e., the line to which any subsequent editing operations will refer). Commands which modify the line pointer's location are called locative commands and they are listed in Table 5-6.

The user can issue commands which control the positioning of the line pointer and cause the pointer to be set to a line identified only by a text string contained in the line. The commands provided enable the user to:

1. Set the line pointer to either the top or bottom of the input file or block buffer.
2. Move the line pointer a specified number of lines away from its current position.
3. Cause a line, identified only by a text string, to become the current line.

NOTES

1. The carriage return key can be used to move the line pointer from the current line to the next line. Alternately, the ALTMODE (or ESCape) key can be used to move the line pointer from the current input line to the previous input line (block mode only).
2. If the VERIFY ON command is in effect, the line is printed when found.

LINE TEXT EDITOR (EDI)

Table 5-6
EDI Locative Commands

Command	Command Format	Description
BEGIN	B[EGIN]	Set current line to the line preceding top line in file or block buffer (identical to TOP).
BOTTOM	BO[TTOM]	Set current line to last line in file or block buffer.
END	E[ND]	Identical to BOTTOM.
FIND	[n]F[IND] (string)	Search current block or input file, beginning at line following current line for the "nth" occurrence of string. String must begin in column 1. Line pointer is set to indicated line.
OLDPAGE (Block Mode Only)	OL[DPAGE] n	Return to TOF and read page n into block buffer.
PAGE (Block Mode Only)	PAG[E] n	Enter block mode. Read page n into block buffer. Page n must be greater than current page number.
PAGE FIND (Block Mode Only)	[n]PF[IND] (string)	Search successive blocks for the "nth" occurrence of string. String must start in column 1.
PAGE LOCATE (Block Mode Only)	[n]PL[OCATE] (string)	Search successive blocks for the "nth" occurrence of string. String may be anywhere in line.
SEARCH AND CHANGE	SC /string-1/string-2/	Locate string-1 and replace all occurrences in line with string-2.
TOP	T[OP]	Identical to BEGIN.

5.5.3.1 BEGIN Command

Function

This command sets the current line pointer to the beginning (top) of the file (line-by-line mode) or block buffer (block mode). The current line is one line preceding the top line in the file or block buffer. This allows text to be inserted at the beginning of a file or block.

Format

B[EGIN]

Example

*B ↵

In this example, the current line pointer is moved to the top of the block buffer (block mode is assumed).

5.5.3.2 BOTTOM Command

Function

This command sets the current line pointer to the end 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 an EOF is reached. The last line is printed if the VERIFY ON command is in effect.

Format

BO[TTOM]

Example

*V ON ↵

*BO ↵

THIS IS THE LAST LINE

In this example, the line pointer is moved to the bottom of the block buffer, and the last line is printed.

5.5.3.3 END Command

Function

This command is identical to the BOTTOM command.

Format

E[ND]

LINE TEXT EDITOR (EDI)

5.5.3.4 FIND Command

Function

This command searches the block buffer or input file, beginning at the line following the current line for "string", which must begin in column 1 of the lines searched. If "string" is not specified, the next line is considered a match. A numeric value of "n" preceding the command results in finding the "nth" occurrence of "string" and positioning the line pointer to the line containing it.

FIND applies to the block buffer if EDI is in block mode and to the input file if EDI is in line-by-line mode.

When the line containing the desired "string" is found, it is printed if the VERIFY ON command is in effect.

Format

[n]F[IND] (string)

Example

*V ON ↵

*F LOOK ↵

LOOK AT THE FIRST CHARACTER IN THE LINE.

In this example, EDI searches the block buffer (or file) for a line that begins with LOOK and prints the line when it is found.

5.5.3.5 OLDPAGE Command

Function

This command causes EDI to enter block mode, if not already in block mode, and read page n into the block buffer. This command is identical to the PAGE command, except that OLDPAGE allows the user to back up to previous page "n". EDI actually accomplishes this by performing a TOF operation followed by a PAGE n operation (see Section 5.5.3.6).

Format

OL[DPAGE] n

Example

*OL 1 ↵

[PAGE 1]

In this example, EDI locates page 1, reads it into the block buffer, and prints the new page number.

5.5.3.6 PAGE CommandFunction

This command causes EDI to enter block mode, if not already in block mode, and read page n into the block buffer. A page is always delimited by form-feeds. If n is less than the current page number, an error message is displayed. Otherwise, the necessary number of RENEW commands is executed to read page n into the block buffer.

Format

PAG[E] n

Example

*PAG 7 ↵

[PAGE 7]

In this example, EDI locates page 7 if the present page number is less than 7, reads page 7 into the block buffer and prints the new page number.

5.5.3.7 PAGE FIND CommandFunction

This command functions identically to the FIND command, except that successive blocks are searched until the "nth" occurrence of "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 line searched.

Format

[n]PF[IND] (string)

NOTE

This command can be used only in block mode.

Example

This command is used in the same manner as the FIND command, except that the specified string can be in a block following the current block.

LINE TEXT EDITOR (EDI)

5.5.3.8 PAGE LOCATE Command

Function

This command causes a search of the current block, starting at the line following the current line, and 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.

Format

[n]PL[OCATE] (string)

NOTE

This command can be used only in block mode.

Example

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.

5.5.3.9 SEARCH AND CHANGE Command

Function

This command causes a search for string-1 in the block buffer (block mode) or input file (line-by-line mode), beginning at the line following the current line. The string may occur anywhere in the line. When string-1 is located, it and all occurrences of the string in that line are replaced by string-2. The located line becomes the current line.

If string-1 is not specified, then the match occurs on the next line, and string-2 is inserted at the beginning of the line. The new current line is printed if the VERIFY ON command is in effect. If string-1 is given but EDI cannot locate the string, EDI returns an * prompt, and the line pointer is positioned at the line preceding the top of the file or block buffer.

Format

SC /string-1/string-2/

LINE TEXT EDITOR (EDI)

Example

If the following incorrect line is contained in the current block:

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

The following command can be issued to correct the errors:

```
*V ON ↵
```

```
*SC /ES/IS/ ↵
```

```
THIS IS THE LINE TO BE ISSUED.
```

The corrected line is printed since the VERIFY ON command is in effect.

5.5.3.10 TOP Command

Function

This command sets the current line pointer to the top of the file (line-by-line mode) or block buffer (block mode). The current line is one line preceding the top line in the file or block buffer, allowing text to be inserted at the beginning of a file or block.

Format

```
T[OP]
```

Example

```
*T ↵
```

This command moves the line pointer to the top of the block buffer or file.

5.5.4 Text Modification and Manipulation Commands

The text modification and manipulation commands enable the user to display, change, and modify text. Table 5-7 contains a list of these commands.

LINE TEXT EDITOR (EDI)

Table 5-7
EDI Text Modification and Manipulation Commands

Command	Command Format	Description
ERASE	ERASE n	Erase the current line and next n-1 lines if in line-by-line mode. Erase the current block buffer and the next (n-1) blocks, if in block mode.
FORM FEED	FF	Insert form-feed into block buffer (used to delimit a page).
LINE CHANGE	[n]LC /string-1/string-2/	Change all occurrences of string-1 in current line (and n-1 lines) to string-2.
LIST ON TERMINAL	LI[ST]	Print on the user terminal all lines remaining in block buffer or in the input file, beginning at current line.
LIST ON PSEUDO-DEVICE	LP	Same as LI, except printing is performed on pseudo-device CL:.
MACRO	MACRO [x] (definition)	Used to define macros. Up to three macros can be defined. Numeric arguments to be passed at execution are identified by a % in definition.
MACRO CALL	MC[ALL]	Used to retrieve macro definitions stored in file MCALL;n.
MACRO EXECUTE	[n]Mx [a]	Execute macro [x] [n] times while passing numeric argument [a].
MACRO (IMMEDIATE)	[n]<definition>	Define and execute a macro n times.
OVERLAY	O[VERLAY] [n]	Delete n lines, enter Input mode, and insert new line(s) as typed, in place of original line(s).

LINE TEXT EDITOR (EDI)

Table 5-7 (Cont.)
EDI Text Modification and Manipulation Commands

Command	Command Format	Description
PASTE	PA[STE] /str-1/str-2/	Search all remaining lines in file or block buffer for string-1 and replace with string-2.
SAVE	SA[VE] [n] [filespec]	Save current line and the next n-1 lines in specified file. If filespec is not specified, lines are saved in file SAVE.TMP.
TYPE	TY[PE] [n]	Print next n lines. Line pointer remains at current line.
UNSAVE	UNS[AVE] [filespec]	Insert all lines from specified file following current line. If filespec is not specified, SAVE.TMP is used.

5.5.4.1 ERASE Command

Function

This command causes the current line, or the current block buffer, and the next n-1 blocks to be erased. In line-by-line mode, only a specification of 1 is allowed. This causes the current line to be erased. In block mode, the current block buffer and the next n-1 blocks are erased.

Format

ERASE n

NOTE

If n is not specified, +1 is assumed.

Example

*ERASE 5 ↵

This command causes the content of the current block buffer and the next 4 blocks to be erased and not written into the output file.

5.5.4.2 FORM FEED Command

Function

This command allows the user to insert form-feeds into the text. Form-feed cannot be entered in input mode. The form-feed is inserted after the current line, and the new current line becomes the line containing the form-feed. Form-feeds are used to delimit an EDI page.

Format

FF

Example

```
*P ↵
THIS IS THE LAST LINE ON THE PAGE
*FF ↵
```

In this example, a form feed is inserted into the text following the current line.

5.5.4.3 LINE CHANGE Command

Function

This command is similar to CHANGE, except that all occurrences of string-1 in the current line are changed to string-2. A numeric value "n" preceding the command results in the current line and the next n-1 lines being changed. If string-2 is null, all occurrences of string-1 are deleted. New lines are printed if the VERIFY ON command is in effect.

If string-1 is given but EDI cannot locate the string in the current line, EDI prints [NO MATCH] and returns an * prompt.

Format

[n]LC /string-1/string-2/

Example

If the current line is:

THES ES THE LINE TO BE ESSUED.

Then, the following command could be issued to correct the errors:

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

5.5.4.4 LIST ON TERMINAL Command

Function

This command prints on the user terminal all lines in the block buffer

LINE TEXT EDITOR (EDI)

(block mode) or all remaining lines in the input file (line-by-line mode), beginning at the current line. At the end of the listing, the current line pointer is repositioned to the top of the input file or block buffer.

NOTE

To suppress printing at any point, type CTRL/O.

Format

LI[ST]

Example

*LI ↵

This command causes all remaining lines in the block buffer or all remaining lines in the input file to be printed on the user terminal.

5.5.4.5 LIST ON PSEUDO-DEVICE Command

Function

This command functions in the same manner as the LIST ON TERMINAL command, except that the remaining lines in the block buffer (block mode) or the remaining lines of the input file (line-by-line mode) are listed on the pseudo-device CL:.

Format

LP

Example

*LP ↵

This command causes all remaining lines in the block buffer or all remaining lines in the input file to be printed on the pseudo-device CL:.

5.5.4.6 MACRO Command

Function

This command is used to define macros. Space is available for three macro definitions. The definition portion can be any legal EDI command or string of legal EDI commands connected by the concatenation character. If a numeric argument is to be passed to the macro at execution time, a percent sign (%) must be inserted in the macro definition at the point where the numeric argument is to be substituted. Then, the "a" value, which is passed via the MACRO EXECUTE command, replaces the percent sign when the macro is executed (see Section 5.5.4.8 for a description of the MACRO EXECUTE command).

LINE TEXT EDITOR (EDI)

Format

MACRO x definition

where: x is the macro number (1, 2 or 3).

Example

To find the nth occurrence of the string ABC in the current block and replace that occurrence and all remaining occurrences within the block with the string DEF, the following macro could be used:

```
*MACRO 1 %L ABC&PA /ABC/DEF ↵
```

The following command executes the macro and searches for the 10th and succeeding occurrence of ABC. (See Section 5.5.4.8.)

```
*M 1 10 ↵
```

The following macro definition and subsequent invocation could be used to change all occurrences of the strings ABC and GHI to DEF and JKL, respectively. The substitution is made in the current block, and the next four blocks (five blocks in all).

```
*MACRO 1 PA /ABC/DEF/&PA /GHI/JKL/&RENEW ↵ (MACRO command)  
*5M 1 ↵ (MACRO EXECUTE command)
```

5.5.4.7 MACRO CALL Command

Function

This command allows the user to retrieve up to three macro definitions previously stored by the user in a file. The macro definitions must contain only the "definition" portion of the MACRO command and will be stored in successive macro areas (i.e., the first macro definition goes into macro 1 area, the second definition, goes into macro 2 area, and the third goes into macro 3 area).

The filename of the file used to store the macro definitions must be MCALL;n, where n represents a version of the file. If the desired EDI macro definitions are not contained in the latest version of file MCALL, the version number of the file containing the desired definitions can be forced to the latest version number using the PIP COPY command with the /NV subswitch specified (see Section 2.4.2). The filename must have a null or blank file type.

Format

MC[ALL]

NOTE

Strings of concatenated EDI commands can be written as EDI macro definitions, and up to three EDI macro definitions can be stored in file MCALL;n. The MC command is used to call the latest version of file MCALL and move the three

LINE TEXT EDITOR (EDI)

definitions into the macro storage area. Then the user can execute the desired macro without having to type the complete command.

Example

*MC ↵

This command would retrieve the macro definitions stored in file MCALL;n, where n represents the latest version of the file MCALL.

5.5.4.8 MACRO EXECUTE Command

Function

This command causes macro "x" to be executed "n" times while passing it an optional numeric argument "a". If a macro numeric argument is defined via the percent sign (%) in the macro definition, the numeric argument contained in this command is passed for each execution of the macro (See Section 5.5.4.6.). Before a macro can be executed, it must have been defined via a MACRO command and stored in the EDI macro storage area. If the desired macro definition is stored in a file, the file must be called via a MACRO CALL command to move the definition into the EDI macro storage area.

NOTE

Using this command, any one of the three macro definitions stored in the EDI macro storage area can be executed any number of times.

Format

[n]Mx [a]

where:

n	is the number of times the macro is to be executed.
x	is the macro number.
a	is the numeric argument to be passed when the macro is executed (ignored if % argument is not present in macro definition).

Examples

*2M1 ↵

Execute macro number 1, twice.

*3M2 5 ↵

Execute macro number 2, three times, passing the numeric argument (5) each time the macro is executed.

Section B.2.4 contains an example which illustrates how a file can be edited using the EDI MACRO commands.

5.5.4.9 MACRO (IMMEDIATE) CommandFunction

This command allows the user to define and execute 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. (Macro storage is discussed in the description of MACRO CALL, Section 5.5.4.7.) The macro definition may also be subsequently executed via an M1 command. The command is actually equivalent to the two macro commands:

```
MACRO 1 definition
```

```
nM1
```

Format

```
n<definition>
```

Example

```
*<L ABC&C /ABC/DEF>
```

This command causes EDI to search the current block buffer for the string "ABC" and, when located, to change the string to "DEF".

Section B.2.3 contains an example which illustrates the use of the EDI immediate MACRO command.

5.5.4.10 OVERLAY CommandFunction

This command causes deletion of n lines and replacement with any number of lines typed in by the user. If n is not specified, the current line is deleted and replaced with any number of lines typed. When the OVERLAY command is issued, EDI enters Input mode. The user can enter text via the user terminal. To leave Input mode, a carriage return is typed as the first character in a line.

Format

```
O[VERLAY] [n]
```

NOTE

If n is not specified, a value of +1 is assumed.

Example

```
*O 2
```

This command deletes two lines and causes EDI to enter Input mode.

5.5.4.11 PASTE Command

Function

This command is identical to the LINE CHANGE command, except that all lines remaining in the input file or block buffer are searched, and all occurrences of string-1 are replaced with string-2. Modified lines are printed if the VERIFY ON command is in effect. If string-1 is given, but a match cannot be located, the EDI returns an * prompt. The line pointer is at the top of the buffer or top of file when the command is complete.

Format

PA[STE] /string-1/string-2[/]

Example

If the following lines are to be corrected:

```
THIS ARE LINE 1
THIS ARE LINE 2
THIS ARE LINE 3
```

The command is as follows:

```
*PA /ARE/IS/↵
```

If the VERIFY ON command is in effect, all corrected lines are printed.

NOTE

To discontinue printing, type CTRL/O.

5.5.4.12 SAVE Command

Function

This command causes the current line, and the next n-1 lines, to be saved in the file specified by the file specifier. If the file already exists, a new version is created with the same name, and the appropriate information is saved in the new file.

If no file is specified, a save file is generated, under the name SAVE.TMP.

NOTE

The input file or buffer information that is transferred to the SAVE file remains intact. The new current line pointer will be positioned to the last line saved. The SAVE command does not delete lines in the block buffer or input file.

LINE TEXT EDITOR (EDI)

Format

SA[VE][n] [filespec]

Example

A user can save and later insert small groups of lines in several places in an output file by using the SAVE and UNSAVE commands. Suppose the user has a file called EDIT.MAC which contains six lines to be inserted in a number of places in another file called HELP.MAC. The procedure is:

1. Start an editing session using EDIT.MAC as the input file.
2. Locate the lines to be inserted into HELP.MAC.
3. Issue SAVE 6 command.
(This transfers the six lines to be saved into the file SAVE.TMP.)
4. Issue a KILL command to terminate the editing session.
5. Start a new editing session using HELP.MAC as the input file.
6. Locate each place where the six lines are to be inserted and issue the UNSAVE command (see Section 5.5.4.14).
7. Make further edits to the input file, as desired, or EXIT.

NOTE

The save file is not deleted by EDI and remains on the specified volume until the user deletes it.

5.5.4.13 TYPE Command

Function

This command is functionally the same as the PRINT command, except that the current line pointer is unchanged. That is, after printing the specified number of lines, the line pointer is positioned to the first line printed by the TYPE command. (Compare with the PRINT command - Section 5.4.3.12.)

Format

TY[PE] [n]

NOTE

If n is not specified, a value of 1 is assumed.

Example

See the example of the PRINT command (Section 5.4.3.12).

LINE TEXT EDITOR (EDI)

5.5.4.14 UNSAVE Command

Function

This command retrieves all the lines in a specified file and inserts them immediately following the current line. If no file is specified, the file is defaulted to SAVE.TMP. The new current line pointer is positioned at the last line retrieved from the file. The file used in this command can be any text file; it is often the file created with a SAVE command.

Format

UNS[AVE] [filespec]

Example

If file SEC.DAT;1 contains a group of lines which are to be inserted following the current line, the following command performs the desired operation.

*UNS SEC.DAT;1 ↵

Section B.2.2 contains an example using the EDI SAVE and UNSAVE commands.

5.5.5 EDI Close Operation Commands

The close operation commands are used to terminate EDI operations and to write the remainder of the input file into the output file. Table 5-8 contains a list of these commands.

Table 5-8
EDI Close Operation Commands

Command	Command Format	Description
CLOSE	CL[OSE] [filespec]	Transfer remaining lines in block buffer and input file to output file and close file. If file specifier is used, output file is renamed. EDI> prompt is issued.
CLOSES	CLOSES	Close secondary file.
CLOSE & DELETE	CD[L] [filespec]	Same as CL, except input file is deleted. EDI> prompt is issued.
EXIT & DELETE	ED[X] [filespec]	Same as CDL, except after files are closed and renamed, EDI exits.
KILL	KILL	Input file and output file are closed. Output file is deleted. EDI> prompt is issued.

LINE TEXT EDITOR (EDI)

5.5.5.1 CLOSE Command

Function

This command transfers all remaining lines in the block buffer and input file (in that order) into the output file, and closes both files. If a file specifier is included, the output file is renamed to the specified file. EDI then returns to its initial command sequence, prompts with EDI> and waits for another file specifier to be entered.

NOTE

If a secondary file was opened during the editing session and not closed, it remains open.

Format

CL[OSE] [filespec]

Example

```
*CL ↙  
EDI>
```

This command closes both input and output files, and EDI returns to the initial command sequence.

5.5.5.2 CLOSES Command

Function

This command is used when the user has finished extracting text from a secondary file and wishes to close it. The secondary file is closed and cannot be used for input unless reopened.

Format

CLOSES

5.5.5.3 CLOSE AND DELETE Command

Function

This command transfers all remaining lines in the block buffer and the input file (in that order) into the output file, and closes both files. The input file is then deleted. If a file specifier is included, the output file is renamed to the specified file. In effect, this command acts just like CLOSE, except that the input file is deleted. See NOTE contained in Section 5.5.5.1.

Format

CD[L] [filespec]

5.5.5.4 EXIT AND DELETE Command

Function

This command functions in the same way as the CLOSE and DELETE command, except that, after the files are closed and renamed, EDI exits.

Format

ED[X] [filespec]

5.5.5.5 KILL Command

Function

This command returns EDI to the initial command sequence without retaining the output file. When this command is executed, the input file is closed, and the output file is deleted.

Format

KILL

Example

```
*KILL ↙  
EDI>
```

In this example, the output file is not retained, and EDI returns to the initial command sequence waiting for the next file specifier.

5.6 EDI ERROR MESSAGES

The four classes of EDI error messages are:

- Command level informational and error messages
- File access warning messages
- Error messages requiring EDI restart
- Fatal error messages

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

5.6.1 Command Level Informational and Error Messages

Messages in this class indicate information that is designed to be helpful to the user or identify errors that were encountered in the previous command. All messages in this class are enclosed within square brackets and followed by a prompt for a new command. For example, the following output occurs if a delete command encounters an

LINE TEXT EDITOR (EDI)

end-of-buffer in block mode:

```
[*EOB*]  
*
```

Note that immediately following the message, EDI outputs an asterisk to prompt for the next command.

The messages in this class follow.

[ALREADY PASSED THAT PAGE!]

Description

The user has attempted to access a page number that is less than the current page. Prior pages can be accessed only via the OLDPAGE command.

Suggested User Action

If the PAGE command has been incorrectly entered, retype the command with the proper page number. Otherwise, use an OLDPAGE command to access the desired page.

[BUFFER CAPACITY EXCEEDED BY]
offending line
[LINE DELETED]

Description

A READ, UNSAVE, INSERT, or OVERLAY command has caused the capacity of the block buffer to be exceeded. The line that caused the overflow is displayed and deleted.

Suggested User Action

If a new file is being created, empty the buffer with a WRITE command and continue the editing session.

If an existing file is being edited, it may be possible to continue via a RENEW or WRITE command. Otherwise, use the CLOSE command to close the output file and save all edits. Reopen the output file as the input file and, using the SIZE command, reduce the number of lines read into each buffer; then, using the PAGE LOCATE command, search to the position in the file where editing is to continue.

Occasionally, a file that was not created by EDI causes this message (i.e., an attempt to open the file for input produces this message). If this occurs, the following procedure may be used to successfully edit the file:

1. Start the editing session by specifying a filename that does not correspond to any file in the current directory. This causes EDI to create a new file and enter Input mode.
2. Type carriage return to enter edit mode.

LINE TEXT EDITOR (EDI)

3. Using the SIZE command, reduce the number of lines read into each buffer.
4. Use the KILL command to terminate the creation of the file.
5. When EDI prompts for a new file specifier, enter the name of the desired file.

[CONCATENATING CHAR CHANGED TO "&"]

Description

The user has changed the command concatenation character and an OLDPAGE, TOF, or TYPE command has caused it to be changed back to "&".

Suggested User Action

Use the CC command to reestablish the desired command concatenation character.

[CREATING NEW FILE]

INPUT

Description

The input file specified in the command does not exist and EDI has created a new file. EDI automatically enters Input mode and awaits the input of text lines.

Suggested User Action

If the intent is to create a new file, continue the editing session, entering new lines as required. Otherwise, enter Edit mode by typing carriage return; use the KILL command to terminate the creation of the new file. When EDI prompts for a new file specifier, enter the correct file specification.

[ILL CMD]

Description

A command that is not recognized by EDI has been entered; or a command that is not compatible with the current mode has been attempted (e.g., a READ command in line-by-line mode).

[ILL NUM]

Description

A non-numeric character has been specified in a numeric field, or a negative number has been entered where only positive numbers are allowed.

LINE TEXT EDITOR (EDI)

[ILL STRING CONST]

Description

A search string specified in a CHANGE, LC, PASTE, or SC command contains only one command concatenation character or does not contain a matching string termination character (e.g., PASTE /ALPHABETA, whereas PASTE /ALPHA/BETA is correct).

[ILLEGAL IN BLOCK ON MODE]

Description

An attempt has been made to execute a command that is illegal in block mode.

[ILLEGAL FILE NAME GIVEN IN CLOSE OR EXIT]

or

[FILE WAS NOT RENAMED]

Description

A syntactically incorrect file specifier has been given in a CLOSE or EXIT command, or the attempt to rename the output file has failed.

Suggested User Action

The output file is closed under the name of the input file without any loss of information. The Peripheral Interchange Program (PIP) can be used to rename the file to the desired name.

[MACRO NOT DEFINED]

Description

An attempt has been made to execute a macro with the M command, but the specified macro has not been defined.

Suggested User Action

Use the MACRO command to define the desired macro and then execute it with the M command.

[MACRO NUMERIC ARG UNDEFINED]

Description

A macro has been executed with an M command that did not contain a numeric argument, and the body of the referenced macro contains the numeric argument replacement character "%".

LINE TEXT EDITOR (EDI)

Suggested User Action

Retype the command, specifying the appropriate numeric argument.

[MCALL FILE DOES NOT EXIST]

Description

An MCALL command has been executed to define a set of macros, but the file MCALL does not exist in the current directory.

Suggested User Action

The desired set of macro definitions may exist under another UFD. If this is the case, use PIP to copy or rename the MCALL file into the current directory.

[NO INPUT FILE OPEN]

Description

A PAGE, READ or RENEW command has been attempted and a new file is being created. These commands can be executed only when an existing file is being edited.

[NO MATCH]

Description

A CHANGE command has specified a string to be changed that is not in the current line.

[OVERLAYING PREVIOUSLY DEFINED MACRO]

Description

A MACRO command has resulted in the redefinition of a previously defined macro. This message is intended to make the user aware that the previous definition is no longer in effect.

[SAVE FILE DOES NOT EXIST]

Description

A file was specified in an UNSAVE command that cannot be located in the respective directory.

Suggested User Action

Examine the file specifier to ensure its correctness. If the file specifier is in error, correct the error, then retry the command.

LINE TEXT EDITOR (EDI)

[SECONDARY FILE ALREADY OPEN]

Description

An attempt has been made to open a secondary input file when another secondary input file is already open. Alternatively, a CLOSE or KILL command has been executed, or an error has been encountered that causes EDI to restart, and the secondary file is found to be open from the previous edit. The former case represents an error, whereas the latter informs the user that he still has a secondary file open.

Suggested User Action

Close the secondary input file using the CLOSES command, and then open the desired secondary file with the OPENS command.

[SECONDARY FILE CURRENTLY SELECTED FOR INPUT]

Description

A CLOSE or KILL command has been issued, or an error has caused EDI to restart, when the secondary input file is open and selected for input.

Suggested User Action

Issue an SP command, a CLOSES command and proceed.

[SYNTAX ERROR]

Description

A command has been entered that is syntactically incorrect.

[TOO MANY CHARS]

Description

A CHANGE, LC, PASTE, or SC command has resulted in a line that contains too many characters. EDI limits the length of a line to 90 characters.

Suggested User Action

Retype the line to ensure that the line is valid.

[*BOB*]

Description

The beginning-of-buffer has been reached. The current line pointer is positioned just before the first line in the buffer. Thus, new text lines can be entered before the first line.

[*EOB*]

Description

The end-of-buffer has been reached. The current line pointer now points to the beginning of the buffer. Thus, if new lines are inserted, they appear before the first line in the buffer.

[*EOF*]

Description

The end-of-file has been reached on the input file.

Suggested 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 and then continue editing the file.

5.6.2 File Access Warning Messages

Messages in this class represent attempts on the part of the user to access directories, files, or devices that are not present in the host system. Each message is prefixed with:

EDI --

and, after the message is displayed, EDI returns to command level and prints an asterisk to request input.

The messages in this class follow.

EDI -- DEVICE NOT IN SYSTEM

Description

A FILE, OPENS, SAVE, or UNSAVE command contains the specification of a device that does not exist in the host system.

Suggested User Action

Reenter the command line, specifying only devices available in the system.

EDI -- FILE DOES NOT EXIST

Description

An attempt has been made in a FILE or SAVE command to create a file in a directory that does not exist on the specified volume.

LINE TEXT EDITOR (EDI)

WARNING

The remaining error messages in this class should not occur and represent failures in EDI. If such errors persist, consult your DEC software support representative.

EDI -- BAD DEVICE NAME

EDI -- BAD FILE 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

5.6.3 Error Messages Requiring EDI Restart

The error messages described in this section are caused by conditions encountered by EDI that make it impossible to continue the current editing session. EDI closes all open files (with the exception of the secondary input file), reinitializes, and then prompts for the next file to be edited.

As with file access warning messages, each message in this class is prefixed with:

EDI --

After the appropriate message has been displayed, EDI prompts with:

EDI>

The editing session may be terminated at this point by typing carriage return, or it can be continued by entering the next file specifier. If a secondary file was open when the error condition was encountered, the secondary file must be closed using the EDI commands.

The messages in this class follow.

LINE TEXT EDITOR (EDI)

EDI -- BAD RECORD TYPE - FILE NO LONGER USABLE

Description

The record type defined in the header block of the input file (primary input, secondary input, UNSAVE, or MCALL) is not supported by File Control Services (FCS); thus, the file cannot be used for input to EDI.

Suggested User Action

The referenced file has been created without using FCS, or the file structure on the volume is damaged. If the latter is the case, the validity check of the file structure verification utility (VFY) should be run to determine the extent of the damage. VFY is described in Chapter 8.

EDI -- FILE IS ACCESSED FOR WRITE

Description

The input file (primary input, secondary input, UNSAVE, or MCALL) is currently being written by another task.

Suggested User Action

Wait for the file to be written and then reenter the command line.

EDI -- FILE IS LOCKED TO WRITE ACCESS

Description

The output file (text output, FILE, or SAVE) is currently accessed for read by one or more tasks and is locked against all writers.

Suggested User Action

Wait for all readers of the file to finish, then reenter the command line.

EDI -- ILLEGAL RECORD ATTRIBUTES - FILE NOT USABLE

Description

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.

Suggested User Action

The referenced file has been created without using FCS or the file structure on the volume is damaged. If the latter is the case, the validity check of the file structure verification utility (VFY) should be run to determine the extent of the damage.

LINE TEXT EDITOR (EDI)

EDI -- PRIMARY FILE NOT PROPERLY CLOSED

Description

When the primary input file was last written, a close check was specified, and the writing task did not properly close the file (e.g., the task was aborted). Thus, the file attributes were not written, and the file may contain inconsistent data.

Suggested User Action

Exit from EDI by typing carriage return. Run the Peripheral Interchange Program (PIP) and use the /UN switch to unlock the file. Reinitiate EDI and try to recover the data in the file.

EDI -- PRIVILEGE VIOLATION

Description

A privilege violation occurs during a file access for the following reasons:

1. The specified volume is not mounted.
2. The UIC under which EDI is running does not possess the necessary privileges to access the specified directory.
3. The UIC under which EDI is running does not possess the necessary privileges to access the specified file.

Suggested User Action

If the volume is not mounted, then mount it using the MCR MOUNT command. Otherwise, reinitiate EDI under a UIC that has appropriate access privileges to both the specified directory and the file.

EDI -- RECORD IS TOO LARGE FOR USER BUFFER

Description

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 90 bytes.

EDI -- SECONDARY FILE NOT PROPERLY CLOSED - NOT USABLE

Description

When the secondary input file was last written, a close check was specified, and the writing task did not properly close the file (e.g., the task was aborted). Thus, the file attributes were not written, and the file may contain inconsistent data.

Suggested User Action

Run PIP and use the /UN switch to unlock the file. Reinitiate EDI and try to recover the data in the file.

LINE TEXT EDITOR (EDI)

WARNING

The remaining error messages in this class should not occur and represent failures in EDI. If such errors persist, consult your DEC software support representative.

EDI -- BAD DIRECTORY SYNTAX

EDI -- DUPLICATE ENTRY IN DIRECTORY

EDI -- END OF FILE

EDI -- ILLEGAL RECORD ACCESS BITS - FILE NOT USABLE

EDI -- ILLEGAL RECORD NUMBER - FILE NOT USABLE

5.6.4 Fatal Error Messages

The fatal error messages represent system and/or hardware error conditions which make it impossible for EDI to continue execution. All files are closed and EDI terminates its execution. The output file may be truncated. Each error message is prefixed with:

EDI --

and followed by the exit message:

[EXIT]

on the next line.

The advanced user may be able to utilize the truncated version of an output file in the following manner to save the editing performed prior to the fatal error condition.

1. Use PIP to rename the truncated version of the output file to avoid confusion.
2. Restart the editing session on the original input file.
3. Issue an OPENS command, specifying the renamed file as the secondary file.
4. Issue an SS command to select the secondary file for input.
5. Issue an ERASE command to erase the first block of the input file, unless the truncated output file did not contain the entire first block.
6. Issue as many READ l and WRITE commands as necessary to reach the EOF on the secondary file.
7. Issue an SP command to select the primary file for input.

LINE TEXT EDITOR (EDI)

8. Issue a CLOSES command to close the secondary file.
9. Issue a WRITE command to ensure that the last block was written into the output file.
10. Issue as many READ 1 and ERASE commands as necessary to bypass all input file blocks which are complete in the renamed file.
11. Continue the normal editing session.

The messages in this class follow.

EDI -- CALLER'S NODES EXHAUSTED

Description

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.

Suggested 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

Description

Insufficient space exists on the output volume to extend an output file (text output, FILE, or SAVE).

Suggested User Action

Determine which volume is being written. If it is required that the specified file be written on this volume, then space must be made available. Use PIP to purge (/PU) or delete (/DE) unwanted files.

EDI -- FILE HEADER CHECKSUM ERROR

Description

An input file (primary input, secondary input, UNSAVE, or MCALL) has a header block that does not contain a proper checksum.

Suggested User Action

The file structure on the specified volume is damaged. Run validity check of the file structure verification utility (VFY) to determine the extent of the damage. VFY is described in Chapter 8.

LINE TEXT EDITOR (EDI)

EDI -- FILE HEADER FULL

Description

Insufficient retrieval pointer space exists in the header block to extend an output file (text output, FILE, or SAVE).

Suggested User Action

An attempt is being made to create an output file that is larger than can be described in a file header block. Split the file into two or more files and process them separately.

EDI -- FILE PROCESSOR DEVICE WRITE ERROR

Description

This error message can be received by initiating an editing session on a write-locked device.

Suggested User Action

Unlock device if it is write-locked. Otherwise, a hardware problem may exist. Consult the DEC field service representative.

EDI -- INDEX FILE FULL

Description

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. An attempt has been made to exceed this maximum.

Suggested User Action

Determine which volume is being referenced. If it is required that the specified file be created on this volume, then space must be made available. Use PIP to purge (/PU) or delete (/DE) unwanted files.

WARNING

The following error messages signify hardware problems. If possible, all important files should be removed from the volume. If errors persist, consult the DEC field service representative.

LINE TEXT EDITOR (EDI)

EDI -- BAD BLOCK ON DEVICE

EDI -- FILE PROCESSOR DEVICE READ ERROR

EDI -- HARDWARE ERROR ON DEVICE

EDI -- PARITY ERROR ON DEVICE

WARNING

The remaining error messages in this class should not occur and represent failures in EDI. If such errors persist, consult your DEC software support 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 -- WRITE ATTRIBUTE DATA FORMAT ERROR

EDI -- UNEXPECTED ERROR - EDITOR WILL ABORT

TASK "...EDI" TERMINATED

CHAPTER 6
SOURCE LANGUAGE INPUT PROGRAM (SLP)

6.1 INTRODUCTION TO SLP

The Source Language Input Program (SLP) is a batch-oriented editing program that is used to create and maintain source language files on disk.

6.2 PREPARING TO RUN SLP

Before attempting to use SLP, the user should become familiar with SLP's capabilities, environment, and restrictions.

6.2.1 Capabilities

SLP permits the user to:

1. Create new source files.
2. Create indirect files which contain SLP edit control commands.
3. Edit an already existing source file. The following editing commands are provided:
 - a. Delete
 - b. Replace
 - c. Insert
4. Obtain line-number listings of files. These listings can be used as an aid to editing the file.

6.2.2 Environment

SLP accepts input from the following media:

1. Any RSX-11M supported terminal device (on-line).
2. Card reader or indirect command file.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

3. Any RSX-11M supported volume (on-line or indirect command file).

6.2.3 Restrictions

1. The user must know in advance which lines, by line number, he wants to edit. It is advisable, therefore, for the user to have on hand a current line-number listing of the file he wants to edit (line boundaries are the first character and the carriage return).
2. SLP cannot handle input lines greater than 80 ASCII characters in length. If more than 80 characters are specified an error is declared:
3. Edit commands refer to line numbers that must be in ascending order starting with 1 and continuing throughout the entire file. Form feeds and page directives are treated as though they are simply part of the text.

6.3 INITIATING SLP

All RSX-11M utility programs can be initiated in several ways. These methods are described in Section 1.2. The methods for SLP are:

```
>SLP↵  
>SLP command string↵  
>RUN ...SLP↵  
>RUN ...SLP/UIC=[group,member]↵  
>RUN $SLP↵  
>RUN $SLP/UIC=[group,member]↵
```

6.4 SLP STARTUP

After SLP has been initiated, the user must enter an initial command to direct SLP to perform the desired function (e.g., create a new file, select a file for editing, or list a file). The general format of SLP commands is as follows:

```
outfile[/switch][,listfile[/switch]] =infile[/switch]
```

where:

/switch is one or more of SLP's optional output control switches. The SLP output control switches are described in Section 6.5.

For a complete description of file specifiers, see Section 1.3.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

6.4.1 Defaults in File Specifiers

Defaults in SLP file specifiers are described in Table 6-1.

Table 6-1
Defaults in SLP File Specifiers

Specifier	Default
dev:	<u>Output File and Input File</u> SY0: <u>List File</u> The device specified or implied by the output file specifier.
[uic]	<u>Output File and Input File</u> The UIC under which SLP is currently running. <u>List File</u> The UIC specified or implied by the output file specifier.
filename	Must be specified
.type	Must be specified
;ver	Latest version for input files; latest version plus one for output and listing files.

6.4.2 Examples of SLP Initialization

Example 1.

SLP>KATESFILE.MAC ↵

A new file, KATESFILE.MAC, is to be created on SY0:. SLP expects input from the terminal.

Example 2.

SLP>,LP:=KATESFILE.MAC ↵

This example produces a line-number listing of KATESFILE.MAC on the line printer. No other output file is produced.

Example 3.

SLP>KATESFILE.MAC,LP:=KATESFILE.MAC;1 ↵

In this example, the input file(KATESFILE.MAC;1) is to be edited, producing an updated output file name KATESFILE.MAC with a version

SOURCE LANGUAGE INPUT PROGRAM (SLP)

number one greater than the latest version for the file. In addition, a line-number listing of the output file (KATESFILE.MAC) is produced on the line printer.

6.5 SLP OUTPUT CONTROL SWITCHES

The SLP output control switches are described in Table 6-2. A switch specification consists of a slash (/) followed by a 2-character name for an enabling function, or a slash (/) followed by a minus (-) and a 2-character switch name for a disabling function. If more than one switch is used, each switch is preceded by a slash.

6.6 SLP OUTPUT FILES

When a file is edited, SLP produces an output file on disk under the name specified by the user. If the /AU switch is specified (default condition), the output file contains information about the changes that have been made, so the user can more readily determine how the new file differs from the old one.

Unless specifically suppressed, an audit trail is always created in the output file, indicating changes effected by the edits.

Each line that has been inserted during the last editing session is flagged by appending the characters ****NEW**** to the line.

The line following the inserted line(s) may be flagged by the characters ****N**, where N is a decimal value equal to the number of lines that were deleted from the old file. For example:

```
      ;THIS IS A NEW LINE ADDED TO THE FILE      ;**NEW**
      ;THIS IS THE NEXT LINE                      ;**-1
```

indicates that the new line has simply replaced one of the old lines. That is, the edit command looked like:

```
      -m,m ↵
      ;THIS IS A NEW LINE ADDED TO THE FILE ↵
```

where m is the number of the line that was replaced. There may also be entries of the following kind:

```
      ;THIS LINE IS A REPLACEMENT                ;**NEW**
      ;NEXT OLD LINE                             ;**-16
```

indicating that a new line has been inserted, but 16 lines have been deleted immediately-preceding the next old line.

Lines may also be flagged with just the ****N** characters, with no immediately-preceding new lines, to indicate simply that lines have been deleted, without being replaced.

If the /AU switch is specified when a file is being edited, the current flags are stripped prior to output of the updated file. Thus, the flags are reliable indicators of the editing that was done on the most recent update of the file.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

Table 6-2
SLP Output Control Switches

Switch	Description
/AU or /-AU	<p>The /AU switch causes the editing audit trail to be included, and the /-AU switch suppresses the editing audit trail. That is, new and deleted lines are not flagged when /-AU is in effect. Either switch can be specified in the command string, in either an input or an output file specifier. For example:</p> <p style="text-align: center;">DK1:AX.MAC/-AU,LP:=AX.MAC ↵</p> <p style="text-align: center;">or</p> <p style="text-align: center;">DK1:AX.MAC,LP:=AX.MAC/-AU ↵</p> <p>Both specifications are legal and produce an output file with no audit trail.</p> <p>/AU is the default condition.</p>
/BF or /-BF	<p>When an audit trail is being produced, it may be desirable to specify the blank fill (/BF) switch, especially when the file being edited is a FORTRAN source program. The /BF switch causes the audit trail text to be right-justified by inserting blanks at the end of the text line, rather than tabs.</p> <p>To disable the blank fill function, specify /-BF. Either switch may be specified on input or output files. Neither has any effect if auditing is suppressed.</p> <p>/BF is the default condition.</p>
/DB or /-DB	<p>Listing files are normally single-spaced. To create a double-spaced listing file, the /DB switch must be specified, as in the following example:</p> <p style="text-align: center;">DK0:XYZ.MAC,LP:/DB=DK0:XYZ.MAC;3 ↵</p> <p>/-DB is the default condition.</p>
/SP or /-SP	<p>Listing files may be spooled to a file-structured volume by specifying a listing file as shown in the following example:</p> <p style="text-align: center;">DK:ABC.MAC,DK1:/SP=ABC.MAC ↵</p> <p>This causes the list file to be written to the volume mounted on DK1: prior to being output on a line printer. The /SP switch has no effect if the listing device is not file-structured. That is, if the line printer is specified, no spooling occurs. Output directly to a file-structured device without printing the file can be accomplished by specifying /-SP. The print spooler is described in Appendix C.</p> <p>/SP is the default condition.</p>

6.7 SLP EDIT COMMANDS

Following the initial command line, the user enters text lines, or deletes or corrects lines in the original source file. Text that is to be inserted at the beginning of the file is entered immediately following the initial command line. To correct or replace a line, or lines, or to insert text in the middle or at the end of the file, the user must first specify an edit command, followed by a decimal value referring to a line in the input file.

For example:

-9

The minus sign and line number may appear as the only element on the line, or they may be followed by a comma and a second line number, as shown below:

-9,12

-9,9

SLP interprets the user's purpose by examining the edit command. When a single line number is specified (e.g., -9 alone), SLP interprets the user's purpose to be the insertion of new text lines into the source file. The line number indicates that the new text is to be inserted following the specified line (in the first example, new text would be placed in the file following line 9).

When the user provides an edit command in the second format (-9,12), SLP deletes all text lines from line 9 through line 12, inclusively. The user can follow the edit command with lines of text, which will be inserted into the file in the location previously occupied by the deleted lines (i.e., the first new text line is the new line 9).

The edit command (-9,9) indicates that SLP is to delete line 9. If a text line (or lines) follows, it replaces the deleted line.

NOTE

Line numbers must always be specified in ascending sequence. Thus, -9,8 is illegal, and an error message is printed (refer to Section 6.10). It is also illegal to refer to a line number lower than a line number that was referenced in a prior edit command.

6.7.1 SLP Edit Control Characters

SLP recognizes four edit control characters in character position 1: the "minus" sign (-), the "less than" sign (<), the slash (/), and the "at" sign (@). Table 6-3 describes the function of these characters.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

Table 6-3
SLP Edit Control Characters

Character	Function
- (minus)	<p>Indicates that an editing function is to be performed with reference to the line number(s) specified.</p> <p>-n Insert text following line n.</p> <p>-n,n Delete line n.</p> <p>-n,m Delete lines n through m, inclusively.</p>
/ (slash)	<p>The slash is placed in the first position of a line to indicate that the editing of a file is complete. SLP responds by printing SLP> to inform the user that it has terminated editing on the previously specified file and is now ready to begin editing another. The user responds either by entering a new file specification, or by terminating the editing session altogether by typing CTRL/Z.</p>
@ (at)	<p>The @ character is put in the first location of a line to indicate that SLP is to seek input from an indirect file. That is, input is to be found in a file rather than being entered from the terminal. The user must indicate the device and file by specifying their names immediately after the @ sign. For example:</p> <p style="padding-left: 40px;">@DK:DKSFIL.COR ↵</p> <p>instructs SLP to read input from the file DKSFIL.COR on physical device unit DK0:. Indirect files are more fully described in Section 6.8. Unless otherwise specified, the file type defaults to CMD.</p>
< (less than)	<p>The < character is used when entering a line that begins with one of the special edit control characters. It causes the line to be shifted one character position to the left, resulting in the deletion of the < character and the entry of the desired control character as the first character on the line. This is especially useful when creating correction files. For example:</p> <p style="padding-left: 40px;"><@DIRBLK.COR ↵</p>

SOURCE LANGUAGE INPUT PROGRAM (SLP)

Table 6-3 (Cont.)
SLP Edit Control Characters

Character	Function
	causes the line @DIRBLK.COR to be entered into the file. Subsequently, when this file is referenced by SLP, the line @DIRBLK.COR ↵ will be interpreted as a reference to an indirect file named DIRBLK.COR. Similarly, specifying <-23,29 results in the edit command -23,29 being written into the output file. Thus, when the file is read by SLP, lines 23 through 29 of the specified input file will be deleted.

6.8 INDIRECT FILES

Indirect files contain both editing commands and correction lines to be inserted into the file being edited. These files are input from a device other than the terminal. An example of indirect file usage in SLP follows. See Section 1.4 for a discussion of indirect files.

6.8.1 Creating an Indirect File

The following example shows how corrections and SLP commands are inserted into an indirect file.

Example

1. Initiate SLP by specifying an initial command line that contains an output file specifier and a listing file specifier. The following is a typical command line:

```
SLP>FROG.COR,LP:↵
```

In this example, SLP will generate the file FROG.COR, and will produce a listing of this file as it is generated.

2. Begin entering correction or insertion lines into the file. Supply the appropriate edit commands and insert as text. The correct line numbers for text being edited must be obtained from a listing of the file that is to be edited; new text, however, can be entered without line numbers. In order to generate edit command lines, it is necessary to use the shift command character (<) in the first position of the input

SOURCE LANGUAGE INPUT PROGRAM (SLP)

line. For example, to create the line

```
-29,36
```

as a text line in FROG.COR, you must specify

```
<-29,36↵
```

Thus, a typical correction file might be entered as:

```
<-15↵  
;THIS ROUTINE CALLS THE ERROR PROCESSOR↵  
$HDR:          SAVRG          ;SAVE NONVOLATILE REGISTERS↵  
                BNE 10$      ;IF NE YES↵  
                RETURN↵  
<-40,56↵
```

6.8.2 Using Indirect Files

The following example shows how an indirect file can be used.

Example:

1. Assuming that the corrections contained in AMND.COR are to be applied to a file named WPT.MAC, the file specifier to SLP is:

```
SLP>WPT.MAC,LP:=WPT.MAC↵
```

which is followed by the indirect-file edit control statement referring to the correction file:

```
@AMND.COR↵
```

NOTE

If the initial command line, WPT.MAC,LP:=WPT.MAC, was included in the indirect file, then editing would have been initiated as follows:

```
SPL>@AMND.COR↵
```

2. SLP reads correction input from the latest version of file AMND.COR and returns to the terminal for further input to finish updating WPT.MAC. If the user includes a termination edit control line in AMND.COR (</), SLP responds by printing SLP> at the terminal after terminating editing operations on WPT.MAC. All files are closed and a listing is sent to LP:.

6.9 SLP EDITING EXAMPLES

The following examples indicate the various editing functions that SLP can perform and the command formats used.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

Example 1:

```
SLP>ALBLK.MAC,LP:=ALBLK.MAC;1 ↵  
-23,23 ↵  
; R1=SIZE OF BLOCK TO ALLOCATE IN BYTES. ↵  
-33 ↵  
    MOV      #$FRHD,R2    ;GET ADDRESS OF FREE POOL HEADER ↵  
-36,36 ↵  
-39,39 ↵  
    ASR      R1           ;CONVERT TO WORDS ↵  
/ ↵
```

In this example, the following editing functions are performed:

Line 23 is replaced by a corrected version (i.e.,; R1=SIZE OF BLOCK TO ALLOCATE IN BYTES.).

A new line is inserted after line 33.

Line 36 is deleted (and not replaced).

Line 39 is replaced by a corrected version (i.e.,

ASR R1 ;CONVERT TO WORDS).

The output file is named ALBLK.MAC, and a line-numbered listing is produced on the line printer.

Example 2:

```
SLP>BLKSG.MAC,LP:=BLKSG.MAC;1 ↵  
-55,55 ↵  
    BCS      60$          ;IF CS YES ↵  
-107,107 ↵  
    CALL     $ERMSG      ;OUTPUT ERROR MESSAGE ↵  
/ ↵
```

In Example 2, the following editing functions are performed:

Line 55 is replaced by a corrected line;

Line 107 is replaced by a corrected line.

The output file is named BLKSG.MAC, and a line numbered listing is produced.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

Example 3:

```

SLP>CATB.ABC;1,DK1:CATB.DAT=CATB.ZYX ↵
-15,16 ↵
CNTRL: .BYTE '9,'0 ↵
-33,35 ↵
$CDTD: MOVB #'9,CNTRL ;SET DECIMAL LIMIT ↵
-38,38 ↵
COTB: MOVB #'7,CNTRL ;SET OCTAL LIMIT ↵
-43,45 ↵
        CMPB #' ,R5 ;BLANK? ↵
        BEQ 1$ ;IF EQUAL YES ↵
        CMPB #HT,R5 ;HT? ↵
        BEQ 1$ ;IF EQUAL YES ↵
-47,50 ↵
3$: MOV R5,R2 ;SET TERMINAL CHARACTER ↵
/ ↵

```

Lines 15 and 16 are deleted and replaced by a corrected line;

Lines 33 through 35 are deleted and replaced by the single line starting with \$CDTD;

Line 38 is replaced;

Lines 43 through 45 are replaced by four text lines;

Lines 47 through 50 are deleted and replaced by the single line beginning with 3\$:.

The output file is created with the name CATB.ABC; the list file (CATB.DAT) is written to the volume mounted on DK1: prior to being spooled to the line printer. The input file, CATB.ZYX, remains in its original form.

Example 4:

```

SLP>GETBK.C46,LP: ↵
        .TITLE GETBK ↵
@CMCOD ↵
;GET ALLOCATED BLOCKS ↵
; ↵
/ ↵

```

In Example 4, the user is creating a new file named GETBK.C46. The first text line defines the program title. The next line is an edit control line, which refers SLP to an indirect file named CMCOD.COMD with the default file type CMD. When indirect input is complete, SLP returns to the terminal for further input at the line following @CMCOD.

6.10 SLP ERROR MESSAGES

SLP error messages are issued in two different formats:

- SLP followed by two dashes and the error message. If applicable, the command line in error is printed on the next line.
- SLP followed by two dashes, the error message and the offending filename.

Examples

SLP -- SYNTAX ERROR
RICKSFILE.MAC,LP:=SHIRLEY.MAC;2
or
SLP -- OPEN FAILURE LINE LISTING FILE filename

SLP error messages, descriptions, and suggested user actions are as follows.

SLP -- COMMAND SYNTAX ERROR
command line

Description

The command line format does not conform to syntax rules. This is a fatal error; the currently opened files are closed and SLP is reinitialized.

Suggested User Action

Reenter the command line.

SLP -- ILLEGAL DEVICE NAME
command line

Description

The device specified is not a legal device. This is a fatal error; it causes the editing session to be reinitialized.

Suggested User Action

Reenter the command line.

SLP -- ILLEGAL DIRECTORY
command line segment

Description

The directory is not legally specified. This is a fatal error; it causes the editing session to be reinitialized.

Suggested User Action

Reenter the command line.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

SLP -- ILLEGAL ERROR/SEVERITY CODE p1 p2 p3

Description

This error message indicates an error in the SLP program.

Suggested User Action

Reenter the command line. If the error persists, contact your DEC field support representative.

SLP -- ILLEGAL FILE NAME
command line segment

Description

A file specification is greater than 30 characters in length or contains a wild card (i.e., an asterisk in place of a file specification element). This is a fatal error; it causes the editing session to be reinitialized.

Suggested User Action

Reenter the command line.

SLP -- ILLEGAL GET COMMAND LINE ERROR

Description

The system, for some reason, is unable to read a command line. This indicates an internal system failure or an error in the SLP program.

Suggested User Action

Reenter the command line. If the error persists, contact your DEC field support representative.

SLP -- ILLEGAL SWITCH
command line segment

Description

The switch is not a valid SLP switch or a legal switch is used in an invalid manner. This is a fatal error; it causes the editing session to be reinitialized.

Suggested User Action

Reenter the command line with the correct switch specified.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

SLP -- INDIRECT COMMAND SYNTAX ERROR
command line

Description

The command line format specified for the indirect file does not conform to syntax rules. This is a fatal error; the currently-opened files are closed and SLP is reinitialized.

Suggested User Action

Reenter the command line.

SLP -- INDIRECT FILE DEPTH EXCEEDED
command line

Description

More than one level of indirection has been specified in a command file. This is a fatal error; the currently-opened files are closed, and SLP is reinitialized.

Suggested User Action

Correct the command file and reenter the command line.

SLP -- I/O ERROR COMMAND INPUT FILE

or

SLP -- I/O ERROR COMMAND OUTPUT FILE

or

SLP -- I/O ERROR CORRECTION INPUT FILE filename

or

SLP -- I/O ERROR LINE LISTING FILE filename

or

SLP -- I/O ERROR SOURCE OUTPUT FILE filename

Description

One of the following conditions may exist:

1. A problem exists on the physical device (e.g., device cycled down).
2. Length of command line is greater than 80 characters.
3. File is corrupted or the format is incorrect.

Suggested User Action

1. Determine which of the above condition exists.
2. Rectify the condition.
3. Reenter the command line.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

SLP -- INDIRECT FILE OPEN FAILURE
command line

or

SLP -- OPEN FAILURE CORRECTION INPUT FILE filename

or

SLP -- OPEN FAILURE LINE LISTING FILE filename

or

SLP -- OPEN FAILURE SOURCE OUPUT FILE filename

Description

One of the following conditions may exist:

1. The file is protected against an access.
2. A problem exists with the physical device (e.g., device not on-line).
3. The volume is not mounted.
4. The specified file directory does not exist.
5. The named file does not exist in the specified directory.
6. The available Executive dynamic memory is insufficient for the operation.

These are fatal errors; they causes the editing session to be reinitialized.

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

SOURCE LANGUAGE INPUT PROGRAM (SLP)

SLP -- PREMATURE EOF CORRECTION INPUT FILE filename

Description

An out-of-range line number has been specified in a correction file or from the terminal, e.g., -1000 has been specified for an 800 line file.

Suggested User Action

1. Terminate the current editing session.
2. Restart the editing session, entering the correct line number.

SLP -- PREMATURE EOF COMMAND INPUT FILE

Description

This is caused by typing CTRL/Z at the terminal, which sends an end-of-file to SLP before the / is read. SLP types out SLP>, indicating that a new file specification is expected.

Suggested User Action

Restart the editing session at the point where the CTRL/Z was inadvertently typed.

CHAPTER 7

LIBRARIAN UTILITY PROGRAM (LBR)

7.1 INTRODUCTION TO LBR

The Librarian Utility Program (LBR) allows the user to create, update, modify, list, and maintain object and macro library files. A library file is a direct access file containing one or more modules of the same module type. Library files are organized for rapid access by the Task Builder and MACRO-11 Assembler.

The Librarian and library files, working in conjunction with the MACRO-11 Assembler and the Task Builder, provide fast entry point search time, easy update with minimal copying of entire files, and the ability to handle multiple module types.

Library files contain two directory tables; an entry point table (EPT) that contains entry point names, and a module name table (MNT) that contains module names.

Both the EPT and MNT are alphabetically ordered. Object module names are derived from .TITLE directives, while entry point names are derived from defined global symbols. Once an entry point is located, its associated module can be accessed directly.

Macro module names are derived from .MACRO directives; macro entry point names are not applicable.

7.1.1 Format of Library Files

A library file consists of a header, an entry point table, a module name table, the library modules, and (usually) free space. The entry point table has zero length for macro libraries. See Figure 7-1.

LIBRARIAN UTILITY PROGRAM (LBR)

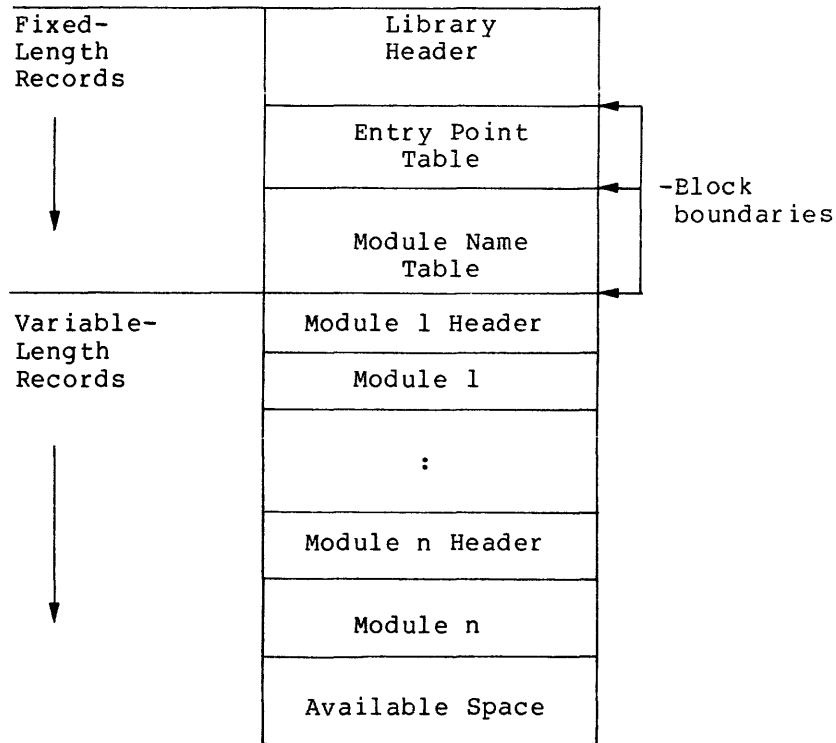


Figure 7-1
General Library File Format

7.1.2 Library Header

The header section is a full block in which the first 23 words are used to describe the current status of the library. Its contents are updated as the library is modified, so the Librarian can access the information it needs to perform its functions (Insert, Compress, etc.). See Figure 7-2.

7.1.3 Entry Point Table

The entry point table consists of 4-word elements containing an entry point name (words 0-1), and a pointer to the module header where the entry point is defined (words 2-3). See Figure 7-3. This table is searched when a library module is referenced by one of its entry points. The table is sequenced in order of ascending entry point names. The entry point table is not used for macro library files.

LIBRARIAN UTILITY PROGRAM (LBR)

OFFSET

WORD	0	NON ZERO ID	LIBRARY TYPE
	2	LBR (LIBRARIAN) VERSION	
	4	(.IDENT FORMAT)	
	6		YEAR
	10	DATE AND	MONTH
	12	TIME LAST	DAY
	14	INSERT	HOUR
	16		MINUTE
	20		SECOND
	22	RESERVED	SIZE EPT ENTR'S
	24	EPT STARTING RELATIVE BLOCK	
	26	NO. EPT ENTRIES ALLOCATED	
	30	NO. EPT ENTRIES AVAILABLE	
	32	RESERVED	SIZE MNT ENTR'S
	34	MNT STARTING REL BLOCK	
	36	NO. MNT ENTRIES ALLOCATED	
	40	NO. MNT ENTRIES AVAILABLE	
	42	LOGICALLY DELETED	
	44	AVAILABLE (BYTES)	
	46	CONTIGUOUS SPACE	
	50	AVAILABLE (BYTES)	
	52	NEXT INSERT RELATIVE BLOCK	
	54	START BYTE WITHIN BLOCK	

Figure 7-2
Contents of Library Header

LIBRARIAN UTILITY PROGRAM (LBR)

WORD	0	GLOBAL SYMBOL	
	1	NAME (RAD50)	
	2	ADDRESS OF	RELATIVE BLK.
	3	MODULE HEADER	BYTE IN BLOCK

Figure 7-3
Format Of Entry Point Table Element

7.1.4 Module Name Table

The module name table is searched when the library module is referenced by its module name, rather than by one of its entry points. It is comprised of 4-word elements; a module name (words 0-1) and a pointer to the module header (words 2-3). See Figure 7-4. The module name table is sequenced in order of ascending module names.

WORD	0	MODULE NAME	
	1	(RAD50)	
	2	ADDRESS OF	RELATIVE BLK.
	3	MODULE HEADER	BYTE IN BLOCK

Figure 7-4
Format of Module Name Table Element

7.1.5 Module Header

Each module starts with an 8-word header, identifying the type and status of the module, its length (number of words), etc. See Figure 7-5.

For object modules, the low-order bit of the attributes byte is set if the module has the selective search attribute. Also, for object modules, the two words of type-dependent information contain the module identification defined by the .IDENT directive at assembly time. For macro modules, these two fields are undefined.

LIBRARIAN UTILITY PROGRAM (LBR)

OFFSET FROM
START OF
MODULE HEADER

0	ATTRIBUTES	STATUS	0=NORMAL MODULE 1=DELETED MODULE
2	SIZE OF		
4	MODULE (BYTES)		
6	DATE	YEAR	
10	MODULE INSERTED	MONTH	
12		DAY	
14	TYPE DEPENDENT		
16	INFORMATION		

Figure 7-5
Module Header Format

7.2 INITIATING LBR

All RSX-11M utilities can be initiated in several ways. These methods are described in Section 1.2. The methods for LBR are:

```
>_LBR ↵
>_LBR command string ↵
>RUN ...LBR ↵
>RUN ...LBR/UIC=[group,member] ↵
>RUN $LBR ↵
>RUN $LBR/UIC=[group,member] ↵
```

7.3 LBR COMMAND STRING

LBR accepts command strings in the following general format:

```
outfile[,listfile]=infile-1[,infile-2,...,infile-n]
```

LBR allows only one level of indirect command file. For a complete description of file specifiers, see Section 1.3; for a description of indirect files, see Section 1.4.

7.4 DEFAULTS IN LBR FILE SPECIFIERS

Defaults in LBR file specifiers are described in Table 7-1.

LIBRARIAN UTILITY PROGRAM (LBR)

Table 7-1
Defaults in LBR File Specifiers

Specifier	Default
dev:	<p><u>Output File</u> SY0:</p> <p><u>Listing File</u> The device which was specified for the output file; otherwise, the default for the output file.</p> <p><u>Input File</u> For the first input file specifier, SY0:.</p> <p>For the second through n input file specifiers, the device specified in the previous input file specifier; otherwise, the default for the previous input file specifier.</p>
[uic]	<p><u>Output File</u> The UIC under which LBR is currently running.</p> <p><u>Listing File</u> The UIC which was specified for the output file; otherwise, the default for the output file specifier.</p> <p><u>Input File</u> For the first input file specifier, the UIC under which LBR is currently running.</p> <p>For the second through n input file specifiers, the UIC specified in the previous input file specifier; otherwise, the default for the previous input file specifier.</p>
filename	No default. Must be specified.
.type	<p><u>Output File</u> Depends on the default in effect (see Section 7.5.4), except when the /CR or /CO switch is specified (see Sections 7.5.1 or 7.5.2, respectively).</p> <p><u>Listing File</u> .LST</p> <p><u>Input File</u> Refer to the descriptions of /CO (Section 7.5.1), /IN (Section 7.5.6), and /RP (Section 7.5.8) switches.</p>
;ver	Latest version of the file, or latest version plus one for the output file when the /CO or /CR switches are specified.
/switch	<p><u>Output File</u> /IN (Insert)</p> <p><u>List File</u> /SP/LI (spool and list module names)</p> <p><u>Input File</u> None.</p>

LIBRARIAN UTILITY PROGRAM (LBR)

7.5 LBR FILE OPTION SWITCHES

LBR file options are in the form of switches appended to file specifiers. These option switches are summarized in Table 7-2.

Table 7-2
LBR File Option Switches

Option	Switch	Function
Compress	/CO	Compress a library file.
Create	/CR	Create a library file.
Delete	/DE	Delete a library module and all of its entry points.
Default	/DF	Specify the default library file type.
Delete Global	/DG	Delete a library module entry point.
Insert	/IN	Insert a module.
List	/LI	List module names.
	/LE	List module names and module entry points.
	/FU	List module names and full module description.
Replace	/RP	Replace a module.
	/-RP	Don't replace a module.
Spool	/SP	Spool the listing for printing.
	/-SP	Don't spool the listing.
Selective Search	/SS	Set selective search attribute in module header.
Squeeze	/SZ	Reduce the size of macro source.

7.5.1 Compress Switch (/CO)

FUNCTION

The Compress Switch provides the user with a facility for rearranging a file by physically deleting all logically deleted records, putting all free space at the end of the file, and making the free space available for new library module inserts. Additionally, the library table specification may be altered for the resulting library. LBR accomplishes this by creating a new file that is a compressed copy of the old library file.

LIBRARIAN UTILITY PROGRAM (LBR)

NOTE

The old library file is not deleted after the new file is created.

The /CO switch can be appended only to the output file specifier.

FORMAT

outfile/CO:size:ept:mnt = infile

where:

outfile is the file specifier for the file that is to become the compressed version of the input file.

NOTE

Default type is .OLB if input file is an object library or .MLB if input file is a macro library.

/CO is the Compress switch.

:size is the size of the new library file in 256-word blocks. If omitted, the size of the old library file is the default size.

:ept is 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. If omitted, the number of EPT's in the old library file is the default value. This parameter is always forced to zero for macro libraries.

NOTE

Maximum number of entries is 4096(10).

:mnt is 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. If omitted, the number of MNT's in the old library file is the default value.

NOTE

Maximum number of entries is 4096(10).

infile is the file specifier of the library file to be compressed.

LIBRARIAN UTILITY PROGRAM (LBR)

NOTE

Default file type is .OLB for object libraries and .MLB for macro libraries. The actual default type is determined by the current default library type (see Section 7.5.4).

EXAMPLE

```
LBR>RICKLIB/CO:100.:128.:64.=SHEILA.OLB↵
```

In this example, file SHEILA.OLB is compressed, and a new file, RICKLIB.OLB, is created with the following attributes:

size = 100(10) blocks

ept = 128(10) entry points

mnt = 64(10) module names

NOTES

1. The new file, RICKLIB.OLB, received a version number that is one version greater than the latest version for the file.
2. Both files, RICKLIB.OLB and SHEILA.OLB, reside in the default directory file on SY0:.

7.5.2 Create Switch (/CR)

FUNCTION

The Create switch provides the user with a facility for allocating a contiguous library file on a direct access device (e.g., disk). It initializes the Library file header, the entry point table, and the module name table.

The /CR switch can be appended only to the output file specifier.

FORMAT

```
outfile/CR:size:ept:mnt:type
```

where:

outfile is the file specifier for the library file being created. The default file type is .OLB if an object library is being created, or .MLB if a macro library is being created.

/CR is the create switch.

:size is the size of the library file in 256-word blocks. The default size is 100(10) blocks.

:ept is the number of entry point table (EPT) entries to allocate. The default value is 512(10) for

LIBRARIAN UTILITY PROGRAM (LBR)

object libraries. This parameter is always forced to zero for macro libraries.

NOTE

Maximum number of entries is 4096(10).

:mnt is the number of module name table (MNT) entries to allocate. The default value is 256(10).

NOTE

Maximum number of entries is 4096(10).

:type is the type of library to be created. Acceptable types are OBJ for object libraries and MAC for macro libraries. The default is the last value specified or implied with the /DF switch (see Section 7.5.4), or OBJ if /DF has not been specified.

NOTE

The EPT and MNT are automatically filled out to the next disk block boundary, if the values specified are not multiples of 64(10).

EXAMPLE

LBR>RICKLIB/CR::128.:64.:OBJ=SHEILA,LAURA,JENNY ↵

In this example, a combination of functions are performed. First, the library file RICKLIB.OLB is created in the default directory on SY0:; RICKLIB has the following attributes:

size = 100(10) blocks (default size),

ept = 128(10) entry points,

mnt = 64(10) module names.

type = .OBJ

Second, object modules from the input files SHEILA.OBJ, LAURA.OBJ, and JENNY.OBJ, which reside in the default directory on SY0:, are inserted into the newly created library file. Insert is the default switch for input files (see Section 7.5.6).

LIBRARIAN UTILITY PROGRAM (LBR)

7.5.3 Delete Switch (/DE)

FUNCTION

The Delete switch provides the user with a facility for deleting library modules and their associated entry points (global symbols) from a library file. Up to 15 library modules and their associated entry points can be deleted with one delete command.

When LBR begins processing the /DE switch, it prints the following message on the initiating terminal:

MODULES DELETED:

As modules are logically deleted from the library file, the module name is printed on the initiating terminal. See the example at the end of this section.

If a specified library module is not contained in the library file, a message is printed on the initiating terminal, and the processing of the current command is terminated. This message is as follows:

LBR -- *FATAL* - NO MODULE NAMED "name"

The /DE switch can be appended only to the library file specifier.

NOTE

When LBR deletes a module from a library file, the module is not physically removed from the file, but is marked for deletion. This means, that although the module is no longer accessible, the file space that the module once occupied is not available for use (unless the deleted module is the last module which was inserted). To physically remove the module from the file and make the freed space available for use, the user must compress the library (see Section 7.5.1).

FORMAT

outfile/DE:module-1[:module-2:...:module-n]

where:

outfile is the file specifier for the library file.

/DE is the delete switch.

:module is the name of the module to be deleted.

LIBRARIAN UTILITY PROGRAM (LBR)

EXAMPLE

```
LBR>RICKLIB/DE:SHEILA:LAURA:JENNY ↵
```

MODULES DELETED:

SHEILA

LAURA

JENNY

In this example, the modules SHEILA, LAURA, and JENNY and their associated entry points are deleted from the latest version of library file SY0:RICKLIB.OLB.

7.5.4 Default Switch (/DF)

FUNCTION

The Default switch provides the user with a facility for specifying the default library file type. Acceptable values are OBJ for object libraries and MAC for macro libraries. A default value of OBJ is used by LBR to process the /DF switch.

Specifying a default value:

1. Sets the default type argument for the Create switch (/CR).
2. Provides a file type default value of .MLB for macro libraries and .OLB for object libraries when opening an output (library) file, except in the cases of /CO and /CR. When /CO is specified, the default applies to the library input file. When /CR is specified, the default type is .OLB if an object library is being created, or .MLB if a macro library is being created. The /DF switch only affects the name of the file to be opened; thereafter, the library header record information is used to determine the type of library file being processed.

The /DF switch can be issued alone or appended to a library file specifier.

FORMAT

outfile/DF:type...

or

/DF:type

where:

outfile is the file specifier for the library file.

/DF is the Default switch.

type is OBJ for object library files and MAC for macro library files.

LIBRARIAN UTILITY PROGRAM (LBR)

NOTE

If a type other than OBJ or MAC is specified, the current default library type will be set to object libraries, and the following message will be displayed:

LBR -- INVALID LIBRARY TYPE SPECIFIED

EXAMPLES

1. LBR>/DF:MAC↵
LBR>RICKLIB=infile↵

File RICKLIB.MLB is opened for insertion.
2. LBR>/DF:MAC↵
LBR>RICKLIB/DF:OBJ=infile↵

File RICKLIB.OLB is opened for insertion.
3. LBR>/DF:MAC↵
LBR>RICKLIB/CR↵

Macro library RICKLIB.MLB is created.
4. LBR>/DF:MAC↵
LBR>RICKLIB/CR:::OBJ↵

Object library RICKLIB.OLB is created.
5. LBR>/DF↵
LBR>TEMP/CO=RICKLIB↵

RICKLIB.OLB is opened for compression. If RICKLIB.OLB is an object library, the file TEMP.OLB is created to receive the compressed output. If RICKLIB.OLB is a macro library (a nonstandard use of the type OLB), the file TEMP.MLB is created.
6. LBR>/DF:OBJ↵
LBR>TEMP/CO=RICKLIB.MLB↵

Assuming that file RICKLIB.MLB is a macro library, the macro library file TEMP.MLB is created to receive the compressed output.

7.5.5 Delete Global Switch (/DG)

FUNCTION

The Delete Global switch provides the user with a facility for deleting a specified entry point (global symbol) from the EPT. Up to 15 entry points may be deleted with one command. This command does not affect the object module which contains the actual symbol definition.

LIBRARIAN UTILITY PROGRAM (LBR)

When LBR begins processing the /DG switch, it prints the following message on the initiating terminal:

ENTRY POINTS DELETED:

As entry points are deleted from the library file, the entry point is printed on the initiating terminal. See the example at the end of this section.

If a specified entry point is not contained in the EPT, a message is printed on the initiating terminal, and the processing of the current command is terminated. This message is as follows:

LBR -- *FATAL* - NO ENTRY POINT NAMED "name"

The /DG switch can only be appended to the library file specifier.

FORMAT

outfile/DG:global-1[:global-2:...:global-n]

where:

outfile is the library file specifier.

/DG is the Delete Global switch.

global is the name of the entry point to be deleted.

EXAMPLE

LBR>RICKLIB/DG:SHEILA:LAURA:JENNY ↵

ENTRY POINTS DELETED:

SHEILA

LAURA

JENNY

In this example, the entry points SHEILA, LAURA and JENNY are deleted from the latest version of the library file named SY0:RICKLIB.OLB.

7.5.6 Insert Switch (/IN)

FUNCTION

The Insert switch provides the user with a facility for inserting library modules into a library file. Any number of input files can be specified, and each file can contain any number of concatenated input modules. For macro libraries, only first-level macro definitions are extracted from the input files. All text outside of the first-level macro definitions is ignored. The /IN switch is the default library file option, and can be appended only to the library file specifier.

LIBRARIAN UTILITY PROGRAM (LBR)

If the user attempts to insert an input module which already exists in the library file, the following message is printed on the initiating terminal:

```
LBR -- *FATAL* DUPLICATE MODULE NAME "name" IN filename
```

Likewise, if the user attempts 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-1[,infile-2,...,infile-n]
```

where:

outfile is the file specifier for the library file into which the input modules are to be inserted. The default type depends on the current default (see Section 7.5.4). It is .OLB if the current default is object libraries or .MLB if the current default is macro libraries.

/IN is the Insert switch.

infile is the file specifier for the input file containing modules to be inserted into the library file. The default type is .OBJ if outfile is an object library and .MAC if outfile is a macro library.

EXAMPLE

```
LBR>RICKLIB/IN=SHEILA,LAURA,JENNY ↵
```

In this example, the modules contained in the latest versions of files SHEILA, LAURA and JENNY, which reside in the default directory on SY0:, are inserted into the latest version of the library file RICKLIB, which also resides in the default directory on SY0:. The default 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.

7.5.7 List Switches (/LI, /LE, /FU)

FUNCTION

The List switches provide the user with a facility for producing a printed listing of the contents of a library file. Three switches allow the user to select the type of listing desired. These switches are as follows:

/LI Produces a listing of the names of all modules in the library file.

/LE Produces a listing of the names of all modules in the library file and their corresponding entry points.

LIBRARIAN UTILITY PROGRAM (LBR)

/FU Produces a listing of the names of all modules in the library file and gives a full module description for each: i.e., size, date of insertion, and module-dependent information.

NOTE

Appendix B.1 contains sample listings of all three types of library listing.

These switches can be appended only to the output file specifier or the list file specifier.

FORMAT

outfile[,listfile]/switch(es)

where:

outfile is the file specifier for the library file whose contents is to be listed.

listfile is the optional listing file specifier. If not specified, the listing is directed to the initiating terminal.

/switch(es) is the list option(s) selected.

NOTE

The /LI switch is the default value, and need not be specified when a listing file has been specified, or when any other list switch is included in the command.

EXAMPLES

1. LBR>RICKLIB/LI ↵

In this example, a listing of the names of all the modules contained in file SY0:RICKLIB.OLB is printed on the initiating terminal.

2. LBR>RICKLIB/LE ↵

In this example, a listing of the names of all the modules and their entry points (contained in file SY0:RICKLIB.OLB) is printed on the initiating terminal.

3. LBR>RICKLIB/FU ↵

In this example, a listing of the names of all the modules, and a full description of each module contained in file SY0:RICKLIB.OLB, is printed on the initiating terminal.

LIBRARIAN UTILITY PROGRAM (LBR)

4. LBR>DK1:[200,200]RICKLIB,LP:/LE/FU ↵

In this example, a listing of the names of all the modules, their entry points, and a full description of each module for file RICKLIB, residing in directory [200,200] on DK1:, is printed on the line printer.

7.5.8 Replace Switch (/RP)

FUNCTION

The Replace switch provides the user with a facility for replacing 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.

When a match occurs on a module name, the existing module is logically deleted, and all of its entries are removed from the EPT. The /RP switch does not imply module replacement on matching entry point names. That condition is always fatal.

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.

FORMAT

The /RP switch can be specified in either of the following formats:

- a. Global format - The /RP switch is appended to the file specifier, and all of the input files are assumed to contain modules to be replaced.
- b. Local format - The /RP switch is appended to an input file specifier, and only the file to which the /RP switch is appended is considered to contain modules to be replaced.

Global Format

outfile/RP=infile-1[,infile-2[,...,infile-n]

where:

outfile is the file specifier for the library file. The default type depends on the current default (see Section 7.5.4). It is .OLB if the current default is object libraries or the .MLB if the current default is macro libraries.

LIBRARIAN UTILITY PROGRAM (LBR)

`/RP` is the Replace switch.

`infile` is the input file specifier for the file that contains modules to be replaced in the library file. The default type is `.OBJ` if `outfile` is an object library, or `.MAC` if it is a macro library.

This format of the `/RP` switch allows the user to specify a list of input files without having to append the `/RP` switch to each of them.

NOTE

Should the user want 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), the user can append `/-RP` or `/NORP` to the desired input file specifier.

Local Format

`outfile=infile-1[/RP][,infile-2[/RP],...,infile-n[/RP]]`

where:

`outfile` is the file specifier for the library file. The local format default is the same as the global format default described above.

`infile` is the input file specifier for the file that contains modules to be inserted or replaced in the output library file. The local format default is the same as the global format default described above.

`/RP` is the Replace switch.

NOTE

Appending the `/RP` switch to an input file specifier constitutes the local format of the switch. This overrides the LBR default (Insert) and instructs LBR to treat the module(s) contained in the specified file as modules to be replaced.

LIBRARIAN UTILITY PROGRAM (LBR)

EXAMPLES

The files used in the following four examples, and the modules contained within each file, are depicted in Figure 7-6. For the examples, these files are assumed to reside in the default directory on the default device, and the initial state of the library file is assumed to be as shown in Figure 7-6.

1. LBR>RICKLIB/RP=SHEILA,LAURA,JENNY ↵

```
MODULE "SHEILA" REPLACED  
MODULE "LAURA1" REPLACED  
MODULE "LAURA2" REPLACED  
MODULE "JENNY1" REPLACED  
MODULE "JENNY2" REPLACED
```

In this example, the global format for the /RP switch is used. Object modules from the input files SHEILA, LAURA, and JENNY replace modules by the same names in the library file named RICKLIB. The resulting library file is shown in Figure 7-7.

2. LBR>RICKLIB=CHRIS,SHEILA/ RP ↵

```
MODULE "SHEILA" REPLACED
```

In this example, the local format of the /RP switch is used. The object module SHEILA from file SHEILA is replaced in the library file RICKLIB. The object modules in the file CHRIS are inserted in the library file. (See Insert switch in Section 7.5.6.) The resulting library file is shown in Figure 7-8.

	Output Library File	Input Files			
File Name	RICKLIB.OLB;1	SHEILA.OBJ;1	LAURA.OBJ;1	JENNY.OBJ;1	CHRIS.OBJ;1
Object	JENNY1	SHEILA	LAURA1	JENNY1	CHRIS1
Modules	JENNY2		LAURA2	JENNY2	CHRIS2
	LAURA1		LAURA3	JENNY3	
	LAURA2				
	SHEILA				

Figure 7-6
Sample Files Used in LBR Examples

LIBRARIAN UTILITY PROGRAM (LBR)

```

RICKLIB.OLB;1

    JENNY1
    JENNY2
    JENNY3      *
    LAURA1
    LAURA2
    LAURA3      *
    SHEILA

*These modules did not exist on the
library file prior to the execution of
this example, but they did exist on the
input files.  LBR, therefore, assumed
that they were to be inserted.  Since
LBR handled these modules as a normal
insert, no message was printed on the
input terminal.
    
```

Figure 7-7
Output Library File After Execution of Example 1

```

RICKLIB.OLB;1

    CHRIS1      **
    CHRIS2      **
    JENNY1
    JENNY2
    LAURA1
    LAURA2
    SHEILA      *

    *This module replaced
    **These modules inserted
    
```

Figure 7-8
Output Library File After Execution of Example 2

LIBRARIAN UTILITY PROGRAM (LBR)

3. LBR>RICKLIB/RP=SHEILA,LAURA,JENNY,CHRIS/-RP ↵

```

MODULE "SHEILA" REPLACED
MODULE "LAURA1" REPLACED
MODULE "LAURA2" REPLACED
MODULE "JENNY1" REPLACED
MODULE "JENNY2" REPLACED
    
```

In this example, the /-RP switch is used to override the global format of the command. Object modules in files SHEILA, LAURA, and JENNY are processed as modules to be replaced, and file CHRIS is processed as a file which contains modules to be inserted. The resulting library file is shown in Figure 7-9.

RICKLIB.OLB;1	
CHRIS1	**
CHRIS2	**
JENNY1	
JENNY2	
JENNY3	*
LAURA1	
LAURA2	
LAURA3	*
SHEILA	

*These modules were inserted by default.

**These modules were specified to be inserted. Had a module of the same name been present, a fatal error message would have been issued. See Example 4 below.

Figure 7-9
Output Library File After Execution of Example 3

4. LBR>RICKLIB/RP=SHEILA,LAURA/-RP,JENNY ↵

```

MODULE "SHEILA" REPLACED
LBR -- *FATAL* -- DUPLICATE MODULE "LAURA1" IN LAURA.OBJ;1
    
```

In this example, only module SHEILA from file SHEILA was replaced. The user specified that the modules in file LAURA not be replaced (/ -RP), but inserted. One of the modules contained in file LAURA duplicated an already existing module in file RICKLIB (see Figure 7-6). Therefore, LBR issued the fatal error message and terminated the processing of the current command.

LIBRARIAN UTILITY PROGRAM (LBR)

7.5.9 Spool Switch (/SP)

The Spool switch is the list file default switch. Whether the switch is specified or not, the results are the same, i.e., the listing file is spooled to the line printer. The listing file can be spooled to any file-structured device (e.g., disk).

After the listing file is created, a request is made to the print spooler task to print the spooled file; printing is performed asynchronously (see Appendix C for a description of the spooler task).

The automatic printing of the listing file can be inhibited by specifying a minus sign (-) or the letters NO between the slash (/) and the SP in the spool switch (/SP or /NOSP). This causes the listing file to be created, but the request to the print spooler task is not issued. Therefore, the file is not automatically printed.

The /SP switch can only be appended to the list file specifier.

FORMAT

outfile,listfile[/SP] or [/-SP]

where:

outfile is the file specifier for the library file.

listfile is the listing file specifier.

/SP or /-SP is the Spool switch.

EXAMPLE

LBR>RICKLIB/DE:SHEILA,RICKLST/-SP ↵

In this example, the following occurs:

1. The module SHEILA and its associated entry points are deleted from the library file SY:RICKLIB.
2. The listing of the contents of resulting library file RICKLIB is written to the list file SY:RICKLST.LST. Since the /-SP switch is specified, the file is not automatically printed.

7.5.10 Selective Search Switch (/SS)

FUNCTION

The Selective Search switch is used to set the selective search attribute bit in the module header of object modules as they are inserted into an object library. The switch has no effect when applied to modules being inserted into a macro library. The switch may be specified only on input files for insertion or replacement operations, and it affects all modules in the input file to which it is applied.

Object modules with the selective search attribute are given special treatment by the Task Builder. Global symbols defined in modules with the selective search attribute are only included in the Task Builder's

LIBRARIAN UTILITY PROGRAM (LBR)

symbol table if they are previously referenced by other modules. Thus, only referenced symbols will be listed with the module in the Task Builder memory allocation file, thereby reducing task build time. The /SS switch should only be applied to object files whose modules contain only absolute (not relocatable) symbol definitions. See the RSX-11M Task Builder Reference Manual for more information.

FORMAT

outfile=infile-1[/SS[,infile-2[/SS],...,infile-n[/SS]]

where:

outfile is the file specifier for the library file.
infile is the file specifier for the input file that contains modules to be selectively searched.
/SS is the Selective Search switch.

7.5.11 Squeeze Switch (/SZ)

FUNCTION

The Squeeze switch provides the user with a facility for reducing the size of macro definitions by eliminating all trailing blanks and tabs, blank lines, and comments from macro text. The /SZ switch is used to conserve memory in the MACRO-11 Assembler and to reduce the size of macro library files. The Squeeze switch has no effect on object libraries.

FORMAT

The /SZ switch can be specified in either of the following formats:

1. Global format - The /SZ switch is appended to the library file specifier, and all of the input files are assumed to contain modules to be squeezed.
2. Local format - The /SZ switch is appended to an input file specifier, and only the file to which the /SZ switch is appended is considered to contain modules to be squeezed.

Global Format

outfile/SZ=infile-1[,infile-2,...,infile-n]

where:

outfile is the file specifier for the library file.
/SZ is the Squeeze switch.
infile is the file specifier for the input file that contains modules to be squeezed before being inserted into the library file.

This format of the /SZ switch allows the user to specify a list of input files without having to append the /SZ switch to each of them.

LIBRARIAN UTILITY PROGRAM (LBR)

NOTE

Should the user want 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), the user can append `/-SZ` or `/NOSZ` to the desired input file specifier.

Local Format

```
outfile=infile-1/SZ[,infile-2[/SZ],...,infile-n[/SZ]]
```

where:

outfile	is the file specifier for the library file.
infile	is the file specifier for the file that contains modules to be squeezed before being inserted into the library file.
/SZ	is the Squeeze switch.

NOTE

LBR uses the following algorithm on each line to be squeezed and inserts the resultant 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.

EXAMPLE

Figure 7-10 illustrates the use of the LBR `/SZ` switch. A file containing input text to be squeezed is illustrated, along with the text actually inserted into the library file after the squeeze operation has been completed.

LIBRARIAN UTILITY PROGRAM (LBR)

```

                                BEFORE BEING SQUEEZED

                                .MACRO  MOVSTR RX,RY,?LBL

;***      - - NOTE :                               ;
;          BOTH ARGUMENTS MUST BE REGISTERS        ;

LBL:      MOVB      (RX)+,(RY)+      ;MOVE A CHARACTER
          BNE      LBL              ;CONTINUE UNTIL NULL SEEN
          DEC      RY               ;BACKUP OUTPUT PTR TO NULL

;END OF MOVSTR
          .ENDM

                                AFTER BEING SQUEEZED

                                .MACRO  MOVSTR RX,RY,?LBL

;***      - - NOTE :
;          BOTH ARGUMENTS MUST BE REGISTERS
LBL:      MOVB      (RX)+,(RY)+
          BNE      LBL
          DEC      RY
          .ENDM

```

Figure 7-10
MACRO Listing Before and After Running LBR with /SZ Switch

7.6 COMBINING LIBRARY FUNCTIONS

Two or more library functions may be requested in the same command line. The only exceptions are that COMPRESS cannot be requested with anything else except LIST, and CREATE and DELETE cannot be specified in the same command line.

Functions are performed in the following order:

1. /DF
2. /CR or /CO
3. /DE
4. /DG
5. /IN, /RP, /SS, /SZ
6. /LI, /LE, /FU

EXAMPLE

LBR>FILE/DE:XYZ:\$A,LP:/LE/FU=MODX,MODY/RP↵

LIBRARIAN UTILITY PROGRAM (LBR)

Functions are performed in order, as:

Delete modules XYZ and \$A.

Insert all modules from MODX and MODY, replacing any duplicates of modules in MODY.

Produce a listing of the resultant library file on the line printer with full module descriptions and all entry points.

7.7 LBR CONSTRAINTS

The following constraints apply to LBR:

1. Limit of 65,536(10) words per module.
2. Limit of 65,536(10) blocks per library,
3. Tables should be allocated to maximum anticipated size. Expanding table allocations requires using Compress to copy the entire file.
4. A fatal error results if an attempt is made to insert a module into a library which contains a differently named module with the same entry point. See Insert command, Section 7.5.6.
5. The use of "wild cards" in file specifiers is not allowed (i.e., forms such as *.OBJ, where the "*" is used to indicate "all modules with type .OBJ").

7.8 LBR ERROR MESSAGES

Error messages reported to the user by LBR are of two types: diagnostic and fatal.

Diagnostic error messages inform the user that a condition exists that requires consideration, but the nature of the condition does not warrant termination of the command. Diagnostic messages are issued to TI:, in the format:

```
LBR -- *DIAG* - message
```

Fatal error messages inform the user that a condition exists that caused LBR to terminate the processing of a command. When this occurs, LBR returns to the highest level of command input. For example, if the command is entered in response to the MCR prompt, i.e.,

```
>LBR command
```

then, LBR issues the fatal error message and exits. If, however, the command is entered in response to the LBR prompt, i.e.,

```
LBR>command
```

LBR issues the fatal error message and reprompts.

LIBRARIAN UTILITY PROGRAM (LBR)

Fatal error messages are issued to TI: in the format:

LBR -- *FATAL* - message

NOTE

If a fatal error occurs during the processing of an indirect command file, the command file is closed, the fatal error message, followed by the command line in error, is issued to TI:, and LBR returns to the highest level of command input.

7.8.1 Effect of Fatal Errors on Library Files

The status of a library file after fatal errors is:

1. In general, output errors leave the library in an indeterminate state.
2. During the deletion process, the library is rewritten prior to the printing of the individual module-/entry-point-deleted messages.
3. During the replacement process, the library is rewritten prior to the printing of the individual module-replaced messages.
4. During the insert process, the library is rewritten after the insertion of all modules in each individual input file, i.e., between input files.

7.8.2 Error Messages

LBR -- ILLEGAL GET COMMAND LINE ERROR CODE

Description

The system, for some reason, is unable to read a command line. This is an internal system failure.

Suggested User Action

Reenter the command line. If the problem persists, consult software support representative.

LBR -- INPUT ERROR ON filename

Description

The file system, while attempting to process an input file, has detected an error.

LIBRARIAN UTILITY PROGRAM (LBR)

A problem exists with the physical device (e.g., device cycled down).

Suggested User Action

Reenter the command line.

LBR -- COMMAND SYNTAX ERROR
command line

Description

The user has entered a command in a format that does not conform to syntax rules.

Suggested User Action

Reenter the command line, using the correct syntax.

LBR -- OUTPUT ERROR ON filename

Description

A write error has occurred on the output file. One of the following conditions may exist:

1. The volume is full.
2. The device is write-protected.
3. The hardware has failed.

Suggested User Action

If the volume is full, the user should delete all unnecessary files and rerun LBR.

If the device is write-protected, the user should write-enable the device, and reenter the command line.

If the hardware has failed, the user can swap devices and reenter the command line or wait until the device is repaired and rerun LBR.

LBR -- ILLEGAL SWITCH
command line

Description

The user specified a non-LBR switch or a legal switch in an invalid context.

Suggested User Action

Reenter the command line with the correct switch specification.

LIBRARIAN UTILITY PROGRAM (LBR)

LBR -- INSUFFICIENT DYNAMIC MEMORY TO CONTINUE

Description

The partition in which LBR is running is too small.

Suggested User Action

Remove the task (LBR), install it in a larger partition, and reenter the command line.

LBR -- INVALID LIBRARY TYPE SPECIFIED

Description

The user specified an illegal library type in a CREATE or DEFAULT command. The file types OBJ and MAC are the only valid specifications. See Sections 7.5.2 and 7.5.4.

Suggested User Action

Reenter the command line with OBJ or MAC specified.

LBR -- COMMAND I/O ERROR

Description

One of the following conditions may exist:

1. A problem exists on the physical device (e.g., not cycled up).
2. The file is corrupted or the format is incorrect (e.g., record length exceeds 132 bytes).

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

LBR -- INDIRECT FILE OPEN FAILURE
command line

Description

The requested indirect command file does not exist as specified. One of the following conditions may exist:

1. The user directory area is protected against access.
2. A problem exists on the physical device (e.g., device cycled down).
3. The volume is not mounted.
4. The specified file directory does not exist.

LIBRARIAN UTILITY PROGRAM (LBR)

5. The file does not exist as specified.
6. Insufficient dynamic memory in Executive.

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

LBR -- INDIRECT COMMAND SYNTAX ERROR
command line

Description

The user specified an indirect file in a format that does not conform to syntax rules.

Suggested User Action

Reenter the command line with the correct syntax.

LBR -- BAD LIBRARY HEADER

Description

Either the file is not a library file or the file is corrupted.

Suggested User Action

1. If the file is not a library file, reenter the command line with a proper library file specified.
2. If the file is a proper library file, the user should run the file structure verification utility (VFY) against the volume to determine if it is corrupted (see Chapter 8).
3. If the volume is corrupted, it must be reconstructed before it can be used.

LBR -- INDIRECT FILE DEPTH EXCEEDED
command line

Description

An attempt has been made to exceed one level of indirect command files.

Suggested User Action

Rerun the job with only one level of indirect command file.

LIBRARIAN UTILITY PROGRAM (LBR)

LBR -- I/O ERROR ON INPUT FILE filename

Description

A read error has occurred on an input file. One of the following conditions may exist:

1. A problem exists on the physical device (e.g., not cycled up).
2. The file is corrupted or the format is wrong (record length exceeds 132 bytes).

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

LBR -- OPEN FAILURE ON FILE filename

Description

The file system, while attempting to open a file, has detected an error. One of the following conditions may exist:

1. The user directory area is protected against an open.
2. A problem exists on the physical device (e.g., device cycled down).
3. The volume is not mounted.
4. The specified file directory does not exist.
5. The file does not exist as specified.
6. Insufficient contiguous space to allocate the library file (compress and create only).
7. Insufficient dynamic memory in Executive.

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

LBR -- INVALID EPT AND/OR MNT SPECIFICATION

Description

The user, when specifying a /CR or /CO command, entered an EPT or MNT value which was greater than 4096(10).

Suggested User Action

Reenter the command line with the correct value specified.

LIBRARIAN UTILITY PROGRAM (LBR)

LBR -- POSITIONING ERROR ON filename

Description

The device is write-locked.

Suggested User Action

If the device is write-locked, write enable it and reenter the command line.

LBR -- EPT OR MNT EXCEEDED IN filename

Description

The EPT or MNT table limit has been reached during the execution of an Insert or Replace command.

Suggested User Action

1. Copy the library, increasing the table space via the COMPRESS command.
2. Reenter the command line.

LBR -- DUPLICATE MODULE NAME "name" IN filename

Description

An attempt has been made to insert (without replacement) a module into a library that already contains a module with the specified name.

Suggested User Action

1. Determine if the specified input file is the correct file.
2. If the input file is correct, the user must decide whether to delete the duplicate module from the library file and insert the new one, or replace the duplicate module by rerunning LBR with the /RP switch appended to the input file specifier.

LBR -- GET TIME FAILED

Description

This error occurs when LBR attempts to execute a Get Time Parameters directive and fails. The error is caused by a system malfunction.

Suggested User Action

Reenter the command line. If the problem persists, consult software support representative.

LIBRARIAN UTILITY PROGRAM (LBR)

LBR -- NO MODULE NAMED "module"

Description

The user has attempted to delete a module that is not in the specified library file.

Suggested User Action

1. Determine if the module name is misspelled or if the wrong library file is specified.
2. Reenter the command line with the module name correctly specified.

LBR -- INVALID NAME -- "name"

Description

A module name or entry point that contains a non-Radix-50 character has been specified for deletion.

NOTE

Radix-50 characters consist of the letters A through Z, the numbers 0 through 9, and the special characters period (.) and dollar sign (\$).

Suggested User Action

Reenter the command line with a valid name.

LBR -- LIBRARY FILE SPECIFICATION MISSING

Description

The user has entered a command without specifying the library file.

Suggested User Action

Reenter the command line with the library file specified.

LBR -- ILLEGAL SWITCH COMBINATION

Description

The user has specified switches that cannot be executed in combination. See Section 7.6.

Suggested User Action

Reenter the command line, specifying the switches in the proper sequence.

LIBRARIAN UTILITY PROGRAM (LBR)

LBR -- NO ENTRY POINT NAMED "name"

Description

The user has attempted to delete an entry point that is not in the specified library file.

Suggested User Action

1. Determine if the entry point is misspelled or if the wrong library file is specified.
2. Reenter the command line with the entry point correctly specified.

LBR -- DUPLICATE ENTRY POINT NAME "name" IN filename

Description

An attempt has been made to insert a module into a library file when both contain an identically-named entry point.

Suggested User Action

1. Determine if the specified input file is the correct file. If not, reenter the command line, specifying the correct input file.
2. If the input file is the correct file, the user may delete the duplicate entry point from the library and rerun.

LBR -- TOO MANY OUTPUT FILES SPECIFIED

Description

The user has specified more than two output files; LBR makes the following assumptions:

1. The first output file specified is the output library file.
2. The second output file specified is the listing file.
3. The third through n files specified to the left of the equal sign are ignored.

Suggested User Action

No action is required. LBR continues as though the extra file(s) were not specified.

LBR -- EXACTLY ONE INPUT FILE MUST APPEAR WITH /CO

Description

The user has specified no file or more than one input library file in the /CO command.

LIBRARIAN UTILITY PROGRAM (LBR)

Suggested User Action

Reenter the command line with only one input file specified.

LBR -- FATAL COMPRESS ERROR

Description

The user's input library file is corrupted or is not a library file.

Suggested User Action

No recovery is possible. The file in question must be reconstructed.

LBR -- EPT OR MNT SPACE EXCEEDED IN COMPRESS

Description

The user has specified an EPT or MNT table size for the output library file that is not large enough to contain the EPT or MNT entries used in the input library file.

Suggested User Action

Reenter the command line with a larger EPT or MNT table size specified.

LBR -- ERROR IN LIBRARY TABLES, FILE filename

Description

The library file is corrupted or is not a library file.

Suggested User Action

If the file is corrupted, no recovery is possible; the file must be reconstructed.

If the file is not a library file, the user should reenter the command line with the correct library file specified.

LBR -- INVALID FORMAT, INPUT FILE filename

Description

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.

Suggested User Action

Reenter the command line with the correct input file specified.

LIBRARIAN UTILITY PROGRAM (LBR)

LBR -- OPEN FAILURE ON LBR WORK FILE

Description

The file system, while attempting to open the LBR work file, has detected an error.

NOTE

The LBR work file is created on the volume from which LBR was installed.

One of the following conditions may exist:

1. The volume is full.
2. The device is write-protected.
3. A problem exists with the physical device.
4. Insufficient dynamic memory in Executive.

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

LBR -- MARK FOR DELETE FAILURE ON LBR WORK FILE

Description

When LBR begins processing commands, it automatically creates a work file and marks it for delete. For some reason, this marking for delete failed.

The work file constitutes a lost file, because it does not appear in any file directory.

Suggested User Action

The file may be deleted by running the file structure verification utility (VFY) (see Chapter 8).

LBR -- ILLEGAL FILENAME command line

Description

The user has entered one of the following:

1. A file specifier which contains a wild card.
2. A file specifier which contains neither a filename nor file type.

LIBRARIAN UTILITY PROGRAM (LBR)

Suggested User Action

Reenter the command line correctly.

LBR -- ILLEGAL DEVICE/VOLUME
command line

Description

The user has entered a device specifier that does not conform to syntax rules.

NOTE

A device specifier consists of 2 ASCII characters, followed by one or two optional octal digits.

Suggested User Action

Reenter the command line with the correct device syntax specified.

LBR -- ILLEGAL DIRECTORY
command line

Description

The user has entered a UIC that does not conform to syntax rules.

NOTE

UIC syntax consists of a left square bracket, followed by one to three octal digits, a comma, one to three octal digits, and terminated by a right square bracket.

Suggested User Action

Reenter the command line with the correct UIC syntax.

LBR -- WORK FILE I/O ERROR

Description

A write error has occurred on the LBR work file. One of the following conditions may exist:

1. The volume is full.
2. The device is write-protected.
3. The hardware has failed.

LIBRARIAN UTILITY PROGRAM (LBR)

Suggested User Action

If the volume is full, the user should delete all unnecessary files and rerun.

If the device is write-protected, the user should write enable the device, and reenter the command line.

If the hardware has failed, the user can swap devices and retry the command, or wait until the device is repaired and rerun LBR.

LBR -- VIRTUAL STORAGE REQUIREMENTS EXCEED 65536 WORDS

Description

This error may occur with maximum size libraries in conjunction with a single command line which logically deletes a large number of modules and entry points, and continues to replace them with an equally large number of modules and entry points having highly dissimilar names.

Normally, this message indicates some sort of internal system error.

Suggested User Action

Rerun the job, but divide the complicated command line into several smaller command lines which do the same operations.

CHAPTER 8

FILE STRUCTURE VERIFICATION UTILITY (VFY)

8.1 INTRODUCTION TO VFY

The File Structure Verification Utility (VFY) program provides:

1. The ability to check the readability and validity of a file-structured volume.
2. The ability to print out the number of available blocks on a file-structured volume (/Free).
3. The ability to search for files which are in the index file, but not in any directory, i.e., files which are "lost" in the sense that they cannot be accessed by filename (/LOst) (see RSX-11 I/O Operations Reference Manual for a description of the index file).
4. The ability to list all files in the index file, showing the file ID, filename, and owner (/LIst).
5. The ability to mark as "used" all the blocks that appear to be available, which are actually allocated to a file (/UPdate).
6. The ability to rebuild the storage allocation bit map so that it properly reflects the information in the index file (/REbuild).
7. The ability to restore files that are marked for delete (/DElete).
8. The ability to perform a read check on every allocated block on a file-structured volume (/RC).

NOTES

1. There should be no other activity on the volume; in particular, activities which create new files, extend existing files, or delete files while VFY is running.
2. VFY must not be aborted while a /UP, /RE or /DE command is being processed. Aborting VFY while it is in the process of modifying the storage allocation or index files may seriously endanger the integrity of that volume.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

8.2 INITIATING VFY

All RSX-11M utility programs can be initiated in several ways. These methods are described in Section 1.2. The methods for VFY are:

```
>VFY ↵  
>VFY command string ↵  
>RUN ...VFY ↵  
>RUN ...VFY/UIC=[group,member] ↵  
>RUN $VFY ↵  
>RUN $VFY/UIC=[group,member] ↵
```

VFY normally operates in a read-only mode, assuming that the scratch file, if required, is on another device. VFY requires write-access under the following conditions:

1. If the /UP or /RE switch is used, VFY requires write-access to the storage allocation map ([0,0]BITMAP.SYS).
2. If the /DE switch is specified, VFY requires write-access to the index file ([0,0]INDEXF.SYS).
3. If the /LO switch is specified and lost files are found, VFY requires write-access to the [1,3] user file directory.

VFY may be run under any UIC if only read access is required. If write access is required, VFY must run under a system UIC.

8.3 VFY COMMAND STRING

All commands to VFY are issued by entering a VFY command string through the initiating terminal. The VFY command string is formatted as follows:

```
listfile,scratchdev=indev/switch  
  
or  
  
indev/switch (This is a short form of TI:,indev=indev/switch)
```

where:

listfile specifies the output listing file in the following format:

```
dev:[uic]filename.typ;ver
```

scratchdev specifies the device on which the scratch file produced by VFY is to be written. This parameter is in the following format:

```
dev:
```

FILE STRUCTURE VERIFICATION UTILITY (VFY)

The scratch file is used by VFY during the verification scan and during the lost file scan. It is created but is not entered in a directory. Therefore, it is invisible to the user. The scratch file is automatically deleted upon termination of the VFY program.

NOTE

If the user has reason to suspect that his system disk is of questionable integrity, the scratch file should be forced onto another device by utilizing this parameter.

It is recommended that the scratch file always be assigned to another volume. The scratch file is not used for the /FREE and /LIST commands.

indev specifies the volume to be verified. This parameter is in the following format:

dev:

/switch specifies the function to be performed. This parameter is in the following format:

/sw

The VFY command switches are described in detail in Section 8.4. If no switch is specified, the VFY program performs a validity check.

For a complete description of command strings, see Section 1.3.

8.3.1 Defaults in File Specifiers

Default file specifiers are listed in Table 8-1.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

Table 8-1
VFY Default File Specifiers

Element	Default Value
dev:	<u>Output listing device</u> TI: <u>Scratch file device</u> SY0: <u>Volume to be verified</u> SY0:
[uic]	The UIC under which VFY is currently running.
filename	No default - must be specified.
.typ	No default - must be specified.
;ver	Latest version plus 1.

8.4 VFY COMMAND SWITCHES

VFY commands are specified in the form of switches appended to the VFY command string. Command switches and functions are summarized in Table 8-2.

Table 8-2
VFY Functions and Switches

Function	Switch	Purpose
Validity Check	Null	Check readability and validity of the volume mounted on specified device.
Delete	/DE	Reset marked-for-delete indicators.
Update	/UP	Allocate blocks which appear to be available but have been allocated to a file.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

Table 8-2 (Cont.)
VFY Functions and Switches

Function	Switch	Purpose
Rebuild	/RE	Recover blocks which appear to be allocated but are not contained in a file.
Free	/FR	Print out the available space on a volume.
Lost	/LO	Scan entire file structure looking for files which are not in any directory.
List	/LI	List entire index file by file identification.
Read Check	/RC	Check entire volume to see if every block of every file can be read.

8.4.1 Validity Check

Validity Check (no command switch) checks the readability and validity of the volume mounted on the specified device. This feature entails reading all the file headers in the index file and checking that all the disk blocks referenced in the map area of each file header are marked as allocated in the bit map (i.e., allocated to that file).

Rules for running the Validity Check:

1. The volume to be checked must be mounted as a Files-11 structured volume, as follows:

>MOU dev: ↵

2. The volume may be write-protected if:
 - a. It is not the system volume; or
 - b. The required scratch file is directed to another file-structured volume.

When the validity check is completed, a listing of the results is printed. This output is described in Section 8.4.1.1.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

8.4.1.1 File Error Reporting - After the volume has been verified, and the normal output messages have been printed, error conditions are reported. All errors for a given file are preceded by a file identification line that identifies the file in error. This line is formatted as follows:

```
FILE ID nn,nn filename.type;version OWNER [g,m]
```

where:

nn,nn is the unique file identification number assigned to the file by the system at file-creation time.

filename is the user filename.

.type is the file type (i.e., OBJ for object file).

;version is the version number of the file.

[g,m] is the UIC which owns the file.

This file identification line is followed by one or more of the following messages:

I/O ERROR READING FILE HEADER-ERROR CODE -32

Failed to read the file header for the specified file ID.

BAD FILE HEADER

Software checks on the validity of the file header indicate that the header has been corrupted.

MULTIPLE ALLOCATION n,n

The specified (double precision) logical block number is allocated to more than one file. If this error occurs, a second pass is automatically taken which will indicate all files that share each multiply allocated block. The second pass is taken after all file headers have been checked (see Section 8.4.1.3).

BLOCK IS MARKED FREE n,n

The specified logical block number is allocated to the indicated file but is not marked as allocated in the storage allocation map. (see Section 8.4.1.4).

BAD BLOCK NUMBER n,n

The specified block number was found in the header for this file but is illegal for the device (out of range). This indicates a corrupted file header.

FILE IS MARKED FOR DELETE

This indicates that a system failure occurred while the specified file was being deleted. The deletion was not completed and the file header still exists (see Section 8.4.1.2).

FILE STRUCTURE VERIFICATION UTILITY (VFY)

HEADER MAP OUT OF SYNC

This indicates an error in the header map area which also indicates a corrupted file header.

The last error message for the file is followed by a summary line for that file, as follows:

SUMMARY: MULT=nn, FREE=nn, BAD=nn.

where:

MULT	is the number of multiple block allocations.
FREE	is the number of blocks marked free that should have been allocated.
BAD	is the number of bad retrieval pointers in the file header.

NOTE

If the output for VFY is directed to a terminal device, and the user does not wish to see all the error messages for a given file entering CTRL/O terminates the listing of all further error messages for that particular file, i.e., all messages but the summary line.

8.4.1.2 Files Marked-for-Delete -- If a file has been marked-for-delete but the deletion process was not completed, the user has two options: the file can be restored, if still required, and its consistency checked, or the deletion process can be completed to recover the lost space. These operations are described below.

- Restoring a File

To restore a file marked-for-delete, the disk volume must be mounted using the MCR MOUNT command with the /UNL switch specified. For example:

```
>MOU DK0:/UNL↵
```

Then, run VFY specifying the /DE switch to reset the marked-for-delete indicators in file headers. Once the delete indicator has been reset, run VFY specifying the /LO switch to scan the entire file structure.

NOTE

The deletion process may have proceeded partially and a portion at the end of the file may be missing. This condition can be detected by a directory listing obtained using the PIP /FU switch (see Section 2.4.8).

FILE STRUCTURE VERIFICATION UTILITY (VFY)

- Deleting a File

Files that are marked-for-delete can be deleted directly with PIP, once their unique File ID has been obtained via a validity check. The File ID appears as the first entry in the file identification line which precedes each list of file errors (see Section 8.4.1.1). The following example illustrates how the File ID is used with PIP to delete a file:

Example:

```
>PIP /FI:12:20/DE↵
```

In this example, the file with File ID 12,20 is deleted 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-delete; however, the file is completely deleted.

Once files have been restored or deleted, run VFY with the /RE switch specified to assure the consistency of the volume's storage allocation map.

8.4.1.3 Deletion Of Multiply Allocated Blocks - If the file structure contains multiply allocated blocks, it is necessary to delete files until there are no more such blocks. An automatic rescan of the volume identifies which files share which blocks. This rescan lists the first as well as subsequent files containing the multiply allocated blocks. Once the user has this information, he must then determine which, if any, of the files can be saved and delete the rest, using the Delete function provided by the PIP utility.

NOTE

Extreme caution should be taken in deleting multiply allocated files. After the files have been deleted, VFY should be run once again to ensure that all of the multiply allocated files have been deleted.

8.4.1.4 Elimination Of Free Blocks - Once there are no multiply allocated blocks, the next concern is the elimination of blocks that are marked FREE in the storage allocation map, but which are actually allocated to a file. To cause these blocks to be reallocated in the storage allocation map, the user must rerun the validity check specify the /UP switch. This allocates all blocks that should have been marked as allocated. See Section 8.4.3 for a description of the /UP switch.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

NOTE

Once there are no multiply allocated blocks and no blocks marked free that are actually in use, the file structure is safe for writing new files and extending existing files. However, if there were such errors, there may be files which have had data blocks overwritten as the result of multiple allocation.

8.4.1.5 Recovering Lost Blocks - The user can determine whether any blocks have been lost on a file-structured volume by examining the last two lines of output from the validity check. The last two lines of output give the free space on the volume. The first line of the two tells how much room is available according to the index file (i.e., the number of blocks that are not in use by any file in the index file). The last line specifies how much room is available according to the storage allocation map. Assuming there are no other errors, these two figures should agree. If the index file indicates that more blocks are free than the storage allocation map, then those blocks are "lost" in the sense that they appear to be allocated, but no file contains them. Lost blocks may be recovered by rerunning the validity check specifying the /RE switch. See Section 8.4.4 for a description of the /RE switch.

8.4.2 DELETE Switch (/DE)

FUNCTION

The DELETE switch allows the user to reset the marked-for-delete indicators in the file header area of those files which are marked for deletion, but which were never actually deleted.

FORMAT

listfile,scratchdev=indev/DE

or

indev/DE

FILE STRUCTURE VERIFICATION UTILITY (VFY)

NOTES

1. The volume must be mounted with the /UNL switch.
2. VFY must be running under a system UIC.

8.4.3 UPDATE Switch (/UP)

FUNCTION

The UPDATE switch allows the user to allocate all blocks that appear to be available but are actually allocated to a file.

FORMAT

listfile,scratchdev=indev/UP

or

indev/UP

NOTES

1. Files with multiply allocated blocks must be deleted from the file structure before the update can be run.
2. The volume being updated must be write-enabled.
3. VFY must be running under a system UIC.
4. The scratch file should be on another volume. If this is impossible, the volume must be dismounted immediately after VFY terminates. (Failure to do this may result in partial updating of the storage allocation map.) Then the volume should be mounted again, and the scratch file must be deleted manually. VFY issues a detailed message in this case specifying the name of the scratch file to be deleted.

The message is:

```
VFY -- TO COMPLETE THE STORAGE MAP
      UPDATE DISMOUNT THE VOLUME
      IMMEDIATELY. THEN MOUNT IT
      AND DELETE THE FOLLOWING
      FILE: [g,m] filespec
```

FILE STRUCTURE VERIFICATION UTILITY (VFY)

where:

[g,m]	is the UIC.
filespec	is the name of the file to be deleted.

8.4.4 REBUILD Switch (/RE)

FUNCTION

The REBUILD switch allows the user to recover blocks that are lost in the sense that they appear to be allocated, but no file contains them.

FORMAT

listfile,scratchdev=indev/RE

or

indev/RE

NOTES

1. Multiply allocated blocks must be removed (deleted) from the file structure before the rebuild can be run.
2. The volume being updated must be write-enabled.
3. VFY must be running under a system UIC.
4. The scratch file should be on another volume. If this is impossible, the volume must be dismounted immediately after VFY terminates. (Failure to do this may result in partial updating of the storage allocation map.) Then the volume should be mounted again, and the scratch file must be deleted manually. VFY issues a detailed message in this case, specifying the name of the scratch file to be deleted.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

8.4.5 FREE Switch (/FR)

FUNCTION

The FREE switch provides the user with the ability to print out the available space on a specified volume.

FORMAT

listfile=indev/FR

or

indev/FR

The output from the /FR command is shown below:

dev: HAS nnnn. BLOCKS FREE, nnnn. BLOCKS USED OUT OF nnnn.

8.4.6 LOST Switch (/LO)

FUNCTION

The LOST switch provides the facility to scan the entire file structure looking for files which are not in any directory and, thus, are lost in the sense that they cannot be referenced by filename. A list of the files is produced, and if the "lost file directory" [1,3] exists on that volume, all the files will be entered in that directory.

FORMAT

listfile,scratchdev=indev/LO

or

indev/LO

8.4.7 LIST Switch (/LI)

FUNCTION

The LIST switch provides the facility to list the entire index file by file identification. The output for each file specifies the file number, file sequence number, filename, and owner. A typical index file listing is illustrated in Figure 8-1.

FORMAT

listfile,scratchdev=indev/LI

or

indev/LI

FILE STRUCTURE VERIFICATION UTILITY (VFY)

```
VFY>DK:/LI ↵
LISTING OF INDEX ON DK0:

FILE ID 000001,000001 INDEXF.SYS;1  OWNER [1,1]
FILE ID 000002,000002 BITMAP.SYS;1  OWNER [1,1]
FILE ID 000003,000003 BADBLK.SYS;1  OWNER [1,1]
FILE ID 000004,000004 000000.DIR;1  OWNER [1,1]
FILE ID 000005,000005 CORIMG.SYS;1  OWNER [1,1]
FILE ID 000006,000006 001001.DIR;1  OWNER [1,1]
FILE ID 000007,000007 001002.DIR;1  OWNER [1,2]
FILE ID 000010,000010 EXEMC.MLB;1  OWNER [1,1]
FILE ID 000011,000011 RSXMAC.SML;1  OWNER [1,1]
FILE ID 000012,000012 NODES.TBL;1  OWNER [1,1]
FILE ID 000013,000036 QIOSYM.MSG;311  OWNER [1,2]
FILE ID 000014,000037 F4PCOM.MSG;1  OWNER [1,2]
```

Figure 8-1
VFY Listing Sample Using the /LI Switch

8.4.8 READ CHECK Switch (/RC)

FUNCTION

The READ CHECK switch provides the facility to check that every block of every file on a specified volume can be read.

FORMAT

listfile=indev/RC[:n]

or

indev/RC[:n]

NOTE

Since the READ CHECK is a read-only operation, the volume can be write-protected.

The optional parameter [:n] is the blocking factor which indicates the number of file blocks to be read at a time. The default value is the maximum number of blocks in dynamic memory available to VFY.

The dynamic memory available may be increased by installing VFY in a larger partition. Five blocks are available when VFY is installed in an 8K partition, and four blocks are added for each 1K increment.

For the fastest possible read check, the maximum block factor should be used. Whenever an error is encountered, each block of the portion-in-error is reread individually to determine which data block(s) cannot be read.

When an error is detected, a file identification line is listed in the following format:

FILE STRUCTURE VERIFICATION UTILITY (VFY)

FILE ID nn,nn filename.typ;ver. blocks used/blocks allocated

Following this line, an error message is listed. If a blocking factor other than 1 is in use, an error message in the following form will be issued:

ERROR STARTING AT VBN n1,n2 LBN n1,n2 - ERROR CODE -err

Following the first error message, there should be one or more error messages indicating the exact block(s) in error. The second error message line(s) will be in the following form:

ERROR AT VBN n1,n2 LBN n1,n2 - ERROR CODE -err

If an "ERROR STARTING AT" line is displayed without one or more "ERROR AT" lines, a multiblock read operation on the selected device has failed, but the data blocks appear to be individually readable.

NOTES

1. If the VBN of the unreadable block listed in the "ERROR AT" line is beyond the block-used-count, the data portion of the file is all right.
2. The negative number printed after the ERROR CODE message is -4 to indicate a device parity error. Other error codes are contained in Appendix I of the RSX-11 I/O Operations Reference Manual.

8.5 VFY ERROR MESSAGES

VFY -- COMMAND SYNTAX ERROR

Description

The command entered does not conform to command syntax rules.

Suggested User Action

Reenter the command line with the correct syntax specified.

VFY -- FAILED TO ALLOCATE SPACE FOR TEMP FILE

Description

The volume specified for the temporary scratch file is full.

Suggested User Action

Use PIP to delete all unnecessary files and rerun VFY.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

VFY -- FAILED TO ATTACH DEVICE

or

VFY -- FAILED TO DETACH DEVICE

or

VFY -- ILLEGAL DEVICE

Description

The file specifier entered contains an illegal device.

Suggested User Action

Reenter the command line with the correct device specified.

VFY -- ILLEGAL SWITCH

Description

The switch specified is not a valid VFY switch or a valid switch is used illegally.

Suggested User Action

Reenter the command line with the correct switch specified.

VFY -- I/O ERROR ON INPUT FILE

or

VFY -- I/O ERROR ON OUTPUT FILE

Description

One of the following conditions may exist:

1. The device is not on-line.
2. The device is not mounted.
3. The hardware has failed.

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

VFY -- NO DYNAMIC MEMORY AVAILABLE - PARTITION TOO SMALL

Description

VFY does not have enough buffer space to run.

Suggested 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

Description

One of the following conditions may exist:

1. VFY is not running under a system UIC, but should be.
2. The named file does not exist in specified directory.
3. The volume is not mounted.
4. The specified file directory does not exist.

Suggested User Action

1. Determine which of the above conditions exists.
2. Rectify the condition.
3. Reenter the command line.

8.6 VFY ERROR CODES

If VFY cannot access the message file, errors are reported in the following format:

VFY -- ERROR CODE nn.

where:

nn. is one of the error codes contained in Table 8-3.

Refer to Section 8.5 for error descriptions and suggested user actions.

FILE STRUCTURE VERIFICATION UTILITY (VFY)

Table 8-3
VFY Error Codes

ERROR CODES	VFY ERROR MESSAGE IS:
1.	ILLEGAL DEVICE
2.	OPEN FAILURE ON BIT MAP
3.	OPEN FAILURE ON TEMPORARY FILE
4.	FAILED TO ALLOCATE SPACE FOR TEMP FILE
5.	FAILED TO DETACH DEVICE
6.	FAILED TO ATTACH DEVICE
7.	COMMAND SYNTAX ERROR
8.	I/O ERROR ON INPUT FILE
9.	I/O ERROR ON OUTPUT FILE
10.	ILLEGAL SWITCH
11.	OPEN FAILURE ON LISTING FILE
12.	OPEN FAILURE ON INDEX FILE
13.	NO DYNAMIC MEMORY AVAILABLE - PARTITION TOO SMALL

APPENDIX A
COMMANDS AND SWITCHES

A.1 INTRODUCTION

This appendix presents a summary of the commands and/or switches used by the RSX-11M utilities described in this manual. Each of the numbered sections of this appendix corresponds, in number, to the chapter discussing that utility. For example, Chapter 2 and Section A.2 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 PIP COMMAND SUMMARY

APPEND

outfile[/FO]=infile-1 [,infile-2,...,infile-n]/AP[/FO] where /FO is File Owner	Opens an existing file (outfile) and appends the input file(s) onto the end of it.
--	--

COPY AND MERGE

outfile[/switch]=infile-1 [,infile-2,...,infile-n] [/switch] /switch=BL:n[.]	Creates a copy of a file on the same or another device.
CO	Block allocated.
-CO	Contiguous output.
	Non-contiguous output.
FO	File Owner.
NV	See Table 2-3 for a complete description of these switches.
SU	New Version.
	Supersede.

DEFAULT

dev:[group,member]/DF	Changes the default device and/or UIC.
-----------------------	--

DELETE

infile-1[,infile-2,...,infile-n]/DE	Deletes files.
-------------------------------------	----------------

COMMANDS AND SWITCHES

ENTER

outfile=infile-1
[,infile-2,...,infile-n]/EN[/NV]
where /NV is New Version.

Enters a synonym for file in a directory with an option to force the version number of "outfile" to one greater than the latest version for the file.

FREE

dev:/FR

Prints out the available space on a volume.

IDENTIFY

/ID

Causes the version of PIP currently in use to be displayed on the terminal.

LIST

[listfile]=infile-1[,...,infile-n]/LI
where [listfile] defaults to
TI: if not specified.

Lists one or more directories with an option to specify directory listing formats.

Alternate Mode Switches

/BR Brief format.
/FU[:n] Full format.
/TB Total blocks format.

For a complete description of these switches, see Table 2-4.

PROTECT

infile-1/PR[/SY[:RWED] [/OW[:RWED]]
[/GR[:RWED]] [/WO[:RWED]] [/FO]
where SY is system access rights.
OW is owner access rights.
GR is group access rights.
WO is world access rights.

Alters file protection. See Section 2.4.9 for a complete description of these switches.

RWED is read, write, extend,
delete privilege.
FO is File Owner subswitch.

PURGE

infile-1[,infile-2,...,infile-n]/PU[:n]

Deletes a specified range of obsolete versions of a file.

REMOVE

infile-1[,infile-2,...,infile-n]/RM

Removes an entry from a directory file.

COMMANDS AND SWITCHES

RENAME

outfile=infile-1
[,infile-2,...,infile-n]/RE
[/NV]
where NV is New Version.

Changes the name of a file with an option to force the version number of "outfile" to one greater than the latest version for the file.

SPOOL

infile-1[,infile-2,...,infile-n]
/SP

Specifies a list of files to be printed.

UNLOCK

infile-1[,infile-2,...,infile-n]
/UN

Unlocks a file which was locked as a result of being improperly closed.

UPDATE

outfile=infile-1
[,infile-2,...,infile-n]/UP[/FO]
where FO is File Owner.

Opens an existing file(s) (infile) and writes it, from the beginning, onto outfile.

A.3 FLX COMMAND SUMMARY

The FLX commands have the following format:

outfile=infile-1[,infile-2,...,infile-n]
/switch

Performs file conversion between DOS-11, RT-11 and Files-11 formats.

where switch = BL:n

Indicates the number of contiguous blocks to be allocated to the output file.

BS:n

Specifies the block size for cassette tape output.

CO

Indicates that the output file is to be contiguous.

DE

Deletes files from a DOS-11 or RT-11 volume.

DI

Causes a directory listing of DOS or RT volumes; or DOS or RSX cassette tape volumes to be listed.

COMMANDS AND SWITCHES

DO	Identifies the file as a DOS-11 formatted file.
FA:n	Formatted ASCII.
FB:n	Formatted binary.
FC	Indicates that FORTRAN carriage control conventions are to be used.
ID	Requests the current version number of FLX.
IM:n	Image mode.
LI	Same as DI.
NU:n	Used with /ZE and /RT switches to specify the number of directory blocks to allocate.
RS	Indicates that file is a Files-11 formatted file.
RT	Indicates that file is an RT-11 formatted file.
SP	Indicates that the converted file is to be spooled via the print spooler.
UI	Indicates that the output file is to have the same UIC as the input file.
VE	Verify after write (for cassette only).
ZE	Initializes DOS and RT volumes and cassettes for DOS or RSX files.

See Tables 3-2 and 3-3 for a complete description of these switches.

COMMANDS AND SWITCHES

A.4 DMP COMMAND SUMMARY

The DMP utility has one command.

outfile=infile/switch	Dumps a file onto outfile.
where switch = AS	Data should be dumped in ASCII mode.
BA:n:m	Specifies a base block address.
BL:n:m	Specifies the first and last logical blocks to be dumped.
BY	Data should be dumped in byte octal format.
HD	Includes the file header in the data dumped.
ID	Causes the current version of DMP to be printed on the listing.
LB	Causes starting (logical) block number and a contiguous or non-contiguous indication for the file to be printed.
MD[:n]	Controls line number sequencing during a memory image dump.

See Table 4-1 for a complete description of these switches.

A.5 EDI COMMAND SUMMARY

ADD
A[DD] (string) Add the text specified by "string" to the end of the current line.

ADD AND PRINT
AP (string) Same as ADD, except the new current line is printed.

COMMANDS AND SWITCHES

BEGIN

B[EGIN]

Sets the current line pointer to the top of the block buffer or input file.

BLOCK ON or OFF

BL[OCK][ON] or [OFF]

Switch text access modes.

BOTTOM

BO[TTOM]

Sets the current line pointer to the bottom of block buffer or input file.

CHANGE

[n]C[HANGE] /string-1/string-2

Search for string-1 in the current line and replace it with the text specified in string-2. The integer n allows the user to repeat the command, thus allowing string-2 to be substituted for string-1 n times.

CLOSE

CL[OSE] filespec

Transfer the remaining lines in the block buffer and the input file into the output file, then close both the input file and the output file.

CLOSES

CLOSES

Close secondary input file and begin selecting lines from the input file.

CLOSE AND DELETE

CDL filespec

Same as the CLOSE command, except that the input file is deleted.

COMMANDS AND SWITCHES

CONCATENATION CHARACTER

CC character

Change command concatenation character to the specified character (default is &).

CTRL/Z

↑Z

Same as EXIT if in Edit mode; otherwise, it causes an immediate exit of EDI.

DELETE

D[ELETE] [n] or [-n]

Delete the current and next n-1 lines, if n is positive; delete n lines preceding the current line, but not the current line, if n is negative.

DELETE AND PRINT

DP [n] or [-n]

Same as DELETE, except that the new current line is printed out.

END

E [ND]

Same as the BOTTOM command.

ERASE

ERASE [n]

Erase the entire block buffer, the current line, and the next n blocks.

EXIT

EX[IT]

Same as CLOSE command, except that, when files are closed, EDI exits.

EXIT AND DELETE

ED[X] filespec

Exit from the editing session, close the output file, delete the input file.

FORM FEED

FF

Insert form feed into block buffer.

COMMANDS AND SWITCHES

<u>FILE</u> FI[LE] filespec	Transfer lines from the input file to the file specified by filespec.
<u>FIND</u> [n]F[IND] (string)	Find the line starting with "string" or, if n is specified, the nth line starting with "string".
<u>INSERT</u> I[NSERT] (string)	Insert "string" immediately following the current line. If "string" is null, EDI enters Input mode.
<u>KILL</u> KILL	Terminate this editing session; close input and output files; delete the output file.
<u>LINE CHANGE</u> [n]LC /string-1/string-2	Same as CHANGE, except that all occurrences of string-1 in the current line are changed to string-2.
<u>LIST ON TERMINAL</u> LI[ST]	Print on user terminal all lines in block buffer or all remaining lines in input file, starting with current line.
<u>LIST ON PSEUDO-DEVICE</u> LP	List the text in the block buffer or input file on the pseudo-device CL:, starting with the current line.
<u>LOCATE</u> [n]L[OCATE] string	Search the block buffer for "string" or, if n is specified, the nth occurrence of "string".

COMMANDS AND SWITCHES

<u>MACRO</u> MA[CRO] x definition	Define macro x to be definition".
<u>MACRO CALL</u> MC[ALL]	Retrieve macros from the latest version of file MCALL;n.
<u>MACRO EXECUTE</u> [n]Mx [a]	Execute macro x for n executions, passing it the numeric argument a.
<u>MACRO IMMEDIATE</u> [n]<definition>	Allows the user to define and execute a macro n times in one step.
<u>NEXT</u> N[EXT] [n] or [-n]	Establish a new current line + or - n lines from the current line.
<u>NEXT PRINT</u> NP [n] or [-n]	Same as Next command, but the new current line is printed.
<u>OLD PAGE</u> OL[DPAGE] n	Back up to page n.
<u>OPENS</u> OP[ENS] filespec	Open secondary input file.
<u>OUTPUT ON or OFF</u> OU[TPUT] [ON] or [OFF]	Turn output on or off.
<u>OVERLAY</u> O[VERLAY] [n]	Delete the current line and the next n-1 lines, and enter Input mode.
<u>PAGE</u> PAG[E] [n]	Enter block mode, if not already in block mode, and read page n into the block buffer.

COMMANDS AND SWITCHES

PAGE FIND

[n]PF[IND] (string)

Identical to FIND command, except that it searches successive pages until the nth occurrence of "string" is found.

PAGE LOCATE

[n]PL[OCATE] (string)

Same as LOCATE command, except that successive pages are searched for the value specified by "string".

PASTE

PA[STE] /string-1/string-2

Same as the LINE CHANGE command, except that all lines in the remainder of the input file or block buffer are searched for string-1. Wherever found, string-1 is replaced with string-2.

PRINT

P[RINT] [n]

Print the next line, and the next n-1 lines, on the terminal.

READ

REA[D] [n]

Read the next n pages into the block buffer.

RENEW

REN[EW] [n]

Write the current buffer and read in the next. If n is specified, repeat n-1 times.

RETYPE

R[ETYPE] (string)

Replace the current line with the text of "string". If "string" is null, the line is deleted.

SAVE

SA[VE] [n] [filespec]

Save the current line, and the next n-1 lines, in the file specified by filespec.

COMMANDS AND SWITCHES

SEARCH & CHANGE

SC /string-1/string-2

Search for string-1, in the block buffer or input file starting with the line following the current line. When string-1 is found, replace all occurrences in line with string-2.

SELECT PRIMARY

SP

Select primary input file.

SELECT SECONDARY

SS

Select secondary input file.

SIZE

SIZE n

Specify maximum number of lines to be read into the block buffer on a single READ.

TAB ON or OFF

TA[B] [ON] or [OFF]

Turn automatic tabbing on or off.

TOP

T[OP]

Same as BEGIN command.

TOP OF FILE

TOF

Return to the top of the input file, in block mode, and save all pages previously edited.

TYPE

TY[PE] [n]

Same as PRINT command, except that the current line pointer does not change.

UNSAVE

UNS[AVE] [filespec]

Retrieve the lines which were previously saved on filespec and insert them immediately following the current line.

UPPER CASE ON or OFF

UC [ON] or [OFF]

Turn upper case conversion on or off.

COMMANDS AND SWITCHES

VERIFY ON or OFF

V[ERIFY] [ON] or [OFF]

Allows user to select whether or not locative and change commands are to be verified.

WRITE

W[RITE]

Write the current block to the output file, and erase the contents of the buffer.

A.6 SLP COMMAND SUMMARY

The SLP utility has only one command.

outfile[,listfile/SP or/-SP]=infile
[/switch]

Perform batch-oriented editing to create and maintain source language files on disk.

where switch = AU and -AU

Enable and disable the editing audit trail, which indicates the changes made during the most recent editing session.

BF and -BF

Enable and disable blank fill when an audit trail is being produced.

DB and -DB

Enable and disable double-spaced listing.

and where SP and -SP

Enable and disable the spooling of listing files to a file structured volume.

See Table 6-2 for a complete description of these switches and Table 6-3 for a description of the SLP edit control characters.

COMMANDS AND SWITCHES

A.7 LBR COMMAND SUMMARY

COMPRESS

outfile/CO:size:ept:mnt:=infile

Creates a new library file and transfers contents, but physically deletes logically deleted records in the file and puts all free space at the end of the file.

CREATE

outfile/CR:size:ept:mnt:type

Allocates a contiguous library file on a direct access device.

DELETE

outfile/DE:module-1
[:module-2:....:module-n]

Deletes library modules and their associated entry points from a file.

DEFAULT

outfile/DF:type...
or
/DF:type

Specifies default library file type.

DELETE GLOBAL

outfile/DG:global-1
[:global-2:....:global-n]

Deletes specified library module entry points from a file.

INSERT

outfile[/IN]=infile-1
[,infile-2,....,infile-n]

Inserts library modules into a library file.

LIST

outfile[,listfile]/switch(es)
where /switch(es)= LI

Lists all modules in the library file.

LE

Lists all modules in the library file and all their entry points.

FU

Lists all modules in the library file and provides a full module description including size, date of insertion, and version.

COMMANDS AND SWITCHES

REPLACE

outfile/RP=infile-1
[,infile-2,...,infile-n]

or

outfile=infile-1[/RP]
[,infile-2[/RP],...,infile-n[/RP]]

Inserts, and in certain cases, replaces library modules in a library file.

SPOOL

outfile,listfile/SP

The listing file is spooled out for printing.

SELECTIVE SEARCH

outfile=infile-1/SS
[,infile-2[/SS],...,
infile-n[/SS]]

Sets selective search attribute bit in object module header.

SQUEEZE

outfile/SZ=infile-1
[,infile-2,...,infile-n]

or

outfile=infile-1/SZ
[,infile-2[/SZ],...,
infile-n[/SZ]]

Reduces size of macro sources.

A.8 VFY COMMAND SUMMARY

DELETE

listfile, scratchdev=indev/DE
or indev/DE

Resets the marked-for-delete indicators in the file header area of those files marked for deletion, but which were never actually deleted.

FREE

listfile=indev/FR
or indev/FR

Prints out the available space on a volume.

LIST

listfile, scratchdev=indev/LI
or indev/LI

Lists the entire index file by file identification.

COMMANDS AND SWITCHES

LOST

listfile, scratchdev=indev/LO
or indev/LO

Scans the entire file structure looking for files that are not in any directory.

READ CHECK

listfile=indev/RC[:n]
or indev/RC[:n]

Checks that every block of every file on specified volume can be read.

REBUILD

listfile, scratchdev=indev/RE
or indev/RE

Recovers blocks that appear to be allocated, but which are not contained in any file.

UPDATE

listfile,scratchdev=indev/UP
or indev/UP

Allocates blocks that appear to be available, but which are actually allocated to a file.

APPENDIX B

LBR, EDI AND DMP EXAMPLES

B.1 SAMPLE LISTINGS FOR LBR LIST SWITCHES (OBJECT LIBRARY)

B.1.1 List Module Names

LBR>MAC,LP:↵

or

LBR>MAC,LP:/LI↵

DIRECTORY OF FILE MAC.OLB;1
OBJECT MODULE LIBRARY CREATED BY: LBR VXM2VM
LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
FILE SPACE AVAILABLE: 00015 WORDS

ASGMT
ASSEM
CNDTL
CODHD
DATDR
ENBDS
ENDLN
ENDPS
EXPRS
FLOAT
GETLN
GMARG
INFIL
INIFL
INOFL
LABEL
LISTC
LISTG
MACRO
MACRS
MCALL
MLIBS
MSCDR
NDRCT
PROCSI
PROPC
PROSW
PST
READ
REPT
ROLHD
RSDAT
RSEXEC

R5UNP
 SECTR
 SETDIR
 SETDN
 SETIMM
 SETMX
 SPACE
 STMNT
 SYMBL
 WORDR
 WRITE

B.1.2 List Module Names and Full Module Information

LBR>MAC,LP:/FU ↵

or

LBR>MAC,LP:/LI/FU ↵

DIRECTORY OF FILE MAC.OLR;1
 OBJECT MODULE LIBRARY CREATED BY: LBR VX02VM
 LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50
 MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
 EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
 FILE SPACE AVAILABLE: 00015 WORDS

ASGMT	SIZE:00264	INSERTED:17-JUL-74	IDENT:02
ASSEM	SIZE:00749	INSERTED:1-AUG-74	IDENT:05M
CNDTL	SIZE:00727	INSERTED:31-JUL-74	IDENT:04
CODHD	SIZE:00923	INSERTED:17-JUL-74	IDENT:06
DATDR	SIZE:00414	INSERTED:17-JUL-74	IDENT:07
ENRDS	SIZE:00248	INSERTED:1-AUG-74	IDENT:06
ENDLN	SIZE:00812	INSERTED:31-JUL-74	IDENT:06
ENDPS	SIZE:01060	INSERTED:17-JUL-74	IDENT:04
FXPRS	SIZE:01211	INSERTED:31-JUL-74	IDENT:06
FLOAT	SIZE:00035	INSERTED:17-JUL-74	IDENT:02
GETLN	SIZE:00670	INSERTED:17-JUL-74	IDENT:05
GMBRG	SIZE:00290	INSERTED:17-JUL-74	IDENT:01
INFIL	SIZE:00941	INSERTED:9-SEP-74	IDENT:12
INIFL	SIZE:00493	INSERTED:31-JUL-74	IDENT:01
INOFI	SIZE:00960	INSERTED:9-SEP-74	IDENT:01
LABEL	SIZE:00400	INSERTED:1-AUG-74	IDENT:04
LISTC	SIZE:00284	INSERTED:17-JUL-74	IDENT:04
LSTNG	SIZE:00566	INSERTED:17-JUL-74	IDENT:07
MACRO	SIZE:02019	INSERTED:17-JUL-74	IDENT:013
MACRS	SIZE:01540	INSERTED:9-SEP-74	IDENT:09
MCALL	SIZE:00264	INSERTED:31-JUL-74	IDENT:01
MLIBS	SIZE:00807	INSERTED:31-JUL-74	IDENT:06
MSCDR	SIZE:00843	INSERTED:31-JUL-74	IDENT:08
NDRCT	SIZE:00258	INSERTED:31-JUL-74	IDENT:02
PROCSI	SIZE:00216	INSERTED:17-JUL-74	IDENT:01
PROPC	SIZE:00865	INSERTED:17-JUL-74	IDENT:02
PROSW	SIZE:00258	INSERTED:17-JUL-74	IDENT:03
PST	SIZE:01307	INSERTED:17-JUL-74	IDENT:04
READ	SIZE:00198	INSERTED:17-JUL-74	IDENT:01
REPT	SIZE:00473	INSERTED:31-JUL-74	IDENT:01
ROLHD	SIZE:00685	INSERTED:1-AUG-74	IDENT:05
RSDAT	SIZE:00374	INSERTED:17-JUL-74	IDENT:06
RSEKEC	SIZE:00974	INSERTED:1-AUG-74	IDENT:17M
R5UNP	SIZE:00117	INSERTED:17-JUL-74	IDENT:01

```

SECTR  SIZE:00551  INSERTED:1-AUG-74  IDENT:04
SETDIR  SIZE:00126  INSERTED:17-JUL-74  IDENT:02
SETDN   SIZE:00670  INSERTED:31-JUL-74  IDENT:06
SETIMM  SIZE:00292  INSERTED:17-JUL-74  IDENT:02
SETMX   SIZE:00131  INSERTED:17-JUL-74  IDENT:01
SPACE   SIZE:00449  INSERTED:22-SEP-74  IDENT:04
STMNT   SIZE:00315  INSERTED:17-JUL-74  IDENT:03
SYMBOL  SIZE:00732  INSERTED:17-JUL-74  IDENT:04
WORDR   SIZE:00141  INSERTED:17-JUL-74  IDENT:02
WRITE   SIZE:00189  INSERTED:17-JUL-74  IDENT:01

```

B.1.3 List Module Names, Full Module Information and Module Entry Points (Global Symbols)

LBR>MAC,LP:/FU/LE ↵

or

LBR>MAC,LP:/LI/FU/LE ↵

```

DIRECTORY OF FILE MAC.OLR;1
OBJECT MODULE LIBRARY CREATED BY: LBR VX02VM
LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
FILE SPACE AVAILABLE: 00015 WORDS

```

```

** MODULE:ASGMT  SIZE:00264  INSERTED:17-JUL-74  IDENT:02

```

```

  ASGMT  ASGMTF

```

```

** MODULE:ASSEM  SIZE:00749  INSERTED:1-AUG-74  IDENT:05M

```

```

  ALLOC$  ASSEM  CLSALL  EDRITS  LCRITS  MACP1  XCTPAS  XCTPRG

```

```

** MODULE:CNDTL  SIZE:00727  INSERTED:31-JUL-74  IDENT:04

```

```

  CNDBAS  CNDTOP  FNDC  IF  IFF  IFT  IFTF  IIF

```

```

** MODULE:CODHD  SIZE:00923  INSERTED:17-JUL-74  IDENT:06

```

```

  CPXSTL  INSIZE  OBJDMP  OBJINI  OBJLOC  OBJPNT  OBJSEC  PCRCNT
  PCROLL  PCRTBL  RLDDMP  RLDPNT  STCODE  TSTRLO  ZAPCPX

```

```

** MODULE:DATOR  SIZE:00414  INSERTED:17-JUL-74  IDENT:07

```

```

  BLKB  IDENT  RADIX  RAD5X

```

```

** MODULE:ENBDS  SIZE:00248  INSERTED:1-AUG-74  IDENT:06

```

```

  EDTBAS  EDTTOP  ENABL

```

** MODULE:ENDLN SIZE:00812 INSERTED:31-JUL-74 IDENT:06
 ENDLIN ERRBTS ERRCNT LINBUF LINEND LSTBUF

** MODULE:ENDPS SIZE:01060 INSERTED:17-JUL-74 IDENT:04
 ENDP1 ENDP2

** MODULE:EXPRS SIZE:01211 INSERTED:31-JUL-74 IDENT:06
 ABSERR ABSFXP ABSTRM ABSTST EXPR GLBEXP GLBTM RELEXP
 RELTRM RELTST TERM

** MODULE:FLOAT SIZE:00035 INSERTED:17-JUL-74 IDENT:02

** MODULE:GETLN SIZE:00670 INSERTED:17-JUL-74 IDENT:05
 FFCNT GETLIN LINNUM LPPCNT PAGEXT PAGNUM SEGEN
 GMARG GMARGF RMARG

** MODULE:INFIL SIZE:00941 INSERTED:9-SEP-74 IDENT:12
 CMLM2 CMLM3 CMLM4 CMLM5 CSIM2 CSIM5 FINP1 INPM1
 OPENCH OPNSRC OPSWT1 OPSWT2 OUTERM OUTM1 STKM1 \$OPSWT

** MODULE:INIFL SIZE:00493 INSERTED:31-JUL-74 IDENT:01
 SRCNAM \$INIFL

** MODULE:INOFL SIZE:00960 INSERTED:9-SEP-74 IDENT:01
 LSTNAM OBJNAM \$INOFL

** MODULE:LABEL SIZE:00400 INSERTED:1-AUG-74 IDENT:04
 LABEL LABELF

** MODULE:LISTC SIZE:00284 INSERTED:17-JUL-74 IDENT:04
 LCTBAS LCTTOP LIST PAGE

** MODULE:LSTNG SIZE:00566 INSERTED:17-JUL-74 IDENT:07
 CRLF LINPPG LSTDEV LSTREQ PAGMNE PF0 PF1 PUTKB
 PUTKRL PUTLIN PUTLP SETBYT SETPF0 SETPF1 SETWDB SETWRD

** MODULE:MACRO SIZE:02019 INSERTED:17-JUL-74 IDENT:013
 ALTSAV ASCII ASCIZ BASCND BASCOD BASCPX BASDUM BASDUM
 BASEDT BASLCD BASLH BASLSY BASMAA BASMAR BASMAC BASPST
 BASREG BASSAT BASSEC BASSRC BASSST BASSTK BASSWT BASSYM

BLKW	BYTMOD	CHRPNT	CLCFGS	CLCLOC	CLCMAX	CLCNAM	CLCSEC
CNDROL	CODROL	CPXROL	DMAROL	DSABL	DUMROL	EDTROL	ENDFLG
EOT	ERRMNE	ERR.	FRR.A	ERR.B	ERR.D	ERR.E	ERR.I
ERR.L	ERR.M	ERR.N	ERR.O	ERR.P	ERR.Q	ERR.R	ERR.T
ERR.U	ERR.Z	EVEN	FLAGS	IMPPAS	IMPPAT	IMPURE	IMPURT
IRPC	LCDROL	LIBROL	LSYROL	MAAROL	MABROL	MACP2	MACP2F
MACROL	MEXIT	MODE	MOVBYT	NLIST	ODD	OPCERR	OPCLAS
OVMACR	OVSTMT	PASS	PSTROL	REGBAS	REGROL	REGTOP	RELLVL
ROLBAS	ROLSIZ	RULTOP	RS.CND	RS.COD	RS.CPX	RS.DMA	RS.DUM
RS.EDT	RS.LCD	RS.LIB	RS.LSY	RS.MAA	RS.MAR	RS.MAC	RS.PST
RS.REG	RS.SAT	RS.SEC	RS.SRC	RS.SST	RS.STK	RS.SWT	RS.SYM
RSOABS	RSODOT	SATROL	SAVREG	SECROL	SECTOR	SETXPR	SIZCND
SIZCOD	SIZCPX	SIZDMA	SIZDUM	SIZEDT	SIZLCD	SIZLIB	SIZLSY
SIZMAA	SIZMAB	SIZMAC	SIZPST	SIZREG	SIZSAT	SIZSEC	SIZSRC
SIZSST	SIZSTK	SIZSWT	SIZSYM	SRCROL	SSTROL	STKROL	SWTROL
SYMBEG	SYMBOL	SYMROL	TOPCND	TOPCOD	TOPCPX	TOPDMA	TOPDUM
TOPEDT	TOPLCD	TOPLIB	TOPLSY	TOPMAA	TOPMAR	TOPMAC	TOPPST
TOPREG	TOPSAT	TOPSEC	TOPSRC	TOPSST	TOPSTK	TOPSWT	TOPSYM
VALUE	WORD	XCTLIN	XMIT0	XMIT1	XMIT2	XMIT3	XMIT4
XMIT5	XMIT6	XMIT7					

++ MODULE:MACRS SIZE:01540 INSERTED:9-SEP-74 IDENT:09

ENDLOA	GETBLK	MACR	MACROC	MACROF	MT.MAC	MT.MAX	PROMA
PROMCF	PROMT	SETMAC	WCIMT				

++ MODULE:MCALL SIZE:00264 INSERTED:31-JUL-74 IDENT:01

MCALL

++ MODULE:MLIBS SIZE:00807 INSERTED:31-JUL-74 IDENT:06

CPYMAC	FINSML	GETFID	INISML	SHLFDB			
--------	--------	--------	--------	--------	--	--	--

++ MODULE:MSCDR SIZE:00843 INSERTED:31-JUL-74 IDENT:08

END	ERROR	GLOBL	PRINT	SBTTL	SETHDR	TITLE	
-----	-------	-------	-------	-------	--------	-------	--

++ MODULE:NDRCT SIZE:00258 INSERTED:31-JUL-74 IDENT:02

NARG	NCHR	NTYPE					
------	------	-------	--	--	--	--	--

++ MODULE:PROCSI SIZE:00216 INSERTED:17-JUL-74 IDENT:01

DSADDR	DSMSK	FNADDR	ENMSK	LIADDR	LIMSK	MLMSK	NLADDR
NLMSK	PAMSK	PROCSI	SPMSK				

++ MODULE:PROPC SIZE:00865 INSERTED:17-JUL-74 IDENT:02

AEXP	OPCL00	OPCL01	OPCL02	OPCL03	OPCL04	OPCL05	OPCL06
OPCL07	OPCL08	OPCL09	OPCL10	PROPC			

```

** MODULE:PROSW      SIZE:00258  INSERTED:17-JUL-74  IDENT:03
    PROSW  SWTHAS  SWTTOP

** MODULE:PST        SIZE:01307  INSERTED:17-JUL-74  IDENT:04
    BSYTOP  DFLCND  DFLG8M  DFLGEV  DFLMAC  DFLSMC  PSTBAS  PSTTOP
    SSTBAS  SSTTOP  WRDSYM

** MODULE:READ      SIZE:00198  INSERTED:17-JUL-74  IDENT:01
    GETVBN  $READ

** MODULE:REPT      SIZE:00473  INSERTED:31-JUL-74  IDENT:01
    ENDMAC  IRP      MPUSH  REPT

** MODULE:ROLHD     SIZE:00685  INSERTED:1-AUG-74  IDENT:05
    APPEND  INSERT  LSRFGS  LSFLAG  LSGBAS  LSRCH   LSYBKN  MSRCH
    NEXT    OSRCH  ROLNDX  ROLUPD  SCAN    SCANW   SEARCH  SSRCH
    ZAP

** MODULE:RSDAT     SIZE:00374  INSERTED:17-JUL-74  IDENT:06
    APR,MAX  ENDLVL  ENDMEX  ENDMSK  ENDARD  CONCNT  CRADIA  EDINIT
    EDMASK  EDMRAK  EDMCSI  ED,AMA  ED,GRL  ED,LSR  ED,REG  ENDVEC
    EXMFLG  GMABLK  GMAPNT  LBLEND  LCBEGL  LCENDL  LCFLAG  LCINIT
    LCLVL   LCMASK  LCMCSI  LCSAVE  LCSAVL  LCSBAK  LC.     LC,BEX
    LC,BIN  LC,CND  LC,COM  LC,LD   LC,LOC  LC,MC   LC,MD   LC,ME
    LC,MEB  LC,SEQ  LC,SRC  LC,SYM  LC,TOC  LC,TTM  LIBNUM  MACGSB
    MACLVL  MACNAM  MACNXT  MACTXT  MACART  MSBARG  MSBBLK  MSBCNT
    MSBEND  MSBLGH  MSBMRP  MSBPRP  MSHTXT  MSRTYP  PRGIDN  PRGTTL
    SMLLVL  SRCNUM  STARS  STLBUF  TTLBRK  TTLBUF

** MODULE:RSEXEC    SIZE:00974  INSERTED:1-AUG-74  IDENT:17M
    BUFTAL  CLOSRC  CMIBUF  CMLBLK  CNTTBL  CONT    CSIBLK  DATTIM
    DEFMC   FDBTBL  FOR1    FOR2    GETFLG  GETPLI  HDRITL  IOFTBL
    IOSEOF  IO,ERR  IO,NNU  IO,OPN  IO,OUT  IO,TTY  LOAMAC  LSTFIL
    MACLDG  OBJBUF  OBJFIL  PASSSW  PURGMC  RESTRT  RLDBUF  SPSAV
    SRCCLD  SRCMRK  SRCPNT  SRCSAV  TSTSTK  VBNSAV  $LIMIT  $LSTVZ
    $SWTCH

** MODULE:R5UNP     SIZE:00117  INSERTED:17-JUL-74  IDENT:01
    R5UNP

** MODULE:SECTR     SIZE:00551  INSERTED:1-AUG-74  IDENT:04
    ASECT  CSECT  LIMIT  PSECT  SATBAS  SATTOP  SECINI

```

```

** MODULE:SETDIR  SIZE:00126  INSERTED:17-JUL-74  IDENT:02
    SETDIR

** MODULE:SETDN  SIZE:00670  INSERTED:31-JUL-74  IDENT:06
    SETDN  SETTIM

** MODULE:SETIMM SIZE:00292  INSERTED:17-JUL-74  IDENT:02
    SETDSP  SETIMM

** MODULE:SETMX  SIZE:00131  INSERTED:17-JUL-74  IDENT:01
    SETMAX

** MODULE:SPACE  SIZE:00449  INSERTED:22-SEP-74  IDENT:04
    MRKOUT  REMMAC  SHFMSB  SQZSTK

** MODULE:STMNT  SIZE:00315  INSERTED:17-JUL-74  IDENT:03
    STMNT

** MODULE:SYMBL  SIZE:00732  INSERTED:17-JUL-74  IDENT:04
    ARGENT  ARGPNT  CHSCAN  CITBL  CT.ALP  CT.COM  CT.EQL  CT.LC
    CT.NUM  CT.PC  CT.PCX  CT.SMC  CT.SP  CT.SPT  CT.TAB  CVTNUM
    DIV  DNC  DNCF  EXPFLG  GETCHR  GETNB  GETR50  GETSYM
    GSARG  GSARGF  MUL  MULR50  SETCHR  SETNR  SETR50  SETSYM
    TSTARG  TSTR50

** MODULE:WORDB  SIZE:00141  INSERTED:17-JUL-74  IDENT:02
    RYTE

** MODULE:*WRITE  SIZE:00189  INSERTED:17-JUL-74  IDENT:01
    $QCMO  *WRITE

```

B.1.4 List Module Names and Module Entry Points (Global Symbols)

```

LBR>MAC,LP:/LE ↵
    or
LBR>MAC,LP:/LI/LE ↵

```

```

DIRECTORY OF FILE MAC.OLB;1
OBJECT MODULE LIBRARY CREATED BY: LBR VX02VM
LAST INSERT OCCURRED 22-SEP-74 AT 11:51:50

```

MNT ENTRIES ALLOCATED: 64; AVAILABLE: 20
EPT ENTRIES ALLOCATED: 640; AVAILABLE: 92
FILE SPACE AVAILABLE: 00015 WORDS

** MODULE:ASGMT

ASGMT ASGMTF

** MODULE:ASSEM

ALLDCS ASSEM CLSALL EDRITS LCBITS MACP1 XCTPAS XCTPRG

** MODULE:CNDTL

CNDHAS CNDTOP ENDC IF IFF IFT IFTF IIF

** MODULE:CODHO

CPXSTL INSIZE OBJDMP OBJINI OBJLOC OBJPNT OBJSEC PCRCNT
PCROLL PCRTL RLDDMP RLDPNT STCODE TSTHLD ZAPCPX

** MODULE:DATOR

BLKB IDENT RADIX RAD50

** MODULE:ENRDS

EDTRAS EDTTOP ENABL

** MODULE:ENDLN

ENDLIN ERRBTS ERRCNT LINBUF LINEND LSTRUF

** MODULE:ENDPS

ENDP1 ENDP2

** MODULE:EXPRS

ABSERK ABSEXP ABSTRM ABSTST EXPR GLBEXP GLRTRM RELEXP
RELTRM RELTST TERM

** MODULE:FLOAT

** MODULE:GETLN

FFCNT GETLIN LINNUM LPPCNT PAGEXT PAGNUM SEQEND

GMARG GMARGF RMARG

** MODULE:INFIL

CMLM2 CMLM3 CMLM4 CMLM5 CSIM2 CSIM5 FINP1 INPM1
OPENCH OPNSRC OPSWT1 OPSWT2 OUTERM OUTM1 STKM1 \$OPSWT

** MODULE:INIFL

SRCNAM \$INIFL

** MODULE:INDFL

LSTNAM OBJNAM \$INDFL

** MODULE:LABEL

LABEL LABELF

** MODULE:LISTC

LCTBAS LCTTOP LIST PAGE

** MODULE:LSTNG

CRLF LINPPG LSTDEV LSTREQ PAGMNE PF0 PF1 PUTKR
PUTKBL PUTLIN PUTLP SETBYT SETPF0 SETPF1 SETWDB SETWRD

** MODULE:MACRO

ALTSV ASCII ASCIZ BASCND BASCOD BASCPX BASDMA BASDUM
BASEDT BASLCD RASLIH BASLSY BASMAA BASMAB BASMAC BASPST
BASREG BASSAT RASSEC BASSRC BASSST BASSTK BASSWT BASSYM
BLKW RYTMOD CHRPNL CLCFGS CLCLOC CLCMAX CLCNAM CLCSEC
CNDROL CODROL CPXROL DMAROL DSABL DUMROL EDROL ENDFLG
EOT ERRMNE ERR.A ERR.R ERR.D ERR.E ERR.I
ERR.L ERR.M ERR.N ERR.O ERR.P ERR.Q ERR.R ERR.T
ERR.U ERR.Z EVEN FLAGS IMPPAS IMPPAT IMPURE IMPURT
IRPC LCDROL LIRROL LSYROL MAAROL MABROL MACP2 MACP2F
MACROL MEXIT MODE MOVBYT NLIST ODD OPCERR OPCLAS
OVMACR OVSTMT PASS PSTROL REGHAS REGROL REGTOP RELVL
ROLBAS ROLSIZ ROLTOP RS.CND RS.COD RS.CPX RS.DMA RS.DUM
RS.EDT RS.LCD RS.LIH RS.LSY RS.MAA RS.MAB RS.MAC RS.PST
RS.REG RS.SAT RS.SEC RS.SRC RS.SST RS.STK RS.SWT RS.SYM
RSZABS RSZDOT SATROL SAVREG SECROL SECTOR SETXPR SIZCND
SIZCOD SIZCPX SIZDMA SIZDUM SIZEDT SIZLCD SIZLIH SIZLSY
SIZMAA SIZMAB SIZMAC SIZPST SIZREG SIZSAT SIZSEC SIZSRC
SIZSST SIZSTK SIZSWT SIZSYM SRCROL SSTROL STKROL SWTROL
SYMBEG SYMBOL SYMROL TOPCND TOPCOD TOPCPX TOPDMA TOPDUM
TOPEDT TOPLCD TOPLIB TOPLSY TOPMAA TOPMAB TOPMAC TOPPST
TOPREG TOPSAT TOPSEC TOPSRC TOPSST TOPSTK TOPSWT TOPSYM
VALUE WORD XCTLIN XMIT0 XMIT1 XMIT2 XMIT3 XMIT4
XMIT5 XMIT6 XMIT7


```

** MODULE:MACRS
    PROMCF  PROMT  SETMAC  WCIMT

** MODULE:MCALL
    MCALL

** MODULE:MLIBS
    CPYMAC  FINSML  GETFID  INJSML  SMLFDH

** MODULE:MSCDR
    END      ERROR  GLOBL  PRINT  SBTTL  SETHDR  TITLE

** MODULE:NDRCT
    NARG     NCHR   NTYPE

** MODULE:PROCSI
    DSADDR  DSMSK  ENADDR  ENMSK  LIADDR  LIMSK  MLMSK  NLADDR
    NLMSK   PMSK  PROCSI  SPMSK

** MODULE:PROPC
    AEXP    OPCL00  OPCL01  OPCL02  OPCL03  OPCL04  OPCL05  OPCL06
    OPCL07  OPCL08  OPCL09  OPCL10  PROPC

** MODULE:PROSW
    PROSW   SWTBAS  SWTTOP

** MODULE:PST
    RBYTOP  DFLCND  DFLGRM  DFLGEV  DFLMAC  DFLSMC  PSTHAS  PSTTOP
    SSTHAS  SSTTOP  WRDSYM

** MODULE:READ
    GETVBN  $READ

** MODULE:REPT
    ENDMAC  IRP     MPUSH  REPT

** MODULE:ROLHD
    APPEND  INSERT  LSRFGS  LSFLAG  LSGBAS  LSRCH  LSYBKN  MSRCH
    NEXT    OSRCH  ROLNDX  ROLUPD  SCAN    SCANW  SEARCH  SSRCH
    7AP

```

++ MODULE:RSDAT

ARGMAX	CNDLVL	CNDMFX	CNDMSK	CNDWRD	CONCNT	CRADIX	EDINIT
EDMASK	EDMBAK	EDMCSI	ED.AMA	ED.GBL	ED.LSB	ED.REG	ENDVEC
EXMFLG	GMAHLK	GMAPNT	LBLEND	LCBEG	LCENDL	LCFLAG	LCINIT
LCLVL	LCMASK	LCMCSI	LCSAVE	LCSAVL	LCSBAK	LC.	LC.BEX
LC.BIN	LC.CND	LC.COM	LC.LD	LC.LOC	LC.MC	LC.MD	LC.ME
LC.MER	LC.SFG	LC.SPC	LC.SYM	LC.TOC	LC.TTM	LIRNUM	MACGSB
MACLVL	MACNAM	MACNXT	MACTXT	MACWRT	MSRAPG	MSRBLK	MSBCNT
MSREND	MSBLGH	MSBMRP	MSBPRP	MSBXT	MSBTYP	PRGIDN	PRGTTL
SMLLVL	SRCNUM	STARS	STLBUF	TTLBRK	TTLBUF		

++ MODULE:RSEXC

HUFTBL	CLOSPC	CMIBUF	CMLBLK	CNTTBL	CONT	CSIBLK	DATTIM
DEFMC	FDBTBL	FDB1	FDR2	GETFLG	GETPLI	HDRITL	IOFTBL
IO\$EOF	IO.ERR	IO.NNU	IO.OPN	IO.OUT	IO.TTY	LOAMAC	LSTFIL
MACLDG	OBJRUF	ORJFIL	PASSSW	PURGMC	RESTR	RLOBUF	SPSAV
SRCCLD	SRCMRK	SRCPNT	SRCSAV	TSTSTK	VRNSAV	\$LIMIT	\$LSTVZ
\$SWTCH							

++ MODULE:RSUNP

RSUNP

++ MODULE:SECTR

ASECT	CSECT	LIMIT	PSECT	SATBAS	SATTOP	SECINI
-------	-------	-------	-------	--------	--------	--------

++ MODULE:SETDIR

SETDIR

++ MODULE:SETDN

SETDN SETTIM

++ MODULE:SETIMM

SETDSP SETIMM

++ MODULE:SETMX

SETMAX

++ MODULE:SPACE

MKKOUT	PEMMAC	SHFMSB	SQZSTK
--------	--------	--------	--------

++ MODULE:STMNT

STMNT

** MODULE:SYMBL

ARGCNT	ARGPNT	CHSCAN	CTTBL	CT.ALP	CT.COM	CT.EOL	CT.LC
CT.NUM	CT.FC	CT.PC*	CT.SMC	CT.SP	CT.SPT	CT.TAB	CVTNUM
DIV	DNC	DNCF	EXPFLG	GETCHR	GETNR	GETR50	GETSYM
GSARG	GSARGF	MUL	MULR50	SETCHR	SETNR	SETR50	SETSYM
TSTARG	TSTR50						

** MODULE:WORDB

BYTE

** MODULE:WRITE

\$QCMO \$WRITE

B.2 SAMPLE LISTING FOR LBR LIST SWITCHES (MACRO LIBRARY)

B.2.1 List Module Names

LBR>MAC,LP:↵

or

LBR>MAC,LP:LI↵

DIRECTORY OF FILE EXEMC.MLB#1
MACRO LIBRARY CREATED BY: LBR VXM3.4
LAST INSERT OCCURRED 2-JUN-75 AT 17:16:25
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 52
EPT ENTRIES ALLOCATED: 2; AVAILABLE: 2
FILE SPACE AVAILABLE: 40789 WORDS

ARODF\$
CLKDF\$
CUCDF\$
CVCDF\$
DEVDF\$
EMDF\$
F11DF\$
HDRDF\$
HWDF\$
PCBDF\$
PKTDF\$
TCBDF\$

B.2.2 List Module Names and Full Module Information

LBR>MAC,LP:/LE/FU ↵

OR

LBR>MAC,LP:/LI/LE/FU ↵

DIRECTORY OF FILE EXEMC.MLB:1
MACRO LIBRARY CREATED BY: LBR Vx03.4
LAST INSERT OCCURRED 2-JUN-75 AT 17:16:25
MNT ENTRIES ALLOCATED: 64; AVAILABLE: 52
EPT ENTRIES ALLOCATED: 0; AVAILABLE: 0
FILE SPACE AVAILABLE: 00789 WORDS

** MODULE:ABODF\$ SIZE:00201 INSERTED:2-JUN-75

** MODULE:CLKDF\$ SIZE:00240 INSERTED:2-JUN-75

** MODULE:CUCDF\$ SIZE:00376 INSERTED:2-JUN-75

** MODULE:CVCDF\$ SIZE:00601 INSERTED:2-JUN-75

** MODULE:DEVD\$ SIZE:01014 INSERTED:2-JUN-75

** MODULE:EMDF\$ SIZE:00299 INSERTED:2-JUN-75

** MODULE:F11DF\$ SIZE:00530 INSERTED:2-JUN-75

** MODULE:HORDF\$ SIZE:00321 INSERTED:2-JUN-75

** MODULE:HWDF\$ SIZE:00368 INSERTED:2-JUN-75

** MODULE:PCBDF\$ SIZE:00221 INSERTED:2-JUN-75

** MODULE:PKTDF\$ SIZE:00233 INSERTED:2-JUN-75

** MODULE:TCDF\$ SIZE:00439 INSERTED:2-JUN-75

SAMPLES OF LISTING AND EDITING

B.3 SAMPLE EDITING OPERATIONS

Four sample editing operations are included in this section to illustrate how the various EDI commands can be used. In the first example, a file is edited using a few basic EDI commands. The second example, illustrates the use of the SAVE, UNSAVE and PASTE commands. In the second example, two save files are generated, modified, and appended to the original file. Any closed file may be appended to or inserted within an open file in the same manner shown in the second example. The third example illustrates how an immediate macro command can be defined and executed in a single step. The last example illustrates how a file containing errors can be edited using the macro commands.

B.3.1 File Editing Sample

```
>EDI PRTBLD.CMD ↵
[PAGE 1]
*P * ↵
;
; COMMAND FILE TO BUILD
; PRNT SYMBIONT
; FOR RSX-11M MAXXED SYSTEM
;
;
; [1,54]PRT/MM/-CP,LP:=PRTBLD/MP
;
; OPTIONS
;
; STACK=40
; PAR=PARK:0:10000
; UNITS=4
; TASK=PRT...
; ASG=CO:2,LP:3
; PRI=60
; UC=[10,1]
;
; SPECIFY
; SPECIFY FLAG WHICH CONTROLS
; FILE DELETION AFTER PRINTING
;
; TO ENABLE DELETION USE
;
; GBLPAT=PRT;$DELET:1
;
; TO INHIBIT DELETION USE
;
; DEFAULT FROM ASSEMBLY IS
; FILE DELETION ENARbled
;
; GBLPAT=PRT:$DELET
/
; [*EOB*]
```

File PRTBLD.CMD is opened for editing. A PRINT * command is issued to print the contents of the file. The following errors are detected:

- 1 - PRNT should be PRINT.
- 2 - MAXXED should be MAPPED.

- 3 - /CP should have been used instead of /-CP.
- 4 - INPUT should be appended to the line containing the word OPTIONS.
- 5 - PARK should be PAR4K.

- 6 - UC should be UIC.
- 7 - The line containing ; SPECIFY should be deleted.

- 8 - The comment line containing the format used to inhibit deletion is missing.
- 9 - ENARbled should be ENABLED.

- 10 - A :1 should be appended to the line following the word \$DELET.
The end of buffer is reached and EDI causes the EOB message to be printed.

```

*TOF ↵
[PAGE 1]
*PL PRNT ↵
; PRNT SYMBIONT
*C/RN/RIN/ ↵
; PRINT SYMBIONT
* ↵
; FOR RSX-11M MAXXED SYSTEM
*C/XX/PP/ ↵
; FOR RSX-11M MAPPED SYSTEM
*NP 3 ↵
[1,54]PRT/MM/-CP,LP:=PRTBLD/MP
*C,/ -CP,/CP/ ↵
[1,54]PRT/MM/CP,LP:=PRTBLD/MP
*PL PAR= ↵
PAR=PARK:0:10000
*C/RK/R4K/ ↵
PAR=PAR4K:0:10000
*NP -3 ↵
; OPTIONS
*AP INPUT ↵
; OPTIONS INPUT
*PL UC ↵
UC=[10,1]
*C/UC/UIC/ ↵
UIC=[10,1]

```

- A TOF command is issued to move the line pointer to top of file and editing is started.
- 1 - A PAGE LOCATE command is issued to locate the first line in error and the line is printed automatically. A CHANGE command is issued to correct the line and the corrected line is verified automatically.
 - 2 - A carriage return is entered following the prompt to move the line pointer and print the next line in error. A CHANGE command is issued to correct the line and the corrected line is verified automatically.
 - 3 - A NEXT PRINT 3 command is issued to locate the next line in error and the line is printed. A CHANGE command is issued to correct the line and the corrected line is verified automatically.
 - 5 - A PAGE LOCATE command is issued to locate the next line in error and the line is printed automatically. A CHANGE command is issued to correct the line and the corrected line is verified automatically.
 - 4 - A line in error was bypassed by mistake; therefore, a NEXT PRINT -3 command is issued to back the line pointer up. An ADD AND PRINT command is used to correct the line
 - 6 - A PAGE LOCATE command is used to located the next line in error and the line is printed automatically. A CHANGE command is issued to correct the line and the corrected line is verified automatically.


```

* ↵
; ↵
* ↵
; SPECIFY
*DP ↵
; SPECIFY FLAG WHICH CONTROLS
*PL INH ↵
; TO INHIBIT DELETION USE
*I ↵
; ↵
; GBLPAT=PRT:$DELET:0 ↵
*PL RB ↵
; FILE DELETION ENARbled
*C/R// ↵
; FILE DELETION ENABLED
* ↵
; ↵
* ↵
GBLPAT=PRT:$DELET
*AP :1 ↵
GBLPAT=PRT:$DELET:1
*TOF ↵
[PAGE 1]

```

- 7 - The line pointer is moved down two lines via the carriage return option to locate the next line in error. A DELETE AND PRINT command is issued to delete the line containing ; SPECIFY and print the next line.
- 8 - A PAGE LOCATE command is issued to locate the point in the file where the new comment lines are to be inserted. EDI is switched to the Input mode, two lines are entered, and EDI is switched back to Edit mode by entering a carriage return as the first character in the line.
- 9 - A PAGE LOCATE command is issued to locate the next line in error. A CHANGE command is issued to correct the spelling error. The line is verified automatically.
- 10 - The line pointer is moved down two lines using two carriage returns to locate the last line in error. An ADD AND PRINT command is issued to append :1 following the word \$DELET.

The necessary corrections are complete so the line pointer is moved to the top of the file via a TOF command.

```

*P * ↵
;
; COMMAND FILE TO BUILD
; PRINT SYMBIONT
; FOR RSX-11M MAPPED SYSTEM
;
;
[1, 54] PRT/MM/CP, LP: =PRTBLD/MP
;
; OPTIONS INPUT
;
STACK=40
PAR=PAR4K:0:10000
UNITS=4
TASK=PRT...
ASG=CO:2, LP:3
PRI=60
UIC=[10,1]
;
; SPECIFY FLAG WHICH CONTROLS
; FILE DELETION AFTER PRINTING
;
; TO ENABLE DELETION USE
;
; GBLPAT=PRT:$DELET:1
;
; TO INHIBIT DELETION USE
;
; GBLPAT=PRT:$DELET:0
;
; DEFAULT FROM ASSEMBLY IS
; FILE DELETION ENABLED
;
; GBLPAT=PRT:$DELET:1
/
[ *EOB* ]
*EX ↵
[EXIT]

```

A PRINT * command is issued to print the complete file with all corrections

An Exit command is issued to close the file and terminate the editing session.

B.3.2 SAVE and UNSAVE Example

```
*LI↵
THIS IS LINE 1 PAGE 1
THIS IS LINE 2 PAGE 1
THIS IS LINE 3 PAGE 1
THIS IS LINE 4 PAGE 1
THIS IS LINE 5 PAGE 1
[*EOB*]
*T↵
*SA 5 SAV1.DAT↵
```

The file to be used in this example is printed via a LIST command.

```
*T↵
*SA 5 SAV2.DAT↵
*CL↵
EDI>SAV1.DAT↵
[PAGE 1]
*LI↵
THIS IS LINE 1 PAGE 1
THIS IS LINE 2 PAGE 1
THIS IS LINE 3 PAGE 1
THIS IS LINE 4 PAGE 1
THIS IS LINE 5 PAGE 1
[*EOB*]
```

The line pointer is returned to the top. A SAVE command is used to save the five lines in a separate file.

The line pointer is returned to the top. A second SAVE command is used to generate a second saved file. The primary input file is closed. The first save file is opened and a LIST command is used to verify the file.

```
*PA/PAGE 1/PAGE 2/↵
THIS IS LINE 1 PAGE 2
THIS IS LINE 2 PAGE 2
THIS IS LINE 3 PAGE 2
THIS IS LINE 4 PAGE 2
THIS IS LINE 5 PAGE 2
*CL↵
```

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

The first save file is closed.

```
EDI>SAV2.DAT ↵  
[PAGE 1]  
*LI ↵  
THIS IS LINE 1 PAGE 1  
THIS IS LINE 2 PAGE 1  
THIS IS LINE 3 PAGE 1  
THIS IS LINE 4 PAGE 1  
THIS IS LINE 5 PAGE 1  
[*EOB*]
```

```
*PA/PAGE 1/PAGE 3/ ↵  
THIS IS LINE 1 PAGE 3  
THIS IS LINE 2 PAGE 3  
THIS IS LINE 3 PAGE 3  
THIS IS LINE 4 PAGE 3  
THIS IS LINE 5 PAGE 3  
*CL ↵  
EDI>START.DAT ↵  
[PAGE 1]
```

The second save file is opened.

The LIST command is used to verify the contents of the file.

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

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

```
*BO ↵  
THIS IS LINE 5 PAGE 1  
*UNS SAV1.DAT ↵  
*UNS SAV2.DAT ↵  
*T ↵  
*LI ↵  
THIS IS LINE 1 PAGE 1  
THIS IS LINE 2 PAGE 1  
THIS IS LINE 3 PAGE 1  
THIS IS LINE 4 PAGE 1  
THIS IS LINE 5 PAGE 1  
THIS IS LINE 1 PAGE 2  
THIS IS LINE 2 PAGE 2  
THIS IS LINE 3 PAGE 2  
THIS IS LINE 4 PAGE 2  
THIS IS LINE 5 PAGE 2  
THIS IS LINE 1 PAGE 3  
THIS IS LINE 2 PAGE 3  
THIS IS LINE 3 PAGE 3  
THIS IS LINE 4 PAGE 3  
THIS IS LINE 5 PAGE 3  
[*EOB*]  
*EX ↵  
[EXIT]
```

The last line in the file is located.
Two UNSAVE commands are used to
append the two save files to the
original input file.
A LIST command is used to
verify the contents of the
combined file.

B.3.3 Use of Immediate Macro Command

```
*LI ↵
ABC IN LINE 1 - ABC
ABC IN LINE 2 - ABC
ABC IN LINE 3 - ABC
ABC IN LINE 4 - ABC
ABC IN LINE 5 - ABC
.
.
.
ABC IN LINE N - ABC
[*EOB*]
*4<F ABC&C/ABC/DEF/> ↵
[OVERLAYING PREVIOUSLY DEFINED MACRO]
ABC IN LINE 1 - ABC
DEF IN LINE 1 - ABC
ABC IN LINE 2 - ABC
DEF IN LINE 2 - ABC
ABC IN LINE 3 - ABC
DEF IN LINE 3 - ABC
ABC IN LINE 4 - ABC
DEF IN LINE 4 - ABC
*
```

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

The immediate macro is defined and executed to find the first four lines which start with ABC and change the first occurrence of the string ABC to DEF. The FIND command causes the line to be printed before the change. The CHANGE command causes the line to be printed after the change.

B.3.4 Use of Macro Commands

*LI ↙
THIS LITTLE FILE HAS
MANY CONNON ETTORS SO
WE CAN SHOW YOU HOW
YHE MACRO CONNANDS CAN
BE USED.
FIRST, YHE DESIRED MACRO
MUST BE DEFINED; YHE LINE
POINTER IS MOVED TO A LINE
WITH AN ETTOR; AND YHEN, YHE
MACRO EXECUTE CONNAND
IS ISSUED TO COTTECT YHE
ETTOR
[*EOB*]
*MACRO 1 C/NN/MM/ ↙
*MACRO 2 SC/TT/RR/ ↙
*MACRO 3 PA/YHE/THE/ ↙
*M3 ↙
THE MACRO CONNANDS CAN
FIRST, THE DESIRED MACRO
MUST BE DEFINED; THE LINE
WITH AN ETTOR; AND THEN, THE
IS ISSUED TO COTTECT THE
*NP2 ↙
MANY CONNON ETTORS SO
*M1 ↙
MANY COMMON ETTORS SO
*M2 ↙
MANY COMMON ERRORS SO

The LIST command is used to print the file and the file is checked for errors. The following errors are located.

1. The string NN is used in place of MM (see macro 1).
2. The string TT is used in place of RR (see macro 2).
3. The string YHE is used in place of THE (see macro 3).

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

Macro 3 is used to change all YHE strings to THE.

NP2 is used to locate a line with errors.

M1 is used to change NN to MM.

M2 is used to change TT to RR

*NP2
THE MACRO COMMANDS CAN
*M1
THE MACRO COMMANDS CAN

*M2
WITH AN ERROR; AND THEN, THE
*
MACRO EXECUTE COMMAND
*M1
MACRO EXECUTE COMMAND
*M2
IS ISSUED TO CORRECT THE
*M2

ERROR
*T
*LI
THIS LITTLE FILE HAS
MANY COMMON ERRORS SO
WE CAN SHOW YOU HOW
THE MACRO COMMANDS CAN
BE USED.
FIRST, THE DESIRED MACRO
MUST BE DEFINED; THE LINE
POINTER IS MOVED TO A LINE
WITH AN ERROR; AND THEN, THE
MACRO EXECUTE COMMAND
IS ISSUED TO CORRECT THE
ERROR.
[*EOB*]

NP2 is used to locate the next line in error.

M1 is used to change NN to MM.

M2 is used to locate the next TT string
and change it to RR.

↵ is used to locate the next line in error.

M1 is used to change NN to MM.

M2 is used to locate the next TT string and
change it to RR.

M2 is used to locate the last error in the
file and correct it.

After all lines have been corrected, the
file is printed using the LIST command.

B.4 SAMPLE DMP LISTINGS

B.4.1 Use of /LB Switch

```
DMP>TI:=SY:BIGMAC.TSK/LB↵  
STARTING BLOCK NUMBER = 0,135163 C
```

```
DMP>TI:=SY:SYSGEN.CMD/LB↵  
STARTING BLOCK NUMBER = 0,001606
```

B.4.2 "Standard" Command Line

This command will dump virtual blocks 1 and 2 in SYSGEN.CMD in ASCII mode.

```
DMP>LP:=SY:SYSGEN.CMD/AS/BL:1:2↵  
DMP>
```

```
DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0  
VIRTUAL BLOCK 0,000001 - SIZE 512, BYTES
```

```
000000  A T ; P ; S Y S G E N P  
000020  A R T I ; A ; S Y S D T E N D  
000040  T E R M I N E S ; S A N S X T E M S  
000060  E A T U E R E S H E S Y N E A S K E A C U A S T  
000100  M B L U E T H E E S A N E A S L D I T A N G M A  
000120  V E A O P U B U I S L Y R N A O H M O R S A  
000140  R E M A P P E D O A I S A O H W O R L D S A  
000160  A M A P P E D O A I S A O H W O R L D S A  
000200  < . A S N E I N W I G I T O R L D S A  
000220  C H I N N E I N W I G I T O R L D S A  
000240  T H I N N E I N W I G I T O R L D S A  
000260  T H I N N E I N W I G I T O R L D S A  
000300  ; N E C C E S S S T A R T L Y / T U I P  
000320  E S I G ; S I B N S P E M A C C F W T S D M  
000340  K S I G ; S I B N S P E M A C C F W T S D M  
000360  = I G ; S I B N S P E M A C C F W T S D M  
000400  P F F B B * / B P I A I L S N N O F R O I N  
000420  F F B B * / B P I A I L S N N O F R O I N  
000440  T S K ; * / B P I A I L S N N O F R O I N  
000460  T S K ; * / B P I A I L S N N O F R O I N  
000500  F I T N S B P I A I L S N N O F R O I N  
000520  F I T N S B P I A I L S N N O F R O I N  
000540  ; * / B P I A I L S N N O F R O I N  
000560  L E A T E F S I O N S N N O F R O I N  
000600  P N O I S K S I O N S N N O F R O I N  
000620  P N O I S K S I O N S N N O F R O I N  
000640  D I S K S I O N S N N O F R O I N  
000660  H T E F S I O N S N N O F R O I N  
000700  T L O 3 ; X I S V S E R ; I I  
000720  L L L E X I S V S E R ; I I  
000740  L L L E X I S V S E R ; I I  
000760  E S S A G E S S E R ; I I
```

DUMP OF DP0:[200,200]SYSGEN,CMD;15 - FILE ID 7157,35146,0
 VIRTUAL BLOCK 0,000002 - SIZE 512, BYTES

```

000000 T E D A 1 S P E D , R N P 0 C C , T E U N T W I
000020 S T E , , . R N P 0 C C , T E U N T W I
000040 H S U 2 C S X P S / A O S S I W S I N D O S F B G F
000060 O A ] D B I E P D C W G E I W S I N D O S F B G F
000080 U T N ] D B I E P D C W G E I W S I N D O S F B G F
000100 L O S ; L P T P / . * / P E M O I S P D E O N H 1 N I D E
000120 D T T E * / . * / P E M O I S P D E O N H 1 N I D E
000140 H T P / . * / P E M O I S P D E O N H 1 N I D E
000160 I E X / P E M O I S P D E O N H 1 N I D E
000180 N X / P E M O I S P D E O N H 1 N I D E
000200 D F I U K D B C S ; J = X P / * 1 B R ; T A E 1 6 Y P S ;
000220 I S I R S ; J = X P / * 1 B R ; T A E 1 6 Y P S ;
000240 C L T C S ; J = X P / * 1 B R ; T A E 1 6 Y P S ;
000260 A E . = X P / * 1 B R ; T A E 1 6 Y P S ;
000280 T F [ A I D / 1 M E 1 A M E X T S O R U C E E I
  
```

B.4.3 Dump Only the Header from SYSGEN.COMD

DMP>LP:=SY:SYSGEN.COMD/HD/BL:Ø:Ø ↵

DMP>

DUMP OF DPØ:[200,200]SYSGEN.COMD;15 - FILE ID 7157,35146,Ø
FILE HEADER

SYSGEN.COMD;15 (7157,35146) 7, /10. 11-OCT-74 11:48
[200,200] [RWED,RWED,RWED,R]

000000	027027	007157	035146	000401	100200	160000	000000	001002
000020	000077	000000	000012	000000	000010	000000	000000	000000
000040	000000	000000	000000	000000	000000	000000	000000	075273
000060	026226	000000	012314	000015	000001	030461	041517	033524
000100	030464	032461	030462	030462	047461	052103	032067	030461
000120	034064	032461	000000	000000	000000	000000	000000	000000
000140	000000	001401	146006	002000	001606	000000	044156	001400
000160	044306	000000	000000	000000	000000	000000	000000	000000
000200	000000	000000	000000	000000	000000	000000	000000	000000
000220	000000	000000	000000	000000	000000	000000	000000	000000
000240	000000	000000	000000	000000	000000	000000	000000	000000
000260	000000	000000	000000	000000	000000	000000	000000	000000
000300	000000	000000	000000	000000	000000	000000	000000	000000
000320	000000	000000	000000	000000	000000	000000	000000	000000
000340	000000	000000	000000	000000	000000	000000	000000	000000
000360	000000	000000	000000	000000	000000	000000	000000	000000
000400	000000	000000	000000	000000	000000	000000	000000	000000
000420	000000	000000	000000	000000	000000	000000	000000	000000
000440	000000	000000	000000	000000	000000	000000	000000	000000
000460	000000	000000	000000	000000	000000	000000	000000	000000
000500	000000	000000	000000	000000	000000	000000	000000	000000
000520	000000	000000	000000	000000	000000	000000	000000	000000
000540	000000	000000	000000	000000	000000	000000	000000	000000
000560	000000	000000	000000	000000	000000	000000	000000	000000
000600	000000	000000	000000	000000	000000	000000	000000	000000
000620	000000	000000	000000	000000	000000	000000	000000	000000
000640	000000	000000	000000	000000	000000	000000	000000	000000
000660	000000	000000	000000	000000	000000	000000	000000	000000
000700	000000	000000	000000	000000	000000	000000	000000	000000
000720	000000	000000	000000	000000	000000	000000	000000	000000
000740	000000	000000	000000	000000	000000	000000	000000	000000
000760	000000	000000	000000	000000	000000	000000	000000	166212

B.4.4 Use of /BA Switch

The first command sets the base block address to 2, the next command causes virtual blocks 3 and 4 to be dumped.

DMP>/BA:Ø:2 ↵

DMP>LP:=SYSGEN.COMD/BY/BL:1:2 ↵

DMP>

DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0
 VIRTUAL BLOCK 0,000003 - SIZE 512, BYTES

```

000000 116 107 040 117 116 040 101 040 115 101 103 110 111 116 105 040
000020 127 111 124 110 040 115 117 122 105 040 124 110 101 116 040 061
000040 066 113 040 127 117 122 104 123 072 000 073 011 131 117 125 040
000060 115 101 131 040 107 105 124 040 101 123 123 105 115 102 114 131
000100 040 114 111 123 124 111 116 107 123 040 117 116 114 131 040 111
000120 106 040 131 117 125 040 104 111 122 105 103 124 040 124 110 105
000140 115 040 124 117 073 000 073 011 124 110 105 040 114 111 116 105
000160 040 120 122 111 116 124 105 122 040 050 114 120 060 072 051 040
000200 117 122 040 101 116 117 124 110 105 122 040 104 111 123 113 056
000220 040 040 124 110 105 122 105 040 111 123 040 116 117 124 040 040
000240 040 103 062 000 073 011 105 116 117 125 107 110 040 123 120 101
000260 103 105 040 117 116 040 124 110 105 040 123 117 125 122 103 105
000300 040 104 111 123 113 040 124 117 040 113 105 105 120 040 101 114
000320 114 040 124 110 105 040 031 000 073 011 101 123 123 105 115 102
000340 114 131 040 114 111 123 124 111 116 107 040 106 111 114 105 123
000360 056 073 001 000 073 105 010 000 122 125 116 040 044 123 107 116
000400 001 000 073 101 063 000 073 040 101 124 040 124 110 111 123 040
000420 120 117 111 116 124 040 127 105 040 122 105 116 101 115 105 040
000440 124 110 105 040 101 123 123 105 115 102 114 131 040 103 117 115
000460 115 101 116 104 040 106 111 114 105 040 046 000 073 040 101 116
000500 104 040 124 110 105 040 123 131 123 124 105 115 040 102 125 111
000520 114 104 040 103 117 115 115 101 116 104 040 106 111 114 105 040
000540 124 117 020 000 056 111 106 106 040 101 040 073 011 133 061 061
000560 054 062 060 135 020 000 056 111 106 124 040 101 040 073 011 133
000600 061 061 054 062 064 135 001 000 073 125 057 000 056 111 106 106
000620 040 101 040 120 111 120 040 133 061 061 054 062 060 135 057 122
000640 105 075 122 123 130 101 123 115 056 103 115 104 073 052 054 122
000660 123 130 102 114 104 056 103 115 104 073 052 011 057 000 056 111
000700 106 124 040 101 040 120 111 120 040 133 061 061 054 062 064 135
000720 057 122 105 075 122 123 130 101 123 115 056 103 115 104 073 052
000740 054 122 123 130 102 114 104 056 103 115 104 073 052 000 077 000
000760 056 101 123 113 040 132 040 104 111 104 040 131 117 125 040 101

```

DUMP OF DP0:[200,200]SYSGEN.CMD;15 - FILE ID 7157,35146,0
VIRTUAL BLOCK 0,000004 - SIZE 512, BYTES

000000	116	123	127	105	122	040	124	110	105	040	123	131	123	107	105	116
000020	040	121	125	105	123	124	111	117	116	123	040	124	117	040	131	117
000040	125	122	040	123	101	124	111	123	106	101	103	124	111	117	116	040
000060	020	000	056	111	106	124	040	132	040	056	107	117	124	117	040	061
000100	060	060	001	000	073	111	074	000	073	040	127	105	040	127	111	114
000120	114	040	105	130	111	124	040	116	117	127	040	123	117	040	124	110
000140	101	124	040	131	117	125	040	115	101	131	040	122	105	123	124	101
000160	122	124	040	124	110	111	123	040	103	117	115	115	101	116	104	040
000200	106	111	114	105	067	000	073	040	106	122	117	115	040	124	110	105
000220	040	102	105	107	111	116	116	111	116	107	056	040	124	110	111	123
000240	040	127	111	114	114	040	101	114	114	117	127	040	101	040	103	114
000260	105	101	116	040	125	120	040	117	106	040	124	110	105	122	070	000
000300	073	040	104	111	123	113	040	101	116	104	040	101	040	116	105	127
000320	040	122	125	116	040	117	106	040	123	107	116	040	124	117	040	123
000340	105	114	105	103	124	040	124	110	105	040	120	122	117	120	105	122
000360	040	117	120	124	111	117	116	123	022	000	073	040	106	117	122	040
000400	131	117	125	122	040	123	131	123	124	105	115	056	001	000	073	040
000420	012	000	056	107	117	124	117	040	061	060	060	060	006	000	056	061
000440	060	060	072	073	037	000	073	040	116	117	127	040	127	105	040	101
000460	123	123	105	115	102	114	105	040	124	110	105	040	105	130	105	103
000500	125	124	111	126	105	040	001	000	073	124	027	000	056	111	106	106
000520	040	101	040	123	105	124	040	057	125	111	103	075	133	061	061	054
000540	062	060	135	000	027	000	056	111	106	124	040	101	040	123	105	124
000560	040	057	125	111	103	075	133	061	061	054	062	064	135	073	013	000
000600	115	101	103	040	100	122	123	130	101	123	115	000	001	000	073	106
000620	062	000	073	040	116	117	127	040	127	105	040	102	125	111	114	104
000640	040	124	110	105	040	103	117	116	103	101	124	105	116	101	124	105
000660	104	040	117	102	112	105	103	124	040	115	117	104	125	114	105	040
000700	106	111	114	105	024	000	073	040	106	117	122	040	124	110	105	040
000720	105	130	105	103	125	124	111	126	105	056	001	000	073	104	024	000
000740	120	111	120	040	122	123	130	061	061	115	056	117	102	123	075	052
000760	056	117	102	112	026	000	120	111	120	040	056	117	102	112	057	122

APPENDIX C

RSX-11M PRINT SPOOLER TASK

The RSX-11M Print Spooler task (PRT...) provides a means of eliminating contention for the system line printer. Rather than waiting for the line printer to become available, a task directs the output intended for the line printer to a disk file. The task issues a Send Data directive to the print spooler, placing a data block, which identifies the file to be spooled, in the print spooler queue. A Request directive then is issued by the task to activate the print spooler, in case it is not already active. All files identified in the print spooler queue are printed in a first in - first out (FIFO) order.

C.1 RECEIVE QUEUE OPERATION

The standard method of placing a user file in the print spooler receive queue (and requesting its execution) is via the PRINT\$ macro call, which is described in the RSX-11 I/O Operations Reference Manual. Files are spooled in this same manner by the RSX-11M utilities which support the spool (/SP) option. Each entry in the print spooler receive queue consists of a 13-word data block containing the file-related information illustrated in Figure C-1.

C.2 TEXT REQUIREMENTS

The print spooler task prints ASCII text with a maximum line length of 132 bytes. It will properly handle files with all modes of FCS carriage control (i.e., standard, embedded and FORTRAN).

RSX-11M PRINT SPOOLER TASK

WORD

1	Filename
2	in
3	RADIX-50
4	File type in RADIX-50
5	File version (binary)
6	Device name in ASCII
7	Unit number (binary)
8	File ID
9	
10	Directory ID
11	
12	
13	

Figure C-1
PRT Send Data Buffer Format

C.3 TASK BUILD INFORMATION

The print spooler task must be built during an RSX-11M system generation because the task image file (PRT.TSK) is not distributed on the standard release kits. Normally, the print spooler is built to delete all files which have been spooled, but the print spooler build file can be edited during system generation to disable the automatic delete feature. When the print spooler is built without automatic delete, spooled files are retained after printing. If the system has a deleting print spooler, all spooled files are deleted after printing. Therefore, the user should know whether or not the system currently running has a deleting print spooler before spooling files.

The print spooler employs double buffering to increase throughput to

RSX-11M PRINT SPOOLER TASK

the line printer. However, when the print spooler is checkpointable (the default from the build file), the Executive limits the print spooler to one outstanding I/O operation, thus effectively reducing it to single buffering. To effectively use the double buffering capability of the print spooler, it must either be built as noncheckpointable during system generation, or installed in the system as noncheckpointable. However, the print spooler should not be made noncheckpointable if this will adversely effect the execution of other tasks in the same partition.

See the RSX-11M System Generation Manual for detailed system generation information.

C.4 PRT ERROR MESSAGES

All error messages issued by PRT are sent to the console terminal via pseudo-device CO:. The error messages have the following format:

PRT -- text

In all but the receive failure error, the messages supply information that identifies the sender task and the file in question. All PRT errors are fatal; upon error detection, printing of the input file is terminated, and a clean up/restart procedure is entered.

In the case of the receive failure error, the sender and file information are unavailable. Furthermore, PRT does not attempt to dequeue additional spool requests because of the nature of this error condition. Instead, PRT exits causing its receive queue to be purged by the system.

RECEIVE FAILURE, d. -- TASK EXITING

Description

The Receive Data or Exit directive failed while attempting to obtain the next file specifier from the queue. The system error code (d.) is printed to identify the error.

NO DEVICE NAME - SENDER: task FILE: filename.typ;ver

Description

The dequeued print request did not contain a device name.

NO FILE ID - SENDER: task FILE: filename.typ;ver

Description

The dequeued print request did not contain a file ID.

RSX-11M PRINT SPOOLER TASK

OPEN FAILURE INPUT FILE - SENDER: task FILE: filename.typ;ver, d.

Description

The specified file could not be opened. One of the following conditions may exist:

1. The file is protected against access for read privileges.
2. A problem exists on the physical device (e.g., device cycled down).
3. The volume is not mounted.
4. The specified file directory does not exist.
5. The named file does not exist in the specified directory.
6. The file is already deleted.

The system error code (d.) is printed to identify the failure.

ATTACH FAILURE - SENDER: task FILE: filename.typ;ver, d.

or

DETACH FAILURE - SENDER: task FILE: filename.typ;ver, d.

Description

The line printer could not be attached/detached (i.e., system does not contain a line printer). The system error code (d.) is printed to identify the error.

PRINT ERROR - SENDER: task FILE: filename.typ;ver, d.

Description

A Queue I/O request to the line printer has failed. The system error code (d.) is printed to identify the error.

I/O ERROR INPUT FILE - SENDER: task FILE: filename.typ;ver, d.

Description

An error was detected while reading the input file. One of the following conditions may exist:

1. A problem exists on the physical device (e.g., device cycled down).
2. Length of the text line is greater than 132 bytes.
3. File is corrupted or the format is incorrect.

The system error code (d.) is printed to identify the error.

INDEX

- Aborting VFY, restrictions when, 8-1
- Access modes, text, 5-5
- Access warning messages, file, 5-9, 5-58
- ADD AND PRINT command, 5-15
- ADD command, 5-15
- ALT Mode key, 5-20
- APPEND command, 2-7
- ASCII mode switch, 4-3
- /AS switch, 4-3
- Asterisk convention, wild cards, 2-4

- Base block address switch, specify, 4-3
- Basic EDI commands, 5-13
- Basic EDI operation and commands, 5-10
- /BA switch, 4-3
- BEGIN command, 5-36
- Block address switch, specify base, 4-3
- Block command, 5-25
- Block mode, 5-5
- Block mode, line-by-line vs., 5-6
- Block switch, logical, 4-4
- Blocks, deletion of multiply allocated, 8-8
- Blocks, elimination of free, 8-8
- Blocks, recovering lost, 8-9
- Blocks switch, specify first and last, 4-3
- /BL switch, 4-3
- BOTTOM command, 5-36
- Buffer capacity exceeded, 5-53
- /BY switch, 4-4
- Byte mode switch, 4-4

- Carriage return, use of, 5-7
- Cassette directory listing, 3-18
- Cassette file formats, 3-5
- Cassette support, FLX, 3-5
- Cassette support, multi-volume, 3-9
- CHANGE command, 5-15
- Changing control mode, 5-4
- Character erase (RUBOUT), 5-7
- Check, validity, 8-5
- CLOSE AND DELETE command, 5-51
- CLOSE command, 5-51
- Close operation commands, EDI, 5-50
- CLOSES command, 5-51
- /CO, COMPRESS switch, 7-7
- Combining library functions, 7-26

- Command,
 - ADD, 5-15
 - ADD AND PRINT, 5-15
 - APPEND, 2-7
 - BEGIN, 5-36
 - BLOCK, 5-25
 - BOTTOM, 5-36
 - CHANGE, 5-15
 - CLOSE, 5-51
 - CLOSE AND DELETE, 5-51
 - CLOSES, 5-51
 - CONCATENATION CHARACTER, 5-25
 - COPY, 2-8
 - CTRL/Z, 1-2, 5-16
 - DEFAULT, 2-12
 - DELETE, 2-13, 5-16
 - DELETE AND PRINT, 5-17
 - END, 5-36
 - ENTER, 2-15
 - ERASE, 5-42
 - EXIT, 5-18
 - EXIT AND DELETE, 5-52
 - FILE, 5-32
 - FIND, 5-37
 - FORM FEED, 5-43
 - FREE, 2-16
 - IDENTIFY, 2-16
 - INSERT, 5-18
 - KILL, 5-52
 - LINE CHANGE, 5-43
 - LIST, 2-17
 - LIST ON PSEUDO-DEVICE, 5-44
 - LIST ON TERMINAL, 5-43
 - LOCATE, 5-19
 - MACRO, 5-44
 - MACRO CALL, 5-45
 - MACRO EXECUTE, 5-46
 - MACRO IMMEDIATE, 5-47
 - MERGE, 2-8
 - NEXT, 5-19
 - NEXT PRINT, 5-20
 - OLDPAGE, 5-37
 - OPENS, 5-26
 - OUTPUT, 5-26
 - OVERLAY, 5-47
 - PAGE, 5-38
 - PAGE FIND, 5-38
 - PAGE LOCATE, 5-39
 - PASTE, 5-48
 - PRINT, 5-20
 - PROTECT, 2-21
 - PURGE, 2-24
 - READ, 5-32
 - REMOVE, 2-25
 - RENAME, 2-26
 - RENEW, 5-22
 - RETYPE, 5-22
 - SAVE, 5-48
 - SEARCH AND CHANGE, 5-39
 - SELECT PRIMARY, 5-27

Command (cont.),
 SELECT SECONDARY, 5-28
 SIZE, 5-29
 SPOOL, 2-28
 TAB, 5-29
 TOP, 5-40
 TOP OF FILE, 5-22
 TYPE, 5-49
 UNLOCK, 2-29
 UNSAVE, 5-50
 UPDATE, 2-30
 UPPER CASE, 5-30
 VERIFY, 5-31
 WRITE, 5-33
 Command, concatenation character, 5-25
 Command conventions, EDI, 5-7
 Command format conventions, 1-6
 Command level informational and error messages, 5-9, 5-52
 Command string,
 DMP, 4-2
 FLX, 3-2
 LBR, 7-5
 PIP, 2-1
 VFY, 8-2
 Command strings, utility, 1-4
 Command switches and subswitches, PIP, 2-3
 Command switches, VFY, 8-4
 Commands,
 EDI close operation, 5-50
 input/output, 5-31
 line pointer (locative), 5-34
 PIP, 2-6
 setup, 5-23
 text modification and manipulation, 5-40
 Compress switch, (/CO), 7-7
 Concatenation character command, 5-25
 Constants, search string, 5-8
 Constraints, LBR, 7-27
 Control characters, SLP edit, 6-6
 Control mode, changing, 5-4
 Control modes, EDI, 5-4
 Control switches, file, 3-15
 Control switches, SLP output, 6-5
 Convention, wild cards, asterisk, 2-4
 Conventions,
 command format, 1-6
 EDI command, 5-8
 system-wide, 1-8
 terminal, 5-7
 COPY command, 2-8
 Creating a file, 5-10
 Creating an indirect file, 6-8
 /CR, CREATE switch, 7-9
 Create switch (/CR), 7-9
 CTRL/Z command, 5-16
 /DE, DELETE switch, 7-11
 DEFAULT command, 2-12
 Defaults in file specifiers,
 EDI, 5-3
 LBR, 7-5
 PIP, 2-2
 SLP, 6-3
 VFY, 8-3
 DEFAULT switch, 7-12
 DELETE AND PRINT command, 5-17
 DELETE command, 2-13, 5-16
 Delete, files marked for, 8-7
 Delete global switch (/DG), 7-13
 DELETE switch (/DE), 7-11, 8-9
 Deleting a file, 8-8
 Deleting DOS files, 3-3
 Deleting RT files, 3-4
 Deletion of multiply allocated blocks, 8-8
 /DF, DEFAULT switch, 7-12
 /DG, DELETE GLOBAL switch, 7-13
 Directory listings,
 DOS, 3-3
 RT, 3-3
 Directory manipulation,
 DOS volume, 3-3
 RT volume, 3-3
 DMP,
 file dump utility, 4-1
 initiating, 4-2
 DMP command strings, 4-2
 DMP error messages, 4-5
 DMP switches, 4-2
 DOS directory listings, 3-3, 3-18
 DOS files, deleting, 3-3
 DOS-11 volumes, initializing, 3-3
 DOS volume directory manipulation, 3-3

 EDI,
 command conventions, 5-7
 control modes, 5-4
 default file specifiers, 5-3
 error messages, 5-52
 error reporting, 5-9
 initiating, 5-2
 line text editor, 5-1
 modes, 5-4
 operation and commands, basic, 5-10
 preparing to run, 5-1
 text access, 5-5
 using, 5-1
 EDI commands,
 basic, 5-13
 close operations, 5-50
 extended, 5-23
 input/output, 5-31
 line pointer control (locative), 5-34

EDI commands (cont.),
 setup, 5-23
 text modification and manipulation, 5-40
 Edit commands, SLP, 6-6
 Edit control characters, SLP, 6-6
 Edit mode, entering text in, 5-11
 Edit modes, EDI, 5-4
 Editing a file, 5-11
 Editing examples, SLP, 6-9
 Editing session, sample, 5-23
 Elimination of free blocks, 8-8
 END command, 5-36
 ENTER command, 2-15
 Entering text in edit mode, 5-11
 Entering text in input mode, 5-11
 Entering text into a file, 5-11
 Entry point table, 7-2
 ERASE command, 5-42
 Error codes,
 PIP, 2-41
 VFY, 8-16
 Error messages,
 DMP, 4-5
 EDI, 5-52
 fatal, 5-10, 5-62
 FLX, 3-20
 LBR, 7-27
 PIP, 2-31
 SLP, 6-12
 VFY, 8-14
 Error messages, command level
 informational and, 5-9, 5-52
 Error messages requiring EDI
 restart, 5-9, 5-59
 Error reporting,
 EDI, 5-9
 file, 8-6
 Exceeded buffer capacity, 5-53
 EXIT AND DELETE command, 5-52
 EXIT command, 5-18
 Extended EDI commands, 5-23

Fatal error messages, 5-10, 5-62
 File access warning messages, 5-9,
 5-58
 File, creating a, 5-10
 File, deleting a, 8-8
 File, entering text into a, 5-11
 File, editing a, 5-11
 File, restoring a, 8-7
 File access warning messages, 5-9,
 5-58
 FILE command, 5-32
 File control switches, 3-15
 File dump utility (DMP), 4-1
 File error reporting, 8-6
 File formats, cassette, 3-5
 File header switch, 4-4
 File identification option, 2-5

File name, 1-5
 File option switches, LBR, 7-7
 Filespec, 5-3
 File specifiers,
 defaults in LBR, 7-6
 defaults in PIP, 2-2
 defaults in SLP, 6-3
 defaults in VFY, 8-3
 list of, 2-2
 File transfer program (FLX), 3-1
 File transfers, 3-2
 File structure verification
 utility (VFY), 8-1
 Files,
 deleting, DOS, 3-3
 deleting, RT, 3-4
 format of library, 7-1
 indirect, 1-7, 6-8
 library, 7-28
 marked-for-delete, 8-7
 output, 5-6
 SLP output, 6-4
 text, 5-6
 FIND command, 5-37
 FLX,
 cassette support, 3-5
 command string, 3-2
 error messages, 3-20
 file transfer program, 3-1
 initiating, 3-2
 input files, 3-10
 output files, 3-9
 paper tape support, 3-10
 switches, 3-11
 Format conventions,
 command, 1-6
 Format of library files, 7-1
 Format mode switches, 3-11
 Formats, cassette file, 3-5
 FORM FEED command, 5-43
 Free blocks, elimination of, 8-8
 FREE command, 2-16
 FREE switch (/FR), 8-12
 /FR, FREE switch, 8-12
 /FU,/LE,/LI, list switches, 7-15

/HD switch, 4-4
 Header,
 library, 7-2
 module, 7-4
 Header switch, file, 4-4

/ID switch, 4-4
 Identification switch, 4-4
 IDENTIFY command, 2-16
 Indirect file, creating an, 6-8
 Indirect files, 1-7, 6-8
 Indirect files, using, 6-9

Inflespc, 1-4
 Informational and error messages,
 command level, 5-9, 5-52
 /IN, INSERT switch, 7-14
 Initializing,
 DOS-11 volumes, 3-3
 RT-11 volumes, 3-4
 Initiation of installed utilities,
 1-2
 Initiation of uninstalled
 utilities, 1-3
 Initiating,
 DMP, 4-2
 EDI, 5-2
 FLX, 3-2
 LBR, 7-5
 PIP, 2-1
 RSX-11M utilities, 1-2
 SLP, 6-2
 VFY, 8-2
 Input and secondary files, 5-7
 Input file, FLX, 3-10
 Input mode, entering text in, 5-11
 Input/Output commands, 5-31
 INSERT command, 5-18
 Insert switch (/IN), 7-14
 Installed utilities, initiation
 of, 1-2

 KILL command, 5-52

 /LB switch, 4-4
 LBR command string, 7-5
 LBR constraints, 7-27
 LBR error messages, 7-27
 LBR file option switches, 7-7
 LBR, initiating, 7-5
 LBR, librarian utility program,
 7-1
 /LE,/LI,/FU, list switches, 7-15
 Librarian utility program (LBR),
 7-1
 Library files, 7-28
 Library files, format of, 7-1
 Library functions, combining, 7-26
 Library header, 7-2
 /LI,/LE,/FU, list switches, 7-15
 /LI, LIST switch, 8-12
 Line-by-line mode, 5-5
 Line-by-line vs. block mode, 5-6
 LINE CHANGE command, 5-43
 Line delete (CTRL/U), 5-7
 Line pointer control (locative)
 commands, 5-34
 Line text editor (EDI), 5-1
 LIST command, 2-17
 LIST ON PSEUDO-DEVICE command,
 5-44
 LIST ON TERMINAL command, 5-43
 LIST switch (/LI), 8-12
 List switches (/LI,/LE,/FU), 7-15
 LOCATE command, 5-19
 Locative commands, line pointer,
 5-34
 Logical block switch, 4-4
 /LO, LOST switch, 8-12
 Lost blocks, recovering, 8-9
 LOST switch (/LO), 8-12

 MACRO CALL command, 5-45
 MACRO command, 5-44
 MACRO EXECUTE command, 5-46
 MACRO IMMEDIATE command, 5-47
 /MD switch, 4-4
 Memory dump switch, 4-4
 MERGE command, 2-8
 Mode,
 block, 5-5
 changing control, 5-4
 line-by-line, 5-5
 Mode switch, ASCII, 4-3
 Mode switch, byte, 4-4
 Mode switches, format, 3-11
 Mode switches, transfer, 3-12
 Modes, EDI control, 5-4
 Module header, 7-4
 Module name table, 7-4
 Multiply-allocated blocks,
 deletion of, 8-8
 Multi-volume cassette support, 3-9

 NEXT command, 5-19
 NEXT PRINT command, 5-20
 Normal operation, VFY, 8-2

 OLDPAGE command, 5-37
 OPENS command, 5-26
 Outflespc, 1-4
 OUTPUT command, 5-26
 Output control switches, SLP, 6-4
 Output files,
 EDI, 5-7
 FLX, 3-10
 SLP, 6-4
 OVERLAY command, 5-47

 PAGE command, 5-38
 PAGE FIND command, 5-38
 PAGE LOCATE, 5-39
 Paper tape support, FLX, 3-10
 PASTE command, 5-48

Peripheral Interchange Program (PIP), 2-1
 PIP commands, 2-6
 PIP command string, 2-1
 PIP command switches and sub-switches, 2-3
 PIP error codes, 2-41
 PIP error messages, 2-31
 PIP, Peripheral Interchange Program, 2-1
 Preparing to run EDI, 5-1
 Preparing to run SLP, 6-1
 PRINT command, 5-20
 PROTECT command, 2-21
 PURGE command, 2-24

 /RC, READ CHECK switch, 8-13
 READ CHECK switch (/RC), 8-13
 READ command, 5-32
 REBUILD switch (/RE), 8-11
 Recovering lost blocks, 8-9
 REMOVE command, 2-25
 RENAME command, 2-26
 RENEW command, 5-22
 REPLACE switch (/RP), 7-17
 global format, 7-17
 local format, 7-18
 /RE, REBUILD switch, 8-11
 Restart, error messages requiring EDI, 5-9, 5-59
 Restoring a file, 8-7
 Restrictions,
 while running VFY, 8-1
 when aborting VFY, 8-1
 RETYPE command, 5-22
 /RP, replace switch, 7-17
 RSX-11M utilities, initiating, 1-2
 RT directory listing, 3-19
 RT directory listings, 3-3
 RT files, deleting, 3-4
 RT volume directory manipulation, 3-3
 RT-11 volumes, initializing, 3-4
 Run EDI, preparing to, 5-1
 Run SLP, preparing to, 6-1
 Running VFY, restrictions while, 8-1

 Sample editing session, 5-23
 SAVE command, 5-48
 SEARCH AND CHANGE command, 5-39
 Search string constants, 5-8
 Secondary files, input and, 5-7
 SELECT PRIMARY command, 5-27
 SELECT SECONDARY command, 5-28
 Selective search switch (/SS), 7-23
 Setup commands, 5-23

 SIZE command, 5-29
 SLP,
 capabilities, 6-1
 edit commands, 6-6
 edit control characters, 6-7
 editing examples, 6-10
 environment, 6-1
 error messages, 6-12
 initialization, examples of, 6-3
 initiating, 6-2
 output control switches, 6-4
 output files, 6-4
 restrictions, 6-2
 source language input program, 6-1
 startup, 6-2
 Source language input program (SLP), 6-1
 Specify base block address switch, 4-3
 Specify first and last blocks switch, 4-3
 SPOOL command, 2-28
 Spool switch (/SP), 7-23
 /SP, spool switch, 7-23
 Squeeze switch (/SZ), 7-24
 Startup, SLP, 6-2
 String,
 DMP command, 4-2
 FLX command, 3-2
 UTILITY command, 1-4
 VFY command, 8-2
 Subswitches, PIP command switches and, 2-3
 Switch,
 /AS, 4-3
 ASCII mode, 4-3
 /BA, 4-3
 /BL, 4-3
 /BY, 4-4
 byte mode, 4-4
 /CO, COMPRESS, 7-7
 /CR, CREATE, 7-9
 /DE, DELETE, 7-11, 8-9
 /DF, DEFAULT, 7-12
 /DG, DELETE GLOBAL, 7-13
 file header, 4-4
 /FR, FREE, 8-12
 /HD, 4-4
 /ID, 4-4
 Identification, 4-4
 /IN, INSERT, 7-14
 /LB, 4-4
 /LI, LIST, 8-12
 logical block, 4-4
 /LO, LOST, 8-12
 /MD, 4-4
 memory dump, 4-4
 /RC, READ CHECK, 8-13
 /RE, REBUILD, 8-11
 /RP, REPLACE, 7-17
 specify base block address, 4-3

- Switch (cont.),
 - specify first and last blocks, 4-3
 - /SP, SPOOL, 7-23
 - /SS, SELECTIVE SEARCH, 7-23
 - /SZ, SQUEEZE, 7-24
 - /UP, UPDATE, 8-10
- Switches (/LI,/LE,/FU), list, 7-15
- Switches and subswitches, PIP command, 2-3
- Switches,
 - DMP, 4-2
 - file control, 3-15
 - FLX, 3-11
 - format mode, 3-11
 - LBR file option, 7-7
 - transfer mode, 3-13
 - VFY command, 8-4
- System-wide conventions, 1-8

- TAB command, 5-29
- Terminal conventions, 5-7
- Text access modes, 5-5
- Text files, 5-6
- Text modification and manipulation commands, 5-40
- TOP commands, 5-40
- TOP OF FILE commands, 5-22
- Transfer mode switches, 3-13
- Transfers, file, 3-2
- Truncated output file, utilizing a, 5-62
- TYPE command, 5-49

- Uninstalled utilities, initiation of, 1-3
- UNLOCK command, 2-29
- UNSAVE command, 5-50
- UPDATE command, 2-30
- UPDATE switch (/UP), 8-10
- UPPER CASE command, 5-30
- Use of carriage return, 5-7
- Use of *, 5-8
- User identification code, 1-5
- Using EDI, 5-1
- Using indirect files, 6-9
- Utilities, initiating RSX-11M, 1-2
- Utility command string, 1-4
- Utilizing a truncated output file, 5-62

- Validity check, 8-5
- VERIFY command, 5-31
- VFY,
 - command string, 8-2
 - command switches, 8-4
 - error codes, 8-16
 - error messages, 8-14
 - file structure verification utility, 8-1
 - initiating, 8-2
 - normal operation, 8-2
- Volume directory manipulation, RT, 3-3
- Volumes, initializing DOS-11, 3-3
- Volumes, initializing RT, 3-4

- Warning messages, file access, 5-9, 5-58
- Wild cards, asterisk convention, 2-4
- WRITE command, 5-33

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- Assembly language programmer
- Higher-level language programmer
- Occasional programmer (experienced)
- User with little programming experience
- Student programmer
- Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country

If you require a written reply, please check here.

digital

digital equipment corporation