PRO/Tool Kit Command Language and Utilities Manual

Order No. AA-X912B-TH

November 1985

This manual describes the Command Language and utilities provided with the PRO/Tool Kit. It is a detailed reference manual for programmers.

REQUIRED SOFTWARE: Host Tool Kit V3.0, or PRO/Tool Kit V3.0

OPERATING SYSTEM: P/OS V3.0

DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts 01754-2571

First Printing, September 1983
Revised November 1985

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

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

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

The specifications and drawings, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.

Copyright © 1985 by Digital Equipment Corporation
All Rights Reserved

The following are trademarks of Digital Equipment Corporation:

CTI BUS	MASSBUS	Rainbow
DEC	PDP	RSTS
DECmate	P/OS	RSX
DECsystem-10	PRO/BASIC	Tool Kit
DECSYSTEM-20	PRO/Communications	UNIBUS
DECUS	Professional	VAX
DECwriter	PRO/FMS	VMS
DIBOL	PRO/RMS	VT
digital™	PROSE	Work Processor
	PROSE PLUS	

CONTENTS

PREFACE xiii

CHAPTER 1 **INTRODUCTION**

1.1 PRO/TOOL KIT DCL 1-1
1.1.1 File/Device/Volume Manipulation Commands . 1-2
1.1.2 Program Development Commands 1-2
1.1.3 Task Manipulation Commands 1-2
1.1.4 Set and Show Commands 1-3
1.1.5 Miscellaneous Commands 1-3
1.2 PRO/TOOL KIT UTILITIES 1-3
1.2.1 Program Development, Frame, and Form Utilities 1-4
1.2.2 RMS Utilities 1-6
1.2.3 Other Utilities 1-7

CHAPTER 2 **USING PRO/TOOL KIT DCL**

2.1 THE DCL COMMAND LINE 2-1
2.1.1 Prompting 2-2
2.1.2 Qualifiers 2-3
2.1.3 HELP 2-5
2.1.4 Abbreviations 2-7
2.1.5 Colon and Equal Sign 2-7
2.1.6 Command Line Continuation 2-8
2.1.7 Comments in Command Lines 2-8
2.1.8 Errors 2-9
2.1.9 PRO/Tool Kit DCL Initialization and Termination Files 2-10
2.2 CORRECTING MISTAKES WITH THE DCL SINGLE LINE EDITOR 2-11
2.3 P/OS FILE SPECIFICATIONS IN DCL 2-13
2.3.1 Wildcard Features in DCL 2-13
2.3.2 Logical Name Translation 2-15
2.4 FOREGROUND AND BACKGROUND PROCESSING . . . 2-15
2.4.1 <CTRL/C> 2-15
2.4.2 ABORT 2-16

CHAPTER 3 **DCL COMMANDS**

3.1 ABORT 3-2
3.2 APPEND 3-4
3.3 ASSIGN 3-8
3.4 ASSIGN/TASK 3-12
3.5 BASIC 3-13

3.6 CANCEL 3-14
3.7 CLEAR 3-15
3.8 COBOL 3-16
3.9 CONTINUE 3-23
3.10 CONVERT 3-24
3.11 COPY 3-30
3.12 CREATE 3-37
3.13 CREATE/DIRECTORY 3-40
3.14 DEASSIGN 3-42
3.15 DEBUG 3-44
3.16 DEFINE 3-45
3.17 DELETE 3-49
3.18 DELETE/DIRECTORY 3-52
3.19 DIBOL 3-53
3.20 DIFFERENCES 3-57
3.21 DIRECTORY 3-61
3.22 DISMOUNT 3-65
3.23 DUMP 3-66
3.24 EDIT OR EDIT/EDT 3-71
3.25 EDIT/PROSE 3-74
3.26 EDIT/SLP 3-75
3.27 EXIT 3-78
3.28 FIX 3-79
3.29 FORMAT 3-81
3.30 FORTRAN 3-82
3.31 HELP 3-88
3.32 INITIALIZE 3-89
3.33 INSTALL 3-91
3.34 LIBRARY 3-93
3.35 LIBRARY/COMPRESS 3-94
3.36 LIBRARY/CREATE 3-96
3.37 LIBRARY/DELETE 3-99
3.38 LIBRARY/EXTRACT 3-100
3.39 LIBRARY/INSERT 3-101
3.40 LIBRARY/LIST 3-103
3.41 LIBRARY/REMOVE 3-105
3.42 LIBRARY/REPLACE 3-106
3.43 LINK 3-108
3.44 LINK/C81 3-123
3.45 LOAD 3-126
3.46 MACRO 3-127
3.47 MAIL 3-135
3.48 MOUNT 3-136
3.49 PASCAL 3-137
3.50 PHONE 3-140
3.51 PRINT 3-141
3.52 PRINT/REMOTE 3-142
3.53 PURGE 3-143
3.54 REMOVE 3-146
3.55 RENAME 3-147
3.56 RUN uninstalled task 3-150

3.57 RUN installed task 3-152
3.58 SET [DAY]TIME 3-155
3.59 SET DEFAULT 3-156
3.60 SET DEVICE 3-157
3.61 SET HOST 3-159
3.62 SET PRIORITY 3-160
3.63 SET PROTECTION 3-161
3.64 SET PROTECTION/DEFAULT 3-164
3.65 SET PROTECTION/NODEFAULT 3-166
3.66 SET TERMINAL 3-167
3.67 SHOW ASSIGNMENTS 3-172
3.68 SHOW CLOCK_QUEUE 3-173
3.69 SHOW COMMON 3-174
3.70 SHOW DAYTIME or SHOW TIME 3-176
3.71 SHOW DEFAULT 3-177
3.72 SHOW DEVICES 3-178
3.73 SHOW LOGICALS 3-179
3.74 SHOW MEMORY 3-180
3.75 SHOW PROTECTION 3-181
3.76 SHOW TASKS/ACTIVE 3-182
3.77 SHOW TASKS/INSTALLED 3-187
3.78 SHOW TASK:taskname/DYNAMIC 3-189
3.79 SHOW TASKS/ACTIVE/DYNAMIC 3-191
3.80 SHOW TASK/LOGICAL_UNITS 3-192
3.81 SHOW TERMINAL 3-193
3.82 SPAWN 3-194
3.83 START 3-195
3.84 START/UNBLOCK 3-196
3.85 STOP/BLOCK 3-197
3.86 SUBMIT/REMOTE 3-198
3.87 TYPE 3-199
3.88 UNLOAD 3-202
3.89 UNLOCK 3-203

CHAPTER 4 **THE INDIRECT COMMAND PROCESSOR**

4.1 INDIRECT COMMAND FILES 4-1
4.2 INDIRECT COMMAND PROCESSOR 4-2
4.3 SUMMARY OF INDIRECT DIRECTIVES 4-4
4.4 SYMBOLS 4-8
4.4.1 Special Symbols 4-9
4.4.1.1 Special Logical Symbols 4-9
4.4.1.2 Special Numeric Symbols 4-11
4.4.1.3 Special String Symbols 4-14
4.4.2 Numeric Symbols and Expressions 4-16
4.4.3 String Symbols, Substrings, and Expressions 4-18
4.4.4 Reserved Symbols 4-19
4.4.5 Symbol Value Substitution 4-19
4.4.5.1 Substitution Format Control 4-20

4.5 SWITCHES 4-21
4.6 Using Indirect Directives 4-24
4.6.1 Logical Tests 4-25
4.7 / - DEFINE LOGICAL END-OF-FILE 4-27
4.8 .ASK - ASK A QUESTION AND WAIT FOR A REPLY . 4-28
4.9 .ASKN - ASK FOR DEFINITION OF A NUMERIC SYMBOL 4-30
4.10 .ASKS - ASK FOR DEFINITION OF A STRING SYMBOL 4-33
4.11 .BEGIN - BEGIN BLOCK 4-35
4.12 .CHAIN - CONTINUE PROCESSING USING ANOTHER FILE 4-36
4.13 .CLOSE - CLOSE SECONDARY FILE 4-37
4.14 .DATA - OUTPUT DATA TO SECONDARY FILE . . . 4-38
4.15 .DEC - DECREMENT NUMERIC SYMBOL 4-39
4.16 .DELAY - DELAY EXECUTION FOR A SPECIFIED PERIOD OF TIME 4-40
4.17 .DISABLE - DISABLE OPTION 4-41
4.18 .ENABLE - ENABLE OPTION 4-42
4.19 .END - END BLOCK 4-46
4.20 .ERASE - DELETE SYMBOLS 4-47
4.21 .EXIT - EXIT CURRENT COMMAND FILE 4-49
4.22 .FORM - ACCESS FORM DRIVER 4-50
4.23 .GOSUB - CALL A SUBROUTINE 4-56
4.24 .GOTO - BRANCH TO A LABEL 4-57
4.25 .IF - TEST IF SYMBOL SATISFIES SPECIFIED CONDITION 4-58
4.26 .IFACT/.IFNACT - TEST IF TASK IS ACTIVE OR DORMANT 4-60
4.27 .IFDF/.IFNDF - TEST IF SYMBOL DEFINED OR NOT DEFINED 4-61
4.28 .IFINS/.IFNINS - TEST IF TASK INSTALLED OR NOT INSTALLED 4-62
4.29 .IFENABLED/.IFDISABLED - TEST IF MODE ENABLED OR DISABLED 4-63
4.30 .IFLOA/.IFNLO - TEST IF DRIVER LOADED OR NOT LOADED 4-64
4.31 .IFT/.IFF - TEST IF SYMBOL TRUE OR FALSE . . 4-65
4.32 .INC - INCREMENT NUMERIC SYMBOL 4-66
4.33 .label: - DEFINE A LABEL 4-67
4.34 .ONERR - BRANCH TO LABEL ON DETECTING AN ERROR 4-68
4.35 .OPEN - OPEN SECONDARY FILE 4-69
4.36 .OPENA - OPEN SECONDARY FILE FOR APPEND . . 4-70
4.37 .OPENR - OPEN FILE FOR READING 4-71
4.38 .PARSE - PARSE STRINGS INTO SUBSTRINGS . . . 4-72
4.39 .PAUSE - PAUSE FOR OPERATOR ACTION 4-73
4.40 .READ - READ NEXT RECORD 4-74
4.41 .RETURN - RETURN FROM A SUBROUTINE 4-76
4.42 .SETT/.SETF/.SETL - SET SYMBOL TO TRUE OR FALSE 4-77

4.43 .SETN - SET SYMBOL TO NUMERIC VALUE 4-78
4.44 .SETO/.SETD - SET SYMBOL TO OCTAL OR DECIMAL 4-79
4.45 .SETS - SET SYMBOL TO STRING VALUE 4-80
4.46 .STOP - TERMINATE COMMAND FILE PROCESSING . 4-82
4.47 .TEST - TEST SYMBOL 4-83
4.48 .TESTDEVICE - TEST DEVICE 4-85
4.49 .TESTFILE - TEST A FILE 4-86
4.50 .TESTPARTITION - TEST A PARTITION 4-87
4.51 .TESTSYSTEM - TEST SYSTEM 4-88
4.52 .TRANSLATE - TRANSLATE A LOGICAL NAME . . . 4-90
4.53 .WAIT - WAIT FOR A TASK TO FINISH EXECUTION 4-91
4.54 .XQT - INITIATE PARALLEL TASK EXECUTION . . 4-92
4.55 COMPATIBILITY WITH COMMAND FILES FROM RSX SYSTEMS 4-93
4.56 INDIRECT MESSAGES 4-94
4.56.1 Information-Only Messages 4-94
4.56.2 Error Messages 4-95

CHAPTER 5 **FILE COMPARE UTILITY (CMP)**

5.1 INVOKING CMP 5-2
5.2 CMP COMMAND FORMAT 5-2
5.3 CMP SWITCHES 5-3
5.4 FORMATS OF CMP OUTPUT FILES 5-6
5.4.1 Differences Format 5-7
5.4.2 Change Bar Format 5-8
5.4.3 SLP Command Input Format 5-8
5.5 CMP MESSAGES 5-9

CHAPTER 6 **FILE DUMP UTILITY (DMP)**

6.1 INVOKING DMP 6-2
6.2 DMP COMMAND FORMAT 6-2
6.3 DMP SWITCHES 6-3
6.4 DMP EXAMPLES 6-8
6.4.1 A Multiple Format Dump 6-8
6.4.2 A Record Dump 6-9
6.4.3 A Header Dump 6-10
6.5 DMP ERROR MESSAGES 6-11

CHAPTER 7 **LIBRARIAN UTILITY PROGRAM (LBR)**

7.1 FORMAT OF LIBRARY FILES 7-2
7.1.1 Library Header 7-2
7.1.2 Entry Point Table 7-3
7.1.2.1 Module Name Table 7-3
7.1.2.2 Module Header 7-3
7.2 LBR RESTRICTIONS 7-10

7.3 INVOKING LBR 7-11
7.4 DEFAULTS FOR LBR FILE SPECIFIERS 7-11
7.5 LBR SWITCHES 7-13
7.5.1 Compress Switch (/CO) 7-15
7.5.2 Create Switch (/CR) 7-16
7.5.3 Delete Switch (/DE) 7-18
7.5.4 Default Switch (/DF) 7-19
7.5.5 Delete Global Switch (/DG) 7-21
7.5.6 Entry Point Switch (/EP) 7-23
7.5.7 Extract Switch (/EX) 7-25
7.5.8 Insert Switch (/IN) for Object and Macro Libraries 7-26
7.5.9 Insert Switch (/IN) for Universal Libraries 7-27
7.5.10 List Switches (/LI, /LE, /FU) 7-29
7.5.11 Modify Header Switch (/MH) 7-30
7.5.12 Replace Switch (/RP) For Macro and Object Libraries 7-31
7.5.13 Replace Switch (/RP) for Universal Libraries 7-37
7.5.14 Selective Search Switch (/SS) 7-39
7.5.15 Squeeze Switch (/SZ) 7-40
7.6 COMBINING LIBRARY FUNCTIONS 7-42
7.7 LBR ERROR MESSAGES 7-43
7.7.1 Effect of Fatal Errors on Library Files . 7-44
7.7.2 LBR Error Messages 7-44

CHAPTER 8 **RESOURCE MONITORING DISPLAY (RMD)**

8.1 INTRODUCTION 8-1
8.1.1 Display Pages 8-1
8.1.2 Setup Pages 8-1
8.2 INVOKING RMD 8-2
8.2.1 Running RMD on a Second Terminal 8-3
8.3 THE HELP DISPLAY 8-3
8.4 THE MEMORY DISPLAY 8-4
8.4.1 Altering the Memory Display from the Setup Page 8-7
8.4.1.1 The FREE Command 8-7
8.4.1.2 The RATE Command 8-7
8.5 THE ACTIVE TASK DISPLAY 8-8
8.5.1 Altering the Active Task Display from the Setup Page 8-8
8.5.1.1 The OWNER Command 8-9
8.5.1.2 The PRIORITY Command 8-9
8.5.1.3 The RATE Command 8-9
8.5.1.4 The TASK Command 8-9
8.6 THE TASK HEADER DISPLAY 8-9
8.6.1 Altering the Task Header Display from the Setup Page 8-10

8.6.1.1 The RATE Command 8-11
8.6.1.2 The TASK Command 8-11
8.7 ERROR MESSAGES 8-11

CHAPTER 9 **TASK/FILE PATCH PROGRAM (ZAP)**

9.1 ZAP OPERATING MODES AND SWITCHES 9-2
9.1.1 The List Switch (/LI) 9-3
9.1.1.1 The /LI Switch and Regular Task Image Files 9-3
9.1.1.2 The /LI Switch and Multiuser Task Image Files 9-4
9.1.1.3 The /LI Switch and Resident Libraries 9-4
9.1.1.4 The /LI Switch and I- and D-Space Tasks 9-5
9.2 ADDRESSING LOCATIONS IN FILES 9-5
9.2.1 Relocation Biases 9-5
9.2.2 ZAP Addressing Modes 9-5
9.2.2.1 Using the Task Image Addressing Mode . . 9-7
9.2.2.2 Using the Absolute Addressing Mode . . . 9-7
9.3 INVOKING AND TERMINATING ZAP 9-7
9.3.1 Using Indirect Command Files with ZAP . . 9-8
9.4 THE ZAP COMMAND LINE AND COMMAND LINE ELEMENTS 9-8
9.4.1 ZAP Commands 9-9
9.4.1.1 Open/Close Location Commands 9-9
9.4.1.2 General Purpose Commands 9-10
9.4.1.3 <RETURN> Key 9-10
9.4.2 ZAP Internal Registers 9-10
9.4.3 ZAP Arithmetic Operators 9-11
9.4.4 ZAP Command Line Element Separators . . . 9-12
9.4.5 ZAP Command Line Location-Specifier Formats 9-12
9.4.5.1 The Current Location Symbol 9-13
9.4.5.2 Byte Offset Format 9-13
9.4.5.3 Block Number: Byte Offset Format 9-13
9.4.5.4 Relocation Register, Byte Offset Format 9-14
9.5 USING ZAP OPEN AND CLOSE COMMANDS 9-14
9.5.1 Opening Locations in a File 9-16
9.5.2 Changing the Contents of a Location . . . 9-16
9.5.3 Closing Locations in a File 9-17
9.5.3.1 Closing a Location and Opening the Preceding Location 9-17
9.5.3.2 Closing a Location and Opening an Offset Location 9-18
9.5.3.3 Closing a Location and Opening an Absolute Location 9-18
9.5.3.4 Closing a Location and Opening a Branch Target Location 9-19
9.5.3.5 Closing a Location and Opening a Previous Location 9-19

9.6 USING ZAP GENERAL PURPOSE COMMANDS 9-20
9.6.1 The X Command 9-20
9.6.2 The K Command 9-21
9.6.3 The O Command 9-22
9.6.4 The Equal Sign (=) Command 9-22
9.6.5 The V Command 9-23
9.6.6 The R Command 9-23
9.7 EXAMPLES 9-24
9.8 ZAP ERROR MESSAGES 9-31

CHAPTER 10 **SOURCE LANGUAGE INPUT PROGRAM (SLP)**

10.1 SLP INPUT AND OUTPUT FILES 10-2
10.1.1 The Input File 10-2
10.1.2 Command Input 10-3
10.1.3 The SLP Listing File 10-4
10.1.4 The SLP Output File 10-4
10.2 HOW SLP PROCESSES FILES 10-5
10.3 USING SLP 10-6
10.3.1 Specifying SLP Edit Commands 10-6
10.3.2 Entering SLP Edit Commands 10-8
10.3.2.1 Entering SLP Commands Interactively . . 10-9
10.3.2.2 Entering SLP Commands Using Indirect Command Files 10-10
10.3.2.3 Using SLP Operators 10-10
10.3.3 Updating Source Files With SLP 10-11
10.3.3.1 Generating a Numbered Listing 10-11
10.3.3.2 Adding Lines to a File 10-12
10.3.3.3 Deleting Lines from a File 10-14
10.3.3.4 Replacing Lines in a File 10-16
10.3.4 Creating Source Files Using SLP 10-17
10.4 CONTROLLING SLP 10-18
10.4.1 SLP Switches 10-18
10.4.2 Controlling the Audit Trail 10-19
10.4.3 Setting the Position and Length of the Audit Trail 10-20
10.4.4 Changing the Value of the Audit Trail . 10-21
10.4.5 Temporarily Suppressing the Audit Trail 10-22
10.4.6 Deleting the Audit Trail 10-23
10.5 SLP MESSAGES 10-24
10.5.1 SLP Information Message 10-24
10.5.2 SLP Error Messages 10-24

CHAPTER 11 **CONVERT UTILITY (CVT)**

CHAPTER 12 **PERIPHERAL INTERCHANGE PROGRAM (PIP)**

12.1 INVOKING PIP 12-1
12.2 FORMAT OF PIP COMMANDS 12-2
12.3 PIP DEFAULTS FOR FILE SPECIFICATION FIELDS . 12-3
12.4 PIP SWITCHES AND SUBSWITCHES 12-5
12.4.1 Switches 12-8
12.4.2 Subswitches 12-9
12.5 SPECIFYING WILDCARDS 12-10
12.5.1 Wildcards in Output File Specifications 12-10
12.5.2 Wildcards in Input Specifications . . . 12-11
12.6 PIP ERROR MESSAGES 12-13
12.7 PIP ERROR CODES 12-23

CHAPTER 13 **OBJECT MODULE PATCH UTILITY (PAT)**

13.1 INVOKING PAT 13-2
13.2 PAT COMMAND LINE FORMAT 13-2
13.3 HOW PAT APPLIES UPDATES 13-3
13.3.1 The Input File 13-4
13.3.2 The Correction File 13-4
13.3.3 How PAT and the Task Builder Update Object Modules 13-5
13.3.3.1 Overlaying Lines in a Module 13-5
13.3.3.2 Adding a Subroutine to a Module . . 13-7
13.3.4 Determining and Validating the Contents of a File 13-8
13.4 PAT MESSAGES 13-9
13.4.1 Information Messages 13-10
13.4.2 Command Line Errors 13-10
13.4.3 File Specification Errors 13-11
13.4.4 Input/Output Errors 13-13
13.4.5 Errors in File Contents or Format . . . 13-15
13.4.6 Internal Software Error 13-16
13.4.7 Storage Allocation Error 13-17

CHAPTER 14 **FILE STRUCTURE VERIFICATION UTILITY (VFY)**

14.1 INVOKING VFY 14-1
14.2 VFY COMMAND FORMAT 14-2
14.3 VFY MODE OF OPERATION 14-3
14.4 VFY VALIDITY CHECK 14-3
14.5 VFY SWITCHES 14-4
14.5.1 Directory Validation Switch (/DV) 14-4
14.5.2 Free Switch (/FR) 14-5
14.5.3 List Switch (/LI) 14-6
14.5.4 Lost Switch (/LO) 14-6
14.5.5 Read Check Switch (/RC) 14-6
14.6 FILE ERROR REPORTING 14-7

14.6.1 Files Marked for Delete 14-9
14.6.1.1 Deleting a File Marked for Delete 14-10
14.6.2 Deletion of Multiple-Allocated Blocks . 14-10
14.6.3 Recovering Lost Blocks 14-10
14.7 VFY ERROR MESSAGES 14-11

APPENDIX A **FUNCTIONS INITIATED BY DCL COMMANDS**

APPENDIX B **ERROR MESSAGES**

B.1 GENERAL ERROR MESSAGES B-1
B.2 I/O ERROR MESSAGES B-19

APPENDIX C **SAMPLE EDT INITIALIZATION FILE**

FIGURES

7-1 General Format for Object and Macro Library Files 7-4
7-2 Universal Library File Format 7-5
7-3 Contents of Library Header 7-6
7-4 Format of Entry Point Table Element 7-7
7-5 Format of Module Name Table Element 7-7
7-6 Module Header Format for Object and Macro Libraries 7-8
7-7 Module Header Format for Universal Libraries 7-9
7-8 Sample Files Used in LBR Examples 1-4 . . . 7-35
7-9 Output Library File After Execution of Example 1 7-35
7-10 Output Library File After Execution of Example 2 7-36
7-11 Output Library File After Execution of Example 3 7-36
7-12 Sample Files for Universal Library Replace Example 7-38
7-13 Output Library File After Library Replace . 7-38
7-14 MACRO Listing Before and After Running LBR with /SZ 7-42
8-1 Memory Display for P/OS 8-4
11-1 Output of the CVT Utility 11-1
13-1 Using PAT 13-3

TABLES

3-1	Argument Values for /SHOW Qualifier	3-55
3-2	The Enable and Disable Qualifiers	3-130
3-3	The /SHOW and /NOSHOW Qualifiers	3-132
3-4	Status Bits	3-175
3-5	Display for SHOW TASKS/ACTIVE	3-183
3-6	Task Status Flags	3-184
5-1	Summary of CMP Default Switch Settings	5-5
7-1	LBR File Specifiers Defaults	7-12
7-2	LBR Switches	7-14
9-1	ZAP Arithmetic Operators	9-11
9-2	ZAP Command Line Element Separators	9-12
9-3	ZAP Open and Close Commands	9-14
9-4	ZAP General Purpose Commands	9-20
10-1	SLP Operators	10-11
10-2	SLP Switches	10-18
11-1	Punctuation to Specify Radix	11-2
12-1	PIP Default File Specifications	12-4
12-2	PIP Switches and Subswitches	12-6
12-3	PIP Error Codes and Messages	12-24
14-1	VFY Switches and Functions	14-4
A-1	Functions Initiated by DCL Commands	A-1
B-1	I/O Error Messages	B-20

PREFACE

Manual Objectives

After reading this manual, you will be able to use the command language and utilities included in the PRO/Tool Kit.

This book does not describe how to use the Professional computer, nor does it describe how to write applications. For information on using the Professional, including installing your application, see the *Professional 300 Series User's Guide for Hard Disk System*. For information on writing applications, see the *Tool Kit User's Guide*.

Intended Audience

Readers who already know a command language will benefit most from this manual. You should have a working knowledge of the application development cycle as described in the *Tool Kit User's Guide*.

Structure of This Document

This manual consists of the following chapters:

- **Chapter 1** provides an introduction to the PRO/Tool Kit Digital Command Language (DCL) and an explanation of DCL command line format.
- **Chapter 2** describes the structure of the DCL command line, operation of the DCL Single Line Editor (SLE), how to use P/OS file specifications, and the foreground and background processing feature of P/OS.
- **Chapter 3** lists alphabetically the PRO/Tool Kit DCL commands, and includes a description and syntax for invocation of each command.
- **Chapter 4** details the Indirect Command Processor, which processes command files.
- **Chapter 5** describes the File Compare (CMP) utility, which compares the contents of two ASCII files.

- **Chapter 6** explains the File Dump (DMP) utility, which allows you to examine the contents of a file or volume of files.

- **Chapter 7** describes the Librarian (LBR) utility, which allows you to create, update, modify, list, and maintain library files.

- **Chapter 8** describes the Resource Monitoring Display (RMD), which allows you to display information about the resources in your system.

- **Chapter 9** describes the Task/File Patch Program (ZAP), which allows you to directly examine and modify task image and data files.

- **Chapter 10** describes the Source Language Input Program (SLP), a utility that you can use to maintain and audit source files.

- **Chapter 11** shows you how to convert numbers among different radixes using the Convert Utility (CVT).

- **Chapter 12** tells you how to perform file control operations using the Peripheral Interchange Program (PIP).

- **Chapter 13** describes how to update, or patch, code in a relocatable binary object module using the Object Module Patch Utility (PAT).

- **Chapter 14** tells you how to use the File Structure Verification Utility (VFY), which allows you to check the readability and validity in files, directories, and volumes.

- **Appendix A** contains functions initiated by DCL commands.

- **Appendix B** lists the common error messages for PRO/Tool Kit DCL and the utilities.

- **Appendix C** contains an EDT command file example.

Associated Documents

- **PRO/Tool Kit Installation Guide and Release Notes**

  For directions on installing the PRO/Tool Kit, as well as information specific to the current version of the PRO/Tool Kit, read this document.

- **Professional 300 Series User's Guide for Hard Disk System**

  You should read this manual if you are not familiar with the Professional.

- **Tool Kit User's Guide**

  This manual details the program development cycle, and provides primary information on devices, volumes, and file specifications.

Conventions Used in This Document

Convention/Term	Meaning
[optional]	In a command line, square brackets indicate that the enclosed item is optional. In a file specification, however, square brackets are part of the required syntax.
UPPERCASE	Uppercase words and letters, used in examples, indicate that you should type the word or letter exactly as shown.
lowercase	Lowercase words and letters, used in format examples, indicate that you should substitute a word or value of your own. Usually the lowercase word identifies the type of substitution required.
[s] or [,s]	If you can repeat an item in a command line, then an [s] appears after the item to show its plural form. A comma or other punctuation preceding the letter "s" indicates that you must separate repeated items with that punctuation.
. . .	A vertical ellipsis in a figure or example means that not all of the statements are shown.

Convention/Term	Meaning
<CR>	This manual occasionally shows carriage returns, represented by <CR>, to delineate an example clearly.
<CTRL/x>	This notation indicates that you are to press the key marked "Ctrl" while simultaneously pressing another key on the keyboard.
red	Interactive user input appears in red.
BOLD	In command syntax, the **bold** portion indicates the minimally-acceptable abbreviation for the command.
Tool Kit	This general term refers to the software you use to develop applications to run on a Professional computer.
Host Tool Kit	The Host Tool Kit is Tool Kit software that runs on a host computer, rather than on the Professional itself.
PRO/Tool Kit	The PRO/Tool Kit is the Tool Kit software that runs on the Professional computer.

CHAPTER 1

INTRODUCTION

The PRO/Tool Kit is an application that allows you to perform program development on the Professional. It includes PRO/Tool Kit DCL, a command language, as well as several utility programs that aid you in the development process. This chapter introduces PRO/Tool Kit DCL and the PRO/Tool Kit utilities.

For information on hardware and software requirements for installing the PRO/Tool Kit, as well as installation instructions, see the *PRO/Tool Kit Installation Guide and Release Notes*.

1.1 PRO/TOOL KIT DCL

The PRO/Tool Kit DCL is a subset of Digital Command Language implementations found on the VAX/VMS and RSX-11M-PLUS operating systems. Although some commands are unique to the PRO/Tool Kit, many of the commands derive from both of these operating systems.

PRO/Tool Kit DCL provides you with an extensive set of commands to perform various functions. You can divide the PRO/Tool Kit DCL commands into five groups:

File Manipulation
Program Development
Task Manipulation
Set and Show
Miscellaneous

The following sections describe the commands within each functional group. Chapter 3 presents the commands in alphabetical order.

1.1.1 File/Device/Volume Manipulation Commands

The file manipulation commands are:

APPEND
ASSIGN
BAD
CONVERT
COPY
CREATE
DIRECTORY
DEASSIGN
DEFINE
DELETE
DELETE/DIRECTORY
DIFFERENCES
DIRECTORY
DISMOUNT
DUMP
EDIT
EDIT/PROSE
EDIT/SLP
FORMAT
INITIALIZE
MOUNT
PRINT
PURGE
RENAME
TYPE
UNLOCK

1.1.2 Program Development Commands

The program development commands are:

BASIC
COBOL
DEBUG
DIBOL
FORTRAN
LIBRARY
LIBRARY/COMPRESS
LIBRARY/CREATE
LIBRARY/DELETE
LIBRARY/EXTRACT
LIBRARY/INSERT
LIBRARY/LIST
LIBRARY/REMOVE
LIBRARY/REPLACE
LINK
LINK/C81
LOAD
MACRO
PASCAL
UNLOAD

1.1.3 Task Manipulation Commands

The task-related commands are:

ABORT
ASSIGN/TASK
CANCEL
CONTINUE
FIX
INSTALL
REMOVE
RUN uninstalled task
RUN installed task
SPAWN
START
START/UNBLOCK
STOP/BLOCK

1.1.4 Set and Show Commands

The set and show commands are:

```
SET [DAY]TIME
SET DEFAULT
SET DEVICE
SET HOST
SET PRIORITY
SET PROTECTION
SET TERMINAL
SHOW ASSIGNMENTS
SHOW CLOCK_QUEUE
SHOW COMMON
SHOW [DAY]TIME
SHOW DEFAULT
SHOW DEVICES
SHOW LOGICALS
SHOW MEMORY
SHOW TASKS
SHOW TASKS/ACTIVE
SHOW TASKS/INSTALLED
SHOW TASKS/DYNAMIC
SHOW TASK:taskname/DYNAMIC
SHOW TASKS/ACTIVE/DYNAMIC
SHOW TASK/LOGICAL_UNITS
SHOW TERMINAL
```

1.1.5 Miscellaneous Commands

The miscellaneous commands are:

```
CLEAR
EXIT
HELP
MAIL
PHONE
```

1.2 PRO/TOOL KIT UTILITIES

The PRO/Tool Kit utilities comprise the following:

- The Compare Utility (CMP)
- The EDT, PROSE, and SLP editors
- Frame Development Tool (FDT)
- The PRO/FMS-11 Forms Editor (PROFED)
- The PRO/FMS-11 Forms Utility (PROFUT)
- The Librarian Utility (LBR)
- The Resource Monitoring Display (RMD)

- Various RMS utilities
- The Task/File Patch Program (ZAP)
- The Convert Utility (CVT)
- The Peripheral Interchange Processor (PIP)
- The Object Module Patch Utility (PAT)
- The Verify Utility (VFY)

You can invoke most of these utilities through a corresponding DCL command. Those utilities that you cannot call through DCL must be invoked directly. Refer to the specific utility below for instructions.

1.2.1 Program Development, Frame, and Form Utilities

These utilities allow users to perform the following functions:

- Edit text files using a text editor (such as EDT)
- Develop menus with the Frame Development Tool (FDT)
- Assemble MACRO-11 source files with the Professional MACRO-11 Assembler (PMA)
- Build applications with the Professional Application Builder (PAB)
- Invoke the Record Management Services

You should refer to the corresponding Tool Kit Documentation for a complete description and operating instructions for these utilities.

- **Text Editors** To invoke EDT, the DIGITAL standard text editor, use the commands EDIT, EDIT/EDT, or the following command:

```
$ RUN $EDT
```

  If you prefer, you can use the PROSE editor instead of EDT. To invoke PROSE, use the command EDIT/PROSE. In addition, the SLP editor is available; to invoke SLP, use the command EDIT/SLP or the command RUN $SLP.

- **FDT (Frame Development Tool)**

  FDT creates interactive menus for P/OS applications. The following command invokes this utility:

  ```
  $ RUN $FDT
  ```

  For complete details on FDT, see the *Tool Kit Reference Manual*.

- **PROFED (PRO/FMS-11 Forms Editor)**

  PROFED is an editor that lets you create and modify video forms on the screen. The use of the PRO/FMS-11 Forms utility (PROFUT) translates these forms into files for use by an application program. The following command invokes this utility:

  ```
  $ RUN $PROFED
  ```

  See the *FMS-11/RSX Software Reference Manual* for details on using PROFED.

- **PROFUT (PRO/FMS-11 Forms Utility)**

  PROFUT creates form description files that are the output from the PRO/FMS-11 forms editor. An application program can display these files. The following command invokes this utility:

  ```
  $ RUN $PROFUT
  ```

  See the *FMS-11/RSX Software Reference Manual* for details on using PROFUT.

- **PAB (Professional Application Builder)**

  The Professional Application Builder links one or more object modules into an executable task image, which is the final form of any application or system program. PAB resolves addressing and overlay considerations. To invoke PAB in DCL, use the LINK command or the command RUN $PAB.

  PAB is based on the RSX-11M/M-PLUS Task Builder. Refer to the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual* for additional information on PAB.

- **PMA (Professional MACRO-11 Assembler)**

  PMA assembles and lists MACRO-11 source files. The MACRO command, or the command RUN $PMA, invokes this utility. Refer to Chapter 3 and the *PDP-11 MACRO-11 Language Reference Manual* for further details on the MACRO-11 assembler).

1.2.2 RMS Utilities

The PRO/Tool Kit contains the utilities listed in this section.

NOTE

Refer to the *RSX-11M/M-PLUS RMS-11 Utilities Manual* for a description of the RMS utilities. Only those RMS utilities listed below are supported on the PRO/Tool Kit.

- **RMSCNV**

  RMSCNV is the RMS-11 file conversion utility that moves records between two RMS-11 files of any organization or record format. You can invoke RMSCNV by typing the following:

```
$ RUN $RMSCNV
```

  You can also use the DCL command CONVERT.

- **RMSDES**

  RMSDES allows you to design and create indexed, sequential, and relative files. You can specify the file attributes interactively or read in the attributes of an existing, external data file that you want to re-create with little or no modification. You can invoke RMSDES by typing the following:

```
$ RUN $RMSDES
```

  There is no other DCL command to invoke RSMDES.

- **RMSDSP**

  RMSDSP lists RMS-11 file attributes and structural data. You invoke this utility by typing the following:

  ```
  $ RUN $RMSDSP
  ```

  You can also use the DCL command DIRECTORY/ATTRIBUTES.

- **RMSIFL**

  RMSIFL indexed file load utility builds an indexed file using records from another RMS-11 file of any organization type. This utility uses techniques that are derived from the basic structure of indexed files, rather than from the standard RMS-11 file structure. You invoke this utility by typing the following:

  ```
  $ RUN $RMSIFL
  ```

  There is no other DCL command to invoke RMSIFL.

1.2.3 Other Utilities

Chapters 5 through 14 fully describe the CMP, DMP, LBR, RMD, ZAP, SLP, CVT, PIP, PAT, and VFY utilities, respectively. The syntax you use to invoke these is:

```
$ RUN $utility-name
```

The utility then displays its prompt.

CHAPTER 2

USING PRO/TOOL KIT DCL

You can use PRO/Tool Kit DCL interactively by typing a command at the keyboard, or you can invoke a DCL command file using the Indirect Command Processor (refer to Chapter 4). Whichever method you use, the format of the command is the same.

You enter your commands at the DCL command level (indicated by the $ prompt), and press either the <RETURN> or <DO> key to execute the command. Because P/OS waits for a carriage return before processing the DCL command, you can edit the command line before pressing <RETURN>. (See Section 2.2.)

You are only required to type as many letters of the DCL command as needed to form a unique command. For example, the abbreviation SET TE/UP is a valid, unique command, because there are no duplications of this within the SET TERMINAL command or within the PRO/Tool Kit Command Language (refer to SET TERMINAL in Chapter 3). However, SET TE/IN would not be a unique command, because this abbreviation could stand for either SET TERMINAL/INQUIRE or SET TERMINAL/INTERACTIVE. (Refer to Section 2.1.4 for additional information on abbreviating DCL commands.)

2.1 THE DCL COMMAND LINE

This section introduces the rules governing the use of DCL. The examples in this section are intended to illustrate these rules, not to document the full capabilities of the commands. For more detail, see the individual command descriptions in Chapter 3.

A command consists of a command name, or verb, describing the action the system is to take. Most commands also include one or more parameters and qualifiers to further define the action of the command. Qualifiers are preceded by a slash (/) and parameters are preceded by a space. Both qualifiers and parameters can take arguments. Arguments are preceded by a colon (:). The format of a PRO/Tool Kit DCL command is:

```
$ Command/cmd-qual[s] param1[/param-qual] param2[/param-qual]
```

where:

$	indicates the DCL command level prompt.
Command	is the name of the DCL command.
/cmd-qual	is an optional command qualifier that directs the command either to perform or omit auxiliary functions.
param1	is a parameter that you pass to the command for processing (not required by all commands).
param-qual	is an optional parameter qualifier that causes the DCL command to perform or omit auxiliary functions on the parameter.
param2	is a second parameter that you pass to the DCL command for processing (not required by all commands).

Either the <RETURN> or <DO> key is the terminator that passes DCL commands to the operating system. Unless the action of the <RETURN> key differs greatly from what is expected, the command examples and formats in this manual do not include an indication that each command line is terminated by a carriage return. (See Section 2.1.6 for a discussion of command lines that are too long to fit on one line of your terminal.)

Some commands require parameters or arguments as part of the command line. If you fail to supply a required command element, DCL prompts you with one or two words indicating the general nature of the required element. If you do not understand the prompt, type a question mark (?) for help. (In some cases, an omission causes an error rather than a prompt.)

2.1.1 Prompting

The prompts help you learn the form of a command by requesting that you supply required command elements.

For example, the RENAME command works as follows:

```
$ RENAME
From?  FILE1.LIS
To?  FILE2.MAC
```

The one-line format for RENAME is:

```
$ RENAME FILE1.LIS FILE2.MAC
```

The formats can be mixed. DCL prompts for whatever you leave out. For example:

```
$ RENAME FILE1.LIS
To? FILE2.MAC
```

There are no defaults for prompts. You must supply a response to any prompt. If you do not want to continue with the command, type a CTRL/Z.

2.1.2 Qualifiers

Qualifiers modify the action of the command. Qualifiers always start with a slash (/) and are generally optional.

Qualifiers are either command qualifiers or parameter qualifiers. Most qualifiers are command qualifiers. In this manual, command qualifiers are always shown immediately following the command verb, as in this example:

```
$ TYPE/TODAY *.HLP
```

However, most command qualifiers can appear anywhere in the command line. Another name for these qualifiers is floating qualifiers. The following examples illustrate how command qualifiers can float:

```
$ TYPE *.HLP/TODAY
```

or

```
$ TYPE
File(s)? *.HLP/TODAY
```

or

```
$ TYPE
File(s)? /TODAY
File(s)? *.HLP
```

You can mix formats, as in the following example:

```
$ TYPE/TODAY
File(s)? *.HLP/EXCLUDE:HELPF.HLP;*
```

or

```
$ TYPE/TODAY
File(s)? *.HLP/
Qualifier? EXCLUDE:HELPF.HLP;*
```

or

```
$ TYPE/TODAY
File(s)? /
Qualifier? EXCLUDE:HELPF.HLP;*
File(s)? *.HLP
```

Note that you are prompted for a qualifier when a slash with no qualifier attached appears on the command line. When you supply the qualifier, do not type the slash again.

Regardless of where the qualifier appears, it has the same effect.

Qualifiers described as parameter qualifiers, or filespec qualifiers, cannot float.

Parameter qualifiers do not modify the action of the command; they modify the action of the command as it concerns that particular parameter, or supply additional information needed for the command to execute properly. For instance, in the following example:

```
$ MACRO HIYA, TESTBLD/LIBRARY, PIGEON
```

the qualifier /LIBRARY identifies a particular file as being a library, and the /LIBRARY qualifier cannot float.

Many qualifiers can be negated by prefixing NO or - (minus) to the qualifier name. Thus, the command:

```
$ MACRO/OBJECT SIMPLE.MAC
```

directs the MACRO-11 Assembler to make an object file, while the command:

```
$ MACRO/NOOBJECT SIMPLE.MAC
```

or

```
$ MACRO/-OBJECT SIMPLE.MAC
```

directs the MACRO-11 Assembler to omit the object file. In the first example, /OBJECT is the default qualifier and need not be explicitly included. In other words, unless your MACRO command includes the /NOOBJECT qualifier, an object file is produced.

2.1.3 HELP

HELP is available for all DCL commands by entering the HELP command or by typing a question mark (?) in response to any DCL prompt. For instance, if you want help on the TYPE command, enter the following:

```
$ HELP TYPE
```

The following HELP text is displayed:

```
 TYPE[/qualifier[s]] filespec[s]
        /DATE:dd-mmm-yy
        /SINCE:dd-mmm-yy
        /THROUGH:dd-mmm-yy
        /SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
        /TODAY
        /EXCLUDE:filespec

     The TYPE command displays the contents of text files on your
     terminal.
```

If you wish help on one of the qualifiers for TYPE, enter the following:

```
$ HELP TYPE TODAY

 TYPE/TODAY filespec[s]

     The /TODAY qualifier specifies that you wish the TYPE
     command to type only files created today.
```

The HELP text consists of a brief explanation of the command followed by an illustration of the syntax, showing that TYPE accepts one or more filespecs, and one or more qualifiers.

If you wish help while being prompted by the TYPE command, use the following procedure:

```
$ TYPE
File(s)? ?
 TYPE[/qualifier[s]] filespec[s]
        /DATE:dd-mmm-yy
        /SINCE:dd-mmm-yy
```

```
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec

    The TYPE command displays the contents of text files on your
    terminal.

    File(s)?
```

The same help text is printed on your terminal, but the prompt returns, meaning the TYPE command is still waiting for you to list the file(s) you wish typed.

You can also get help on a specific subtopic while being prompted by a command, by responding to the prompt with a question mark. For example:

```
$ SET
Function? ?
 SET thing

 The SET command can be used to set something. The
 following things can be set with this command:

 [DAY]TIME    DEFAULT    DEVICE    PRIORITY    PROTECTION
 TERMINAL
```

To get help on a specific subtopic of the SET command, enter a question mark followed by the subtopic:

```
Function? ? DEFAULT

 SET DEFAULT [ddnn:][directory]

 The SET DEFAULT command sets your default directory or
 device, or both.

Function?
```

You can also get help by typing a question mark in response to the DCL prompt ($).

If you should decide after reading the help text that you have chosen the wrong command, enter a <CTRL/Z> in response to the prompt to end the execution of the command. (A <CTRL/Z> in response to a prompt always cancels execution of the current DCL command.)

2.1.4 Abbreviations

It is rarely necessary for you to type either the complete command name or the complete qualifier name. You only need to type the characters required to distinguish the command or qualifier from all others.

For example:

- TYPE can be abbreviated as T because it is the only command beginning with that character.
- DELETE can be abbreviated as DEL, but not DE.
- DEASSIGN can be abbreviated as DEA, but not DE.

Three letters is usually enough. Five letters is always enough. You can often omit other parts of commands as well.

The underscore character (_) is used to make DCL commands more readable where two words are needed to name a single command element, such as EDIT/READ_ONLY. However, you need not type the underscore to enter the command. EDIT/READ_ONLY is the same as EDIT/READONLY.

Here are the brief forms for some frequently used commands.

A for ABORT	C for COPY	D for DIRECTORY
DEA for DEASSIGN	E for EDIT	F for FORTRAN
H for HELP	L for LINK	M for MACRO
R for RUN	S for SHOW	T for TYPE

The command descriptions in Chapter 3 indicate the minimum acceptable abbreviation for each command.

2.1.5 Colon and Equal Sign

The command descriptions in this manual show arguments set off by a colon (:). You can always replace such colons with an equal sign (=), as in this example:

```
$ DIFFERENCES/LINES=2 testfile.tmp
```

Refer to Section 3.20 for a description of the DIFFERENCES command. Colons in device names, such as DW1:, and so forth, cannot be replaced by equal signs.

2.1.6 Command Line Continuation

The hyphen (-) is used to indicate line continuation. When you end a command line with a hyphen and a carriage return, the DCL continuation prompt ($-) indicates that you can continue entering the command line. If you are continuing a line from a prompt, such as:

```
Task?
```

that prompt is the indication that the line is being continued. This feature permits you to enter command lines including more characters than your terminal has room for on one line. No DCL command line can be longer than 250 characters. Here is an example of line continuation:

```
$ COPY PROCOM1.PAS,PROCOM2.PAS,PROCOM3.PAS,-<CR>
$- BIGFILE.PAS [WORK]
```

The command is not entered until DCL encounters a line ending with a carriage return not preceded by a hyphen. In the example, the first carriage return does not enter the command because it is preceded by a hyphen. The command is entered following the second carriage return. The carriage return can be on a line by itself.

2.1.7 Comments in Command Lines

You can include comments in a DCL command line using the exclamation point (!).

If the comment ends the command line, only a single exclamation point is needed, as in this example:

```
$ COPY PROCOM1.PAS PROCOM1.TMP !  Temporary file
```

If the comment is within the command line, two exclamation points are needed, as in this example:

```
$ COPY !Temporary File!  PROCOM1.PAS PROCOM1.TMP
```

These comments are ignored and not interpreted in any way by DCL.

Comments can be placed at any natural break in the command line: between qualifiers, between parameters, even as part of a response to a prompt.

2.1.8 Errors

You can correct typing errors or delete the line completely by using the <DELETE> key, or <CTRL/U> (provided you have not terminated the line).

You can cancel the execution of any DCL command by typing a <CTRL/Z>.

If the system detects an error in the command line input, it returns the appropriate error message.

Here are some examples of incorrect commands and the error messages they produce:

```
$ KOPY PRO300.LIS PRINT.LIS
DCL -- Illegal command

$ RENAME/COPY  TEST.PAS PROCOM1.PAS
RENAME -- Illegal or contradictory qualifier
RENAME/COPY TEST.PAS PROCOM1.PAS
     ^


$ DIFFERENCES/LINES=TWO TEST.PAS PROCOM1.PAS
DIFFERENCES -- Numeral expected
DIFFERENCES/LINES=TWO TEST.PAS PROCOM1.PAS
                  ^
```

In the first case, the error was detected by DCL, as indicated by the first part of the error message. There is no DCL KOPY command. The entire command was rejected.

In the second case, the command was entered correctly, but the qualifier was incorrect. The first part of the message shows that the error was detected within the RENAME command itself. The command is reprinted and a circumflex (^) points to the error.

In the third case, the command and qualifier were correct, but the argument was in error. The message explains the error and the circumflex points to the error.

Sometimes the circumflex does not point directly at the error, but at the point at which the command started to go wrong, which may be several characters before the actual error.

Typing mistakes are by far the most common cause of errors. Retyping the command or using the <UP ARROW> and the Single Line Editor (SLE) often eliminates the error. (See the discussion of the Single Line Editor, below.) Other common causes of errors are omitting a space or other delimiter in a command line, specifying

invalid devices or nonexistent files, and failing to type a sufficient number of characters to distinguish the command or command element.

All the DCL error messages are listed and explained in Appendix B of this manual. In some cases, such as assembly errors, you may need to go to another manual for explanation of the error.

2.1.9 PRO/Tool Kit DCL Initialization and Termination Files

When you enter the PRO/Tool Kit, two indirect command files automatically execute. Similarly, two indirect command files automatically execute when you leave the PRO/Tool Kit. These initialization and termination files install and remove tasks that you need for developing applications on the Professional.

The initalization files execute in the following order when you enter the PRO/Tool Kit:

1. **LB:[ZZPRODCL]START.CMD**

   This is the first command file to execute. It executes automatically for all users of the PRO/Tool Kit. Commands that should always be executed during PRO/Tool Kit initialization are located in this file. Only a system manager should modify this file. For example, if all PRO/Tool Kit users on a system require a particular library, the system manager could modify LB:[ZZPRODCL]START.CMD to install that library.

2. **DCLAPPL$DIR:START.CMD**

   This command file is located in a user-specific area, DCLAPPL$DIR, and so it can be different for each user. For instance, if you are the only user who needs a certain library installed, you can modify DCLAPPL$DIR:START.CMD to perform the installation during your PRO/Tool Kit session. Alternatively, you might place the comment character before lines that install tasks you do not use.

The termination command files execute in the following order when you leave the PRO/Tool Kit:

1. **LB:[ZZPRODCL]EXIT.CMD**

   This command file is provided so that you can specify commands to be executed on a system-wide basis during DCL termination. The termination file corresponds to the initialization file LB:[ZZPRODCL]START.CMD.

2. **DCLAPPL$DIR:EXIT.CMD**

   This command file is provided so that you can specify commands to be executed on a user-specific basis during DCL termination. The termination file corresponds to the initialization file DCLAPPL$DIR:START.CMD.

See Chapter 4 for information on indirect command files.

2.2 CORRECTING MISTAKES WITH THE DCL SINGLE LINE EDITOR

This is a brief description of how the Single Line Editor works.

The Single Line Editor can be used in two ways:

- If you make a mistake before you press either <RETURN> or <DO>, you can correct it without retyping the entire command. To correct the command, press the <LEFT ARROW> key until the cursor is just to the right of any character(s) you want to change. Delete the part you do not want with the <X] key and retype. To add a part you left out, move the cursor to the place and type in the omitted part. The rest of the command will move to the right to make room.

- If you make a mistake after you have pressed either <RETURN> or <DO>, press the <UP ARROW> key. The command will be redisplayed for changing. You can display up to nine previous commands this way. You can then use the <DOWN ARROW> key to display the next command.

  While entering DCL commands, the <CANCEL> key erases the current command so you can start over. The <F13> function key erases from where the cursor is to the end of the line.

Normal text characters simply get entered into the command line at the position of the cursor. If there are any characters on, or to the right of the cursor, they will move to the right to make room for the new character.

CORRECTING MISTAKES WITH THE DCL SINGLE LINE EDITOR

The following list describes what keys on the keyboard perform what functions in the single line editor.

<X]	Erases the character to the left of the cursor. If there are any characters on, or to the right of the cursor, they will move to the left to close the space for the erased character.
<REMOVE>	Erases the character on the cursor. If there are any characters to the right of the cursor, they will move to the left to close the space for the erased character.
<RETURN> <DO>	Enters the current command, giving it to the DCL command interpreter for execution. After the command has executed, the line following the one that was just entered (if there is one) will be displayed, with the cursor at the end of the line. To execute it, press <RETURN> or <DO>. To skip over it, press the down arrow.
<LEFT ARROW> ← <RIGHT ARROW> →	Moves the cursor one column to the left or to the right.
<UP ARROW> ↑	Stores the current command, and displays the previous command, leaving the cursor at the end. If there is no previous command, no action is taken.
<DOWN ARROW> ↓	Stores the current command, and displays the next command. If there are no more commands, then just start an empty line.
<F13>	Erases the command from the cursor position to the end of the line.
<EXIT>	Leaves the PRO/Tool Kit and returns to the P/OS Main Menu.
<HELP>	Appends "HELP" to the beginning of the current command, and enters it.
<CANCEL> <CTRL/U> <CTRL/C>	Erases the entire current command, and starts with a blank command.
<CTRL/R> <CTRL/W>	Redisplays the current command.

2.3 P/OS FILE SPECIFICATIONS IN DCL

File specifications, often called filespecs, are required for many DCL commands. The format of a file specification is as follows:

```
nodespec::device:[directory]filename.type;version
```

Note that the nodespec portion of a file specification is valid only on those commands supported by PRO/DECnet, as indicated in the command descriptions in Chapter 3. Your Professional must have PRO/DECnet installed in order to use the nodespec to operate on files at other nodes on the network.

See the *Tool Kit User's Guide* for a complete description of file specifications.

2.3.1 Wildcard Features in DCL

PRO/Tool Kit DCL offers wildcard features that are in addition to the simple wildcard capability available throughout P/OS (see the *Tool Kit User's Guide*). In the simple form of wildcarding, you use the * to replace an entire field in a filespec.

Many commands described in this manual work through PIP, the Peripheral Interchange Program. For the PIP-related commands--DIRECTORY, DELETE, PURGE, COPY, RENAME, TYPE, APPEND, UNLOCK, and SET PROTECTION--a more elaborate form of wildcarding is available. In these commands, within filenames and file types, the * can be used in a more complex manner. The * actually means "match zero or all characters in this position."

Therefore, the command:

```
$ DIRECTORY L*.TXT
```

lists the most recent versions of all files with the type .TXT whose names start with L on the default volume and in the default directory.

And the command:

```
$ DIRECTORY *L*.TXT
```

lists the most recent versions of all files with the type .TXT whose names include an L on the default volume and in the default directory.

The same substitutions can also be used in file types, so that the command:

```
$ DIRECTORY SNOBLO.L*
```

lists the most recent versions of all files with the name SNOBLO and the type beginning with an L on the default volume and in the default directory.

You can use more than one wildcard in filenames and file types.

The command:

```
$ DIRECTORY *F*D*.TXT
```

lists the most recent versions of all files with the type .TXT whose names include an F and a D in that order.

In addition, the PIP-related commands permit the percent sign (%) to be used as a wildcard, but only within filenames and file types. The % means "match exactly one character in this position."

For instance, the following command:

```
$ DIRECTORY %.TXT
```

lists all files with the type .TXT and a single-character filename on the default volume in the default directory.

The command:

```
$ DIRECTORY NOV%%81.TXT
```

lists all files with the type .TXT and a filename consisting of NOV and 81 separated by two characters on the default volume in the default directory.

The wildcards can be combined in a single filespec.

The command

```
$ DIRECTORY %L*T.TM%
```

lists all files whose names begin with a single character followed by an L and end with a T, and with a file type consisting of .TM and another single character on the default volume in the default directory.

Wildcarding, combined with systematic usage of directory structure, filenames, and file types, can add considerable flexibility and convenience to your use of the system.

2.3.2 Logical Name Translation

You can specify a logical name to stand for either a part or whole file specification. For example, you can define the logical name INPUT to have an equivalence consisting of the file specification BIGVOLUME:[USERFILES]NAMES.TXT. Subsequently, you can use INPUT wherever you would normally use the file specification. See the *Tool Kit User's Guide* for more information on logical names.

Note that PRO/Tool Kit commands translate logical names only when a device, directory, or file specification is expected.

2.4 FOREGROUND AND BACKGROUND PROCESSING

Once you initiate a command, you cannot type another command until the first command has completed execution. In this case, you initiated the command in foreground mode. Typing <CTRL/C> or <INTERRUPT/DO> halts the execution of a command or program in foreground mode.

You can initiate commands or programs in background mode through the use of the SPAWN command. Once you have initiated such a command or program, you can issue other commands or run programs at your terminal; you need not wait for the background command or program to complete execution. In order to halt the execution of a command or program running in background mode, you must use the ABORT commmand.

2.4.1 <CTRL/C>

<CTRL/C> or the <INTERRUPT/DO> key sequence aborts the command you just typed at the terminal. <CTRL/C> cancels command entry or processing, and interrupts command or program execution, returning control to DCL. However, <CTRL/C> will not abort a commmand or program that you initiated with the SPAWN command. You must use ABORT to halt the execution of such a command or program.

For example, if you just typed DIRECTORY and realized you did not want to execute the DIRECTORY command, use of <CTRL/C> will abort the action of the DIRECTORY command.

2.4.2 ABORT

This command halts execution of commands and programs running in background mode.

NOTE

You must abort the task name, not the name of the command that invoked its execution. If you use SPAWN to initiate a DCL command in background mode, that DCL command runs as a task with a task name of the form xxxTn, where xxx are the first three characters of the DCL command, and Tn is the name of the terminal from which the command was issued.

To determine the name of the task running in background mode, use the SHOW TASKS command.

NOTE

You should never initiate a command or program that attaches the monitor in background mode, because you cannot abort it.

CHAPTER 3

DCL COMMANDS

This chapter lists the PRO/Tool Kit DCL commands alphabetically and provides description, syntax, options, and command prompts for each command. Note that in the syntax description of each command, the portion of the command shown in **bold** indicates the minimally acceptable abbreviation.

Some of the PRO/Tool Kit DCL commands can be used for network operations, or have qualifiers that enable network operations. In order to use these commands and qualifiers for network operations, you must first install the PRO/DECnet Tool Kit. When you install the PRO/DECnet Tool Kit, the DCL extensions are also automatically installed. See the *PRO/DECnet Installation Guide* for details.

3.1 ABORT

ABORT forces an orderly end to a running task or to the action of a specfic command. Nonprivileged users can abort any task running on TI:. Privileged users can abort any task.

Syntax

ABORT [/COMMAND] [/qualifier] commandname

ABORT/TASK [/qualifier] taskname

qualifier

/COMMAND
/TASK
/TERMINAL:ttnnn:

commandname

Specifies the command whose effect you want to cancel. This parameter can be used only when the /TASK qualifier is not present. You must specify at least the first three characters of the command verb.

taskname

Specifies the name of the task you want to abort. This parameter requires the presence of the /TASK qualifier. If you use the /TASK qualifier and do not specify a task name, you will get the error message "Illegal task name."

Prompts

Taskname? taskname

Command Qualifiers

/COMMAND

Specifies that you want to abort a command. This is the default qualifier and need not be specified.

/TASK

Specifies that you want to abort a task by name.

/TERMINAL:ttnnn:

Specifies that a task from some terminal other than your own be aborted. This is a privileged qualifier.

Example

```
$ SPAWN LINK filespec
$ ABORT LINK
```

In this example, the user issued a LINK command in the background by using the SPAWN command. Rather than waiting for the command to complete, the user entered the ABORT command to terminate the command.

```
$SPAWN LINK filespec $ABORT /TASK LINT1
```

In this example, the user issued a LINK command in the background by using the SPAWN command. Rather than waiting for the command to complete, the user entered the ABORT command to terminate the task build by specifying the task name that the SPAWN LINK command executed.

3.2 APPEND

APPEND attaches records from one or more sequential files to the end of an existing sequential file. For details regarding sequential files, see the manual *PRO/RMS-11: An Introduction*, in the Tool Kit documentation set.

Syntax

APPEND[/qualifier[s]] infile[,s] outfile

qualifier

Can be one or more of the following:

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/NOWARNINGS
/SHARED

With PRO/DECnet DCL extensions, the following qualifiers are also available:

/DATA_TYPE
/LOG
/MACY11
/PRINT
/TRANSFER_MODE
/SUBMIT
/PROTECTION
/OWNER

infile

Specifies the file or files to be appended to the output file.

outfile

Specifies the file to which the input files are appended.

The output file must be an existing sequential file. The output file has the same version number after the APPEND command is executed as it had before the command was issued. The input files appear at the end of the output file in the order they were specified.

No wildcards are permitted in output file specifications.

There are no qualifiers for the output filespec.

Although the output file must exist, it may be an empty file.

Prompts

```
File(s)? infile[,s]
To: outfile
```

Qualifiers

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the APPEND command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the APPEND command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the APPEND command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the APPEND command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the APPEND command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the APPEND command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/[NO]WARNINGS

The /NOWARNINGS qualifier specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

/SHARED

The /SHARED qualifier specifies that you want other users to be able to access the file while you are performing the operation.

Additional Qualifiers Available with PRO/DECnet

Command Qualifiers

/LOG

The /LOG qualifier specifies that the names of files appended and their sizes in blocks are to be displayed as the operation is performed.

/PRINT

The /PRINT qualifier specifies that the file will be queued to the line printer on the remote node for printing. The file will not be deleted by this operation. The file's format must be compatible with the remote system.

/SUBMIT

The /SUBMIT qualifier specifies that the output ifle is a temporary file and will be submitted to the remote batch processor facility when copied. The file will be deleted after it is processed at the remote node.

File Qualifiers

/DATA_TYPE

The /DATA_TYPE qualifier specifies how the data in the file is interpreted. /DATA_TYPE with APPEND requires either the ASCII or IMAGE argument. The default is IMAGE. /DATA_TYPE must be entered after the file specification in the command line.

The ASCII argument transfers files in ASCII record mode and translates the records into an appropriate format for the remote system. You only need to use this argument when transferring text files to remote systems with file systems other than FILES-11 or RMS. File transfers to VAX/VMS and

PDP-11/RSX systems are more efficiently performed using the default, IMAGE.

The IMAGE argument transfers files with their current format and attributes. If a remote system cannot interpret the file format, an error message will be displayed. If the file is an ASCII text file, you can recover by retrying with the ASCII argument.

/MACY11

The /MACY11 qualifier specifies that the input or output file on the remote TOPS-10 or TOPS-20 system is in MACY11 format. This informs the remote DECsystem to handle the file appropriately when copying to or from the P/OS system. /MACY11 must be entered after the file specification in the command line.

/TRANSFER_MODE

The /TRANSFER_MODE qualifier specifies the method of packaging file data during a file transfer. This qualifier requires either the BLOCK or RECORD argument with the APPEND command. RECORD is the default. /TRANSFER_MODE must be entered after the file specification in the command line.

The BLOCK argument transfers files in block mode, using blocks of 512-byte sizes. This is more efficient than using record mode, but should only be used with systems that support FILES-11 or RMS-11, such as VAX/VMS, PDP-11/RSX, or PDP-11/RSTS/E RMS FAL.

The RECORD argument transfers a file one record at a time. A record is a logical unit of data in a file.

/OWNER

This qualifier specifies the UIC of the file. A UIC establishes the owner of a file. See the *Tool Kit User's Guide* for a description of UIC.

/PROTECTION:code

This qualifier specifies the protection code applied to the output file. See the description of the SET PROTECTION command for details on the code.

3.3 ASSIGN

This command creates a logical name and assigns an equivalence string to the specified logical name. An equivalence name can be a device name, another logical name, a file specification, or any other string.

To specify the logical name table where you want to enter a logical name, use /USER, /SESSION, or /SYSTEM. If you do not explicitly specify a logical name table, the default is /SESSION.

Syntax

ASSIGN[/qualifier[s]] equiv_name logical_name

qualifier

Can be one or more of the following:

/USER
/SESSION
/SYSTEM
/FINAL
/TRANSLATION_ATTRIBUTES

equiv_name

Defines the equivalence name to be associated with the logical name in the specified logical name table. The equivalence name string can contain from 1 to 255 characters. If the equivalence name contains quotation marks, enclose the string in quotation marks and use two sets of quotation marks ("") in the places where you want a quotation mark to appear.

Note that if you enclose an equivalence name string in quotation marks the case of alphabetic characters is preserved.

logical_name

Specifies the logical name string. The logical name string can contain from 1 to 255 characters. If the logical name contains quotation marks, enclose the string in quotation marks and use two sets of quotation marks ("") in the places where you want a quotation mark to appear.

Note that if you enclose a logical name string in quotation marks the case of alphabetic characters is preserved.

Prompts

```
Equivalence Name? equiv_name
Logical Name?  logical_name
```

Qualifiers

/USER

Places the logical name in the user logical name table. Logicals created in the user logical name table are deleted when you exit any application (including DCL).

/SESSION

Places the logical name in the session logical name table. Session logical names are deleted when you logout of the system. If you do not explicitly specify a logical name table, the default is /SESSION.

/SYSTEM

Places the logical name in the system logical name table. All system users can access the logical name.

System logical names are never deleted unless an explicit deassign logical operation is performed.

/FINAL

Indicates that the equivalence name string should not be translated iteratively; logical name translation should terminate with the current equivalence string. The /FINAL qualifier is synonymous with:

```
/TRANSLATION_ATTRIBUTES:TERMINAL
```

/TRANSLATION_ATTRIBUTES:[(keyword[,...])]

Specifies one or more attributes to modify a logical name string. The /TRANSLATION_ATTRIBUTES qualifer is positional and must be specified before the equivalence name.

You can specify the following keywords for translation attributes:

CONCEALED Indicates that the logical name is a concealed device name. If you specified the CONCEALED attribute, the logical name must be a device name of the form ddnnn: and the equivalence name must be of the form _ddnnn:[directory.]. The device name in the equivalence name must be present. The [directory.] portion of the equivalence name is optional.

TERMINAL Indicates that the equivalence name string should not be translated iteratively; logical name translation should terminate with the current equivalence string.

If you specify only one keyword, you may omit the parentheses. Only the attributes you specify are set.

Examples

```
$ ASSIGN DW001:[CHARLES] CHARLIE:
$ PRINT CHARLIE:TEST.DAT
```

The ASSIGN command associates the logical name CHARLIE with the device and directory DW001:[CHARLES]. Susequent references to the logical name CHARLIE: result in the correspondence between the logical name CHARLIE: and the disk and directory specified. The PRINT command queues a copy of the file DW001:[CHARLES]TEST.DAT to the default print queue.

```
$ ASSIGN DW001:[CHARLES] CHARLIE: /USER
```

The ASSIGN command associates the logical name CHARLIE: with the directory name [CHARLES] on the device DW001: and makes the assignment in the user logical name table.

```
$ AS/TRANSLATION_ATTRIBUTES:CONCEALED _DW002:[CHARLES.] -
$-PD001:
$ DIRECTORY PD001:[USERFILES]

    Directory PD1:[USERFILES]
    28-JUN-85 11:20

    TEST.DAT          4.        27-JUN-85 17:21

    Total of 4./4. blocks in 1. files
```

The ASSIGN command associates the logical name PD001: with the directory name [CHARLES] on the device DW002. Subsequent references to the device PD001: result in the device and directory specified as being used as the MFD for all file operations.

```
$ ASSIGN/TRANSLATION_ATTRIBUTES:TERMINAL XX001: TEMP:
```

The ASSIGN command associates the logical name TEMP: with the device XX001. Subsequent references to the logical TEMP: will result in the device XX001: being used without any futher translation attempts on the equivalence name XX001:.

3.4 ASSIGN/TASK

ASSIGN/TASK reassigns the logical unit numbers (LUNs) of an installed task from one physical device to another. This reassignment overrides the static LUN assignments in the task's disk image file. ASSIGN/TASK cannot change the LUNs of an active task.

Syntax

ASSIGN/**TA**SK:taskname ddnnn: lun

taskname

Identifies the installed task whose LUN you want to reassign.

ddnnn:

Specifies the device to which you want the LUN reassigned. This can be a physical device or logical device name.

lun

Specifies which LUN you want to reassign to the new device.

Prompts

```
Taskname?   taskname
Device?   ddnnn:
Logical unit?   lun
```

3.5 BASIC

BASIC invokes the PRO/Tool Kit BASIC-PLUS-2 compiler to begin a BASIC session. (Refer to the *BASIC on RSX-11M/M-PLUS Systems* manual and the *BASIC Reference Manual* for additional information.)

Syntax

BASIC

Prompts

None

3.6 CANCEL

CANCEL eliminates entries from the clock queue. Either the RUN$ directive or the time-based forms of the RUN command place entries in the clock queue. CANCEL only affects pending entries in the clock queue and does not affect a task that is currently executing.

Syntax

CANCEL taskname

taskname

Is the name of the inactive, installed task entry that is to be deleted from the clock queue.

Prompts

Taskname? taskname

3.7 CLEAR

This command clears the screen and optionally resets the terminal subsystem to an initial state.

Syntax

CLEAR [/qualifier]

qualifier

/**RE**SET

Prompts

None

Qualifier

/RESET

Emits a reset-to-initial-state (RIS) sequence, which initializes the Terminal Subsystem to default states and clears the screen. The RIS sequence is normally invoked whenever you power-up (boot) the Professional or reset the PRO/Communications terminal emulator. For further information, see the description of the RIS sequence in the *Terminal Subsystem Manual*.

3.8 COBOL

COBOL invokes the PRO/Tool Kit COBOL-81 compiler to compile COBOL language source files.

NOTE

Please refer to the language documentation for additional information.

Syntax

COBOL[/qualifier[s]] filespec

qualifier

Can be one or more of the following:

```
/[NO]ANSI_FORMAT
/[NO]CHECK[:arg]
     [NO]BOUNDS
     [NO]PERFORM
     ALL
     NONE
/CODE:[NO]CIS
/[NO]CROSS_REFERENCE
/[NO]DEBUG
/[NO]DIAGNOSTICS[:filespec]
/[NO]LIST[:filespec]
/NAMES:xx
/[NO]OBJECT[:filespec]
/[NO]OVERLAY_DESCRIPTION
/[NO]SHOW:[NO]MAP
/[NO]SKELETON
/[NO]SUBPROGRAM
/TEMPORARY:device
/[NO]TRUNCATE
/[NO]WARNINGS:[NO]INFORMATIONAL
```

filespec

Specifies the source file to be input to the COBOL-81 compiler.

File input to the COBOL-81 compiler must contain COBOL source code. File specifications must have filenames. The default file type is .CBL.

Prompts

```
File? filespec
```

Qualifiers

/[NO]ANSI_FORMAT

The /ANSI_FORMAT qualifier specifies that the source file is in conventional (or ANSI) format. Conventional format has 80-character lines with Area A beginning in character position 8.

The /NOANSI_FORMAT qualifier specifies that the source file is in terminal format, a DIGITAL-specified format that permits variable length lines with Area A beginning in character position 1.

The default is /NOANSI_FORMAT.

/[NO]CHECK[:arg]

```
[NO]BOUNDS
[NO]PERFORM
ALL
NONE
```

The /CHECK and /CHECK:ALL qualifiers are equivalent--both to each other, and also to /CHECK:BOUNDS and /CHECK:PERFORM in combination. They add object code that checks the ranges of subscripts, indexes, and nested PERFORM statements at run time.

The /CHECK:BOUNDS qualifier compares subscript and index ranges at run time against the ranges defined by corresponding OCCURS clauses. If any range is exceeded during program execution, COBOL-81 issues an error message.

The /CHECK:PERFORM qualifier determines whether or not your program's PERFORM statements are nested properly (if nested at all). If COBOL-81 detects improper nesting during program execution, it issues an error message.

The /NOCHECK and /CHECK:NONE qualifiers are equivalent--both to each other, and also to /CHECK:NOBOUNDS and /CHECK:NOPERFORM in combination. They suppress all range checking.

The /CHECK:NOBOUNDS qualifier suppresses range checking only for subscripts and indexes.

The /CHECK:NOPERFORM qualifier suppresses range checking only for nested PERFORM statements.

The qualifiers that suppress range checking reduce task size and improve program performance.

The default is /CHECK (/CHECK:ALL).

/CODE:[NO]CIS

The /CODE:CIS qualifier tells the compiler to use CIS (Commercial Instruction Set) in the object code it produces.

NOTE

The Professional 300 Series computers do not have the Commercial Instruction Set.

The /CODE:NOCIS qualifier tells the compiler not to use CIS.

These qualifiers override the default for your system. They are used to develop programs that will execute on a different system than the one used for program development. For example, if your system has CIS, you use /CODE:NOCIS to compile a program that will run on a system without CIS.

/[NO]CROSS_REFERENCE

The /CROSS_REFERENCE qualifier causes the compiler to produce a list file and to add two cross-reference tables to the end of the list file: one for data-names and one for procedure-names. In each table, the names you use in your COBOL program are listed alphabetically. Opposite each name is a list of every line number in which that name occurs. A "D" after a number indicates the line in which you define the name. An asterisk (*) after a line number indicates a destructive reference, such as a value assignment to a data-name.

The /NOCROSS_REFERENCE qualifier suppresses production of the cross-reference tables.

The default is /NOCROSS_REFERENCE.

/[NO]DEBUG

The /DEBUG qualifier indicates that you plan to include the COBOL-81 Symbolic Debugger in your task image. To support the Symbolic Debugger, the compiler generates symbol information in the object module for all data-names and procedure-names in your program.

If you include the Symbolic Debugger in your program, you must also use the /DEBUG qualifier to the LINK command.

The /NODEBUG qualifier suppresses generation of symbol information in the object module.

The default is /NODEBUG.

/[NO]DIAGNOSTICS[:filespec]

The /DIAGNOSTICS qualifier creates a diagnostics file that contains the compiler diagnostic summary. If you do not append a file specification, the diagnostics file has the same filename as your source file, and the file type .DIA.

The /NODIAGNOSTICS qualifier suppresses the creation of a diagnostics file.

The default is /NODIAGNOSTICS.

/[NO]LIST[:filespec]

The /LIST qualifier produces a list file which contains both the complete source code and any diagnostic messages. If you do not append a file specification, the list file has the same filename as the source file and has file type .LST.

The /NOLIST qualifier suppresses production of a list file.

The default is /NOLIST.

/NAMES:xx

The /NAMES:xx qualifier tells the compiler to use the two alphanumeric characters you specify as the PSECT kernel for your program. You use this qualifier to ensure unique identification for PSECT kernels when your task image uses both subprograms and segmentation.

/[NO]OBJECT[:filespec]

The /OBJECT qualifier produces an object module. If you do not append :filespec, the object module has the same filename as the source and the file type .OBJ.

The /NOOBJECT qualifier suppresses production of an object module and its associated skeleton ODL file. (See the description of /[NO]SKELETON.)

The default is /OBJECT.

/[NO]OVERLAY_DESCRIPTION

The /OVERLAY_DESCRIPTION qualifier produces two files--an indirect command file and an Overlay Descriptor Language file.

The indirect command file has the same filename as the source and the file type .CMD. It can be input to the Task Builder to create the task image.

The Overlay Descriptor Language file has the same filename as the source and the file type .ODL. It contains pointers to certain support routines that must be included in your task image (for example, system support for file input/output).

If you plan to use the LINK/C81 command to task-build your program, do not use the /OVERLAY_DESCRIPTION qualifier (refer to Section 3.35 for a description of the LINK/C81 command). LINK/C81 also creates a .CMD and an .ODL file, and cannot reference the .CMD and .ODL files created during program compilation.

The /NOOVERLAY_DESCRIPTION qualifier suppresses production of .CMD and .ODL files during program compilation.

The default is /NOOVERLAY_DESCRIPTION.

/[NO]SHOW:[NO]MAP

The /SHOW and /SHOW:MAP qualifiers are equivalent. They cause the compiler to produce a list file and to append two offset maps to the list file. One offset map refers to the Data Division and one to the Procedure Division. The compiler provides these maps for use with ODT (the On-Line Debugging Tool). Consult the *IAS/RSX-11 ODT Reference Manual* for more information.

The /NOSHOW and /SHOW:NOMAP qualifiers are equivalent. They suppress production of the offset maps.

The default is /NOSHOW (/SHOW:NOMAP).

/[NO]SKELETON

The /SKELETON qualifier produces a skeleton ODL file, which specifies the overlay structure for the object module. This file has the same filename as the source and the file type .SKL.

You cannot use the /SKELETON qualifier to override the /NOOBJECT qualifier. That is, an .SKL file cannot be produced when the .OBJ file is suppressed. The /NOSKELETON qualifier suppresses production of the .SKL file.

You cannot use the /NOSKELETON qualifier if you plan to task-build your program with the LINK/C81 command. LINK/C81 assumes the presence of the .SKL file. Although the .SKL file is not strictly necessary for all programs, DIGITAL does not recommend that you suppress its production unless you are familiar with task-building alternatives to LINK/C81.

The default is /SKELETON.

/[NO]SUBPROGRAM

The /SUBPROGRAM qualifier tells the compiler to treat the source file as a subprogram. You should use this qualifier only if the subprogram does not use parameters from the main program; that is, if it does not contain the Procedure Division USING header.

The /NOSUBPROGRAM qualifier tells the compiler to treat the source as a main program.

The default is /NOSUBPROGRAM.

/TEMPORARY:device

The /TEMPORARY qualifier tells the compiler to store the temporary work files it uses during program compilation on the device you specify. Since the default device is the system disk (SY:), this qualifier is useful if there is little system disk space available.

/[NO]TRUNCATE

The /TRUNCATE qualifier tells the compiler to perform decimal truncation on the values of COMP data items. With decimal truncation, the maximum value the data item can contain depends on the item's PICTURE character-string.

The /NOTRUNCATE qualifier tells the compiler to perform binary truncation on the values of COMP data items. With binary truncation, the maximum value a COMP item can contain depends on its storage allocation.

The default is /NOTRUNCATE.

/[NO]WARNINGS:[NO]INFORMATIONAL

The /WARNINGS and /WARNINGS:INFORMATIONAL qualifiers are equivalent. They cause the compiler to issue informational, warning, and fatal diagnostics.

The /NOWARNINGS and /WARNINGS:NOINFORMATIONAL qualifiers are equivalent. They suppress production of informational, warning, and fatal diagnostics.

The default is /WARNINGS (/WARNINGS:INFORMATIONAL).

3.9 CONTINUE

CONTINUE resumes execution of a previously-suspended task. Nonprivileged users can continue tasks initiated from their own terminals. Privileged users can continue any suspended task.

Syntax

CONTINUE taskname

taskname

Is the name of the task to be restarted.

Prompts

Taskname? taskname

3.10 CONVERT

CONVERT invokes the RMSCNV utility, which moves records from one file to another. CONVERT reads records from an input file and writes them to an output file. This conversion depends on both the organization (sequential, relative, or indexed) of the files, and qualifiers included with the CONVERT command. Refer to the *RSX-11M/M-PLUS RMS-11 Utilities Manual* for further information on the RMSCNV utility.

Syntax

CONVERT[/qualifier[s]] input_file output_file

qualifier

Can be one or more of the following:

/[NO]APPEND
/[NO]FIXED_CONTROL
/[NO]IDENTIFICATION
/INDEXED
/KEY[:n]
/[NO]LOG_FILE[:filespec]
/[NO]MASS_INSERT
/MERGE
/[NO]PAD[:[#]arg]
/RELATIVE
/[NO]REPLACE
/SEQUENTIAL
/[NO]TRUNCATE

input_file

Specifies the name of the file that is the source of records to the output file. CONVERT reads records sequentially, regardless of the organization of the input file. CONVERT accepts no wildcards.

output_file

Specifies the name of the file to receive records from the input file. The default file organization for the output file is sequential. If the output file is not sequential, use the /RELATIVE or /INDEXED qualifier to indicate the organization. If the output file is to be sequential, CONVERT can create the file; it need not exist prior to your entering the CONVERT command. CONVERT cannot create indexed or relative files; these must have been created prior to your entering the CONVERT command. Refer to the

RSX-11M/M-PLUS RMS-11 User's Guide for additional information. The CONVERT command does not accept wildcards.

Prompts

```
Input file ? input_file
Output file? output_file
```

Qualifiers

/[NO]APPEND

Specifies that you want RMSCNV to append records to the end of an existing sequential file. If the output file is not sequential, RMSCNV ignores the qualifier. You cannot use both /APPEND and /REPLACE in the same command line.

The default is /NOAPPEND, but the action taken depends on the presence of the /REPLACE qualifier in the command line. If you specified /REPLACE, RMSCNV performs the replace operation. If you did not specify /REPLACE, RMSCNV creates the next higher version of the file.

This qualifier is the equivalent of the /AP switch in an RMSCNV command line.

/[NO]FIXED_CONTROL

The /FIXED_CONTROL qualifier directs RMSCNV to handle variable-with-fixed-control (VFC) format records in either the input file or the output file. If the fixed-control area of the input file and the output file are the same size, RMSCNV performs a straightforward copy. See the RMS-11 documentation supplied with your system for information on how RMSCNV handles other combinations. If you include this qualifier and neither file specifies VFC records, RMSCNV terminates.

This qualifier is the equivalent of the /WF switch in an RMSCNV command line.

The default is /NOFIXED_CONTROL. This means that if one of the named files contains VFC records, the fixed-control area of each record is ignored. That is, if the input file includes VFC records and the output file does not, only the variable portion of each record is written to the output file. If the output file includes VFC records and the input file does not, data is written only into the variable portion of each output record.

/[NO]IDENTIFICATION

Requests that RMSCNV print its current version number and patch level on your terminal. See the RMSCNV documentation for more information.

This qualifier is the equivalent of the /ID switch in an RMSCNV command line.

The default is /NOIDENTIFICATION.

/INDEXED

The /INDEXED qualifier informs RMSCNV that the output is of indexed organization. Regardless of their organization, all input files are read sequentially.

RMSCNV reads each record from the input file, then applies the output file's record format, that is, key placement within the record, to the data. If you do not specify the /KEY qualifier with a value, the key of reference is the primary key; otherwise, it is the key you specify.

This qualifier is the equivalent of the /FO:IDX switch in an RMSCNV command line.

/KEY[:n]

Indicates the key that establishes the order in which records are read sequentially from an indexed input file and written to the output file. n can be from 0 through 9. The default is n=0 and indicates the primary key; n=1 is the first alternate key; n=9 is the ninth alternate key.

This qualifier is the equivalent of the /KR qualifier in an RMSCNV command line.

/[NO]LOG_FILE[:filespec]

Directs RMSCNV to summarize processing in a log. If you do not include a filespec, the log appears on your terminal. If you name a file, that file will be created and written to by RMSCNV. The log includes the following elements:

- The command string in RMSCNV format. This will not be the same as DCL format.
- Copies of all error messages produced during execution.

- An indication of any duplicate-key problems. If the log is appearing on the terminal, the indication consists only of the following message:

  SOME DUPLICATE RECORDS NOT WRITTEN

  If the log is being written to a file, RMSCNV supplies the indicator DUP RCD= followed by the first 72 characters of the record that could not be written.

This qualifier is the equivalent of the /SL qualifier in an RMSCNV command line.

The default is /NOLOG_FILE, meaning that only normal error messages appear on your terminal.

/[NO]MASS_INSERT

Directs RMSCNV to activate the RMS-11 mass insert I/O technique and then use sequential put operations to insert records into the output file.

This is the equivalent of the /MA qualifier in an RMSCNV command line.

The default for nonsequential file organization is /NOMASS_INSERT.

/MERGE

Directs RMSCNV to copy records from the input file into the output file. Both files must be of the same organization.

/[NO]PAD[:[#]arg]

Directs RMSCNV to pad records read from the input file to the output file's record length before writing them to the file. If you specify the qualifier without an argument, the pad character is null. If you do not include the number sign (#), the argument can be any printing ASCII character except: the number sign, question mark (?), or commercial-at sign (@). If you include the number sign, n can be an octal number from 0 through 377, representing the full ASCII character set. This enables you to use the three excluded characters as pad characters by specifying 43 for #, 77 for ?, and 100 for @.

Use this qualifier only when the output file specifies fixed-length records.

This qualifier is the equivalent of the /PD qualifier in an RMSCNV command line.

/RELATIVE

The /RELATIVE qualifier informs RMSCNV that the output file is of relative organization. Regardless of their organization, all input files are read sequentially.

The /RELATIVE qualifier is the equivalent of the /FO:REL qualifier in an RMSCNV command line.

RMSCNV reads records from the input file and writes them into successive record cells of the output file, beginning with cell 1. If RMSCNV encounters a cell containing a record, it terminates with an error message. All records written to that point are in the output file. You should examine the two files to determine how far the processing went.

/[NO]REPLACE

The /REPLACE qualifier directs RMSCNV to supersede an existing sequential file. RMSCNV replaces a file in the output account with the same filename, type, and version number. You cannot use /REPLACE and /APPEND in the same command line.

The /REPLACE qualifier is the equivalent of /SU in an RMSCNV command line.

The default is /NOREPLACE, but the default action depends on whether the command line includes an /APPEND or not. If you specify /APPEND, RMSCNV performs the append operation. If you do not specify /APPEND, RMSCNV creates the next higher version of the file.

/SEQUENTIAL

The /SEQUENTIAL qualifier informs RMSCNV that the output file is of sequential organization. Regardless of their organization, all input files are read sequentially. The default output file organization for RMSCNV is sequential.

This qualifier is the equivalent of the /FO:SEQ qualifier in an RMSCNV command line.

If you do not specify either the /APPEND or /REPLACE qualifier, and the output file is sequential, the output file need not exist before you issue the CONVERT command.

If the output file does not exist, RMSCNV creates an output file with the record attributes of the input file. RMSCNV then reads records from the input file and writes them sequentially into the new output file.

If the output file already exists, and the command does not include the /APPEND qualifier, RMSCNV creates the next higher version of the file. RMSCNV then reads records from the input file and writes them sequentially into the new version of the output file.

If the output file exists and you specify the /REPLACE qualifier, RMSCNV reads records from the input file and writes them sequentially into the output file, starting with the beginning of the file.

If the output file exists and you specify the /APPEND qualifier, RMSCNV reads records from the input file and writes them into the output file, starting with the record position following the last record already in the file.

/[NO]TRUNCATE

Directs RMSCNV to truncate records read from the input file to the output file's record length before writing them into the output file. The trailing bytes of the record are truncated.

The default is /NOTRUNCATE. If you do not specify /TRUNCATE and the input records are too long, RMSCNV terminates.

3.11 COPY

This command creates a sequential file copy of one or more sequential files, or of records with either indexed or relative file organization.

Syntax

COPY[/qualifier[s]] input_filespec[,s] output_filespec

qualifier

Can be one or more of the following:

/[NO]CONTIGUOUS
/REPLACE
/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/[NO]WARNINGS
/PRESERVE_DATE
/ALLOCATION:n[.]
/NONEW_VERSION
/OVERLAY
/SHARED
/OWN

With PRO/DECnet DCL extensions, the following qualifiers are also available:

/CONCATENATE
/DATA_TYPE
/LOG
/MACY11
/NEW_VERSION
/NOSPAN
/PRINT
/SUBMIT
/TRANSFER_MODE
/OWNER
/PROTECTION

input_filespec

Specifies the input file or files to be copied.

You must have READ access to a file to copy it.

Multiple filespecs, separated by commas, are accepted. If you specify multiple input files, they will be concatenated in a single output file in the order that you specify them.

output_filespec

Specifies a single output file to which the input file or files is copied.

You can change the name, type, and version number of the file when you enter this parameter. Wildcards in the place of the name and the type leave the name and type unchanged. If you use a wildcard in either of these fields, you must use a wildcard in both.

COPY always creates the output file. For example, if you type:

```
COPY FILE1.LIS FILE2.LIS
```

and FILE2 already exists, COPY will create a new version of the file one higher than the existing version. If FILE2 does not already exist, COPY will create a file with the name FILE2 and extension .LIS. If you specify a version number for the output file field, then a file of that version number is created. If such a file already exists, the operation fails.

Wildcards are acceptable for output files if the destination is another directory. If you have multiple input files and use wildcards for the output file, you create multiple output files, each with the name and type of the corresponding input file.

You can send copies to devices as well as to directories.

You can also use the COPY command to create multiple copies of the same file with the same or different names.

Prompts

```
From? input_filespec[,s]
To? output_filespec
```

Qualifiers

/[NO]CONTIGUOUS

Specifies that the output file must be contiguous. If this qualifier is not used, then only files that are already contiguous remain contiguous when copied.

The default is /NOCONTIGUOUS.

/REPLACE

If the output file has the same name, type, and version number as an already existing file at the destination, the first file is deleted and the file you have sent replaces it. The name, type and version number stay as they were.

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the COPY command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the COPY command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the COPY command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the COPY command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the COPY command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you don't want COPY to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/[NO]WARNINGS

The /[NO]WARNINGS qualifier specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

/ALLOCATION:n[.]

The /ALLOCATION:n[.] qualifier specifies that n blocks of contiguous space be allocated for the new copy of the file. Unless you append the optional decimal point, the value n is interpreted as an octal value.

/NONEW_VERSION

The /NONEW_VERSION qualifier prevents the version number from being automatically incremented when a file is copied. You must specify this qualifier when you copy all versions of a particular filename and type.

/OVERLAY

The /OVERLAY qualifier causes the contents of the input file or files to be copied into the output file. Whatever is currently in the output file is destroyed, although the file ID remains unchanged. The output file must exist before you issue the command.

/PRESERVE_DATE

The /PRESERVE_DATE qualifier specifies that you want the output file to have the same creation date as the input file. By default, today's date is the creation date of the output file.

/SHARED

The /SHARED qualifier specifies that you want other users to be able to access the file while you are copying it.

/OWN

The /OWN qualifier changes the ownership of the file being copied to the destination directory. After execution, both directories own their respective copies of the file. If you do not specify /OWN, the original UIC owns both copies of the file.

Additional Qualifiers Available with PRO/DECnet

Command Qualifiers

/CONCATENATE

The /CONCATENATE qualifier specifies that the output file should be a single file that contains all of the input files in the order specified. If wildcards are used in the input file specification, the files will be concatenated in random order.

/LOG

The /LOG qualifier specifies that the names of files copied and their sizes in blocks are to be displayed as the operation is performed.

/PRINT

The /PRINT qualifier specifies that the file will be queued to the line printer on the remote node for printing. The file will not be deleted by this operation. The file's format must be compatible with the remote system.

/SUBMIT

The /SUBMIT qualifier specifies that the output file is a temporary file and will be submitted to the remote batch processor facility when copied. The file will be deleted after it is processed at the remote node.

File Qualifiers

/DATA_TYPE

The /DATA_TYPE qualifier specifies how the data in the file is interpreted. With the COPY command, the DATA_TYPE qualifier takes either the ASCII or IMAGE argument. The default is IMAGE. /DATA_TYPE must appear after the file specification in the command line.

The ASCII argument transfers files in ASCII record mode and translates the records into an appropriate format for the remote system. You only need to use this argument when transferring text files to remote systems with file systems other than RMS. File transfers to VAX/VMS and PDP-11/RSX systems, which use RMS, are more efficiently performed using the default, IMAGE.

The IMAGE argument transfers files with their current format and attributes. If a remote system cannot interpret that format, an error message will be displayed. If the file is an ASCII text file, you can recover by retrying with the ASCII argument.

/MACY11

The /MACY11 qualifier specifies that the input or output file on the remote TOPS-10 or TOPS-20 system is in MACY11 format. This informs the remote DECsystem to handle the file appropriately when copying to or from the P/OS system. /MACY11 must appear after the file specification in the command line.

/NEW_VERSION

The /NEW_VERSION qualifier specifies that the output files be created with a version number higher than any existing files of the same name in the directory. /NEW_VERSION must appear after the file specification in the command line.

/NOSPAN

The /NOSPAN qualifier specifies that the output file be created with records that do not span block boundaries. Normally, only files that already have records that do not span block boundaries remain as such when copied. /NOSPAN must appear after the file specification in the command line.

/TRANSFER_MODE

The /TRANSFER_MODE qualifier specifies the method of packaging file data during a file transfer. For the COPY command, the TRANSFER_MODE qualifer takes the argument AUTOMATIC, BLOCK, or RECORD. AUTOMATIC is the default. /TRANSFER_MODE must appear after the file specification in the command line.

The AUTOMATIC argument causes DCL to select either BLOCK or RECORD after checking the file capabilities at the remote node. If you have problems transferring files with the AUTOMATIC argument, try specifying either BLOCK or RECORD. The BLOCK argument transfers files in block mode, using blocks of 512-byte sizes. This is more efficient than using record mode, but should only be used with systems that support RMS-11, such as VAX/VMS, PDP-11/RSX, or PDP-11/RSTS/E RMS FAL.

The RECORD argument transfers a file one record at a time. A record is a logical unit of data in a file.

/OWNER:[UIC]

The /OWNER qualifier specifies the UIC of the output file. A UIC establishes the owner of a file. See the Tool Kit User's Guide for a description of UIC.

/PROTECTION:code

This qualifier specifies the protection code applied to the output file. See the description of the SET PROTECTION command for details on the code.

3.12 CREATE

CREATE creates a sequential file and enables you to type text directly into the file from your terminal without using an editor.

Syntax

CREATE filespec

filespec

Specifies the name of the file to be created.

As soon as the command is entered, the cursor moves down a line. The file is open for input. Any text you type goes into the file. When you have finished entering text, type a CTRL/Z to close the file.

If you want to create an empty sequential file, simply enter the CTRL/Z first.

If you use CTRL/U when creating a sequential file, the text on the line is eliminated, but not the line itself. In other words, CTRL/U leaves a blank line behind when it deletes a line. CTRL/U, CTRL/R, and the DELETE key are the only editing facilities available to you when creating sequential files at the terminal.

Prompts

File? filespec

With PRO/DECnet DCL extensions, the following qualifiers are available:

/CONTIGUOUS
/LOG
/MACY11
/NEW_VERSION
/NOSPAN
/[NO]WARNINGS
/PRINT
/REPLACE
/PROTECTION
/OWNER
/SUBMIT

With PRO/DECnet, the CREATE command creates a file from records you type. In other words, when you press <DO> after entering the CREATE command, all subsequent keystrokes are entered into a file that you named with the CREATE command. To close and store the file, press <CTRL/Z>.

Command Qualifiers

/LOG

The /LOG qualifier specifies that the names of files created and their sizes in blocks are to be displayed as the operation is performed.

/NOWARNINGS

The /NOWARNINGS qualifier specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

/PRINT

The /PRINT qualifier specifies that the file will be queued to the line printer on the remote node for printing. The file will not be deleted by this operation. The file's format must be compatible with the remote system.

/SUBMIT

The /SUBMIT qualifier specifies that the output file is a temporary file and will be submitted to the remote batch processor facility when copied. The file will be deleted after it is processed at the remote node.

File Qualifiers

/CONTIGUOUS

The CONTIGUOUS qualifier specifies that the output file must be contiguous. /CONTIGUOUS must appear after the file specification in the command line.

/MACY11

The /MACY11 qualifier specifies that the input or output file on the remote TOPS-10 or TOPS-20 system is in MACY11 format. This informs the remote DECsystem to handle the file appropriately when copying to or from the P/OS system.

/MACY11 must appear after the file specification in the command line.

/NEW_VERSION

The /NEW_VERSION qualifier specifies that the output files will be created with a version number higher than any existing files of the same name in the directory. /NEW_VERSION must appear after the file specification in the command line.

/NOSPAN

The /NOSPAN qualifier specifies that the output file will be created with records that do not span block boundaries. Normally, only files that already have records that do not span block boundaries remain as such when copied. /NOSPAN must appear after the file specification in the command line.

/REPLACE

With the /REPLACE qualifier, if the output file has the same name, type, and version number as an already existing file at the destination, the first file is deleted and the file you have sent replaces it. The name, type, and version number stay as they were. /REPLACE must appear after the file specification in the command line.

/OWNER:[UIC]

This qualifier specifies the UIC of the file. A UIC establishes the owner of a file. See the *Tool Kit User's Guide* for a description of UIC.

/PROTECTION:code

This qualifier specifies the protection code applied to the output file. See the description of the SET PROTECTION command for details on the code.

3.13 CREATE/DIRECTORY

CREATE/DIRECTORY creates a User File Directory (UFD) on a FILES-11 volume and enters the UFD into the volume's Master File Directory (MFD).

Syntax

CREATE/**DIR**ECTORY [device:] [dirspec] [qualifier[s]]

device:

Specifies the name of the device on which you want the directory created. The default is SY0:. If you do not specify a device or volume name, a directory is created on the default volume, SY0:.

dirspec

Is the name of the directory, enclosed in square brackets, that you want to create. You must specify at least the dirspec parameter.

qualifier

Can be one or more of the following:

/NOWARNINGS
/OWNER_UIC:[UIC]

Prompts

Directory? [device:][dirspec]

Qualifiers

/NOWARNINGS

Supresses error messages that can result from the command.

/OWNER_UIC:[UIC]

Specifies the User Identification Code (UIC) of the directory's owner. A UIC establishes the protection status of the directory. See the *Tool Kit User's Guide* for details on UICs and file protection.

By default, the directory is owned by the issuer of this command. You can specify any UIC as the owner by using the /OWNER_UIC qualifier. However, a non-privileged user cannot always create files in a directory that is owned by another UIC.

3.14 DEASSIGN

This command deletes a logical name assignment created through the use of the ASSIGN command or DEFINE command.

Syntax

DEASSIGN[/qualifier[s]] [logical_name]

qualifier

Can be any combination of the following:

/USER
/SESSION
/SYSTEM
/ALL

logical_name

Is the name of a unique logical name. If you created the logical name using quote characters, you must use quote characters in the logical name to perform the DEASSIGN.

Prompt

Logical name? logical_name

Qualifiers

By default, if you do not specify any of the qualifiers /USER, /SESSION, or /SYSTEM, P/OS performs the operation in the session logical name table.

/ALL

Indicates that all logical names in the specified logical name table be deassigned.

/SESSION

Limits the search for the logical to the session logical name table. This is the default.

/SYSTEM

Limits the search for the logical to the system logical name table. You should not delete any P/OS standard logical names defined in this table. See the *Tool Kit User's Guide* for a list of these logical names.

/USER

Limits the search for the logical to the user logical name table.

3.15 DEBUG

This command executes a debugger that you have built into your task. The command sets the T-bit in the task's Processor Status Word (PSW). To build a debugger into your task, you must specify the /DA qualifier on the PAB command line you use to build your task.

Syntax

DEBUG taskname

taskname

Is the name of the task that you want to debug.

Prompt

None

3.16 DEFINE

This command creates a logical name and assigns an equivalence string to the specified logical name. An equivalence name can be a device name, another logical name, or a file specification.

To specify the logical name table where you want to enter a logical name, use /USER, /SESSION, or /SYSTEM. If you do not explicitly specify a logical name table, the default is /SESSION.

Syntax

DEFINE[/qualifiers] logical_name equiv_name

qualifiers

Can be one or more of the following:

/USER
/SESSION
/SYSTEM
/FINAL
/TRANSLATION_ATTRIBUTES:[(keyword[,...])]

logical_name

Specifies the logical name string. The logical name string can contain from 1 to 255 characters. If the logical name contains quotation marks, enclose the string in quotation marks and use two sets of quotation marks ("") in the places where you want a quotation mark to appear.

Note that if you enclose a logical name string in quotation marks the case of alphabetic characters is preserved.

equiv_name

Defines the equivalence name to be associated with the logical name in the specified logical name table. The equivalence name string can contain from 1 to 255 characters. If the equivalence name contains quotation marks, enclose the string in quotation marks and use two sets of quotation marks ("") in the places where you want a quotation mark to appear.

Note that if you enclose an equivalence name string in quotation marks, the case of alphabetic characters is preserved.

Prompts

```
Logical Name? logical_name
Equivalence Name? equiv_name
```

Qualifiers

/USER

Places the logical name in the user logical name table. Logicals created in the user logical name table are deleted when you exit any application (including DCL).

/SESSION

Places the logical name in the session logical name table. Session logical names are deleted when you logout of the system. If you do not explicitly specify a logical name table, the default is /SESSION.

/SYSTEM

Places the logical name in the system logical name table. All system users can access the logical name.

System logical names are never deleted unless an explicit deassign logical operation is performed.

/FINAL

Indicates that the equivalence name string should not be translated iteratively; logical name translation should terminate with the current equivalence string. The /FINAL qualifier is synonymous with:

```
/TRANSLATION_ATTRIBUTES:TERMINAL
```

/TRANSLATION_ATTRIBUTES:[(keyword[,...])]

Specifies one or more attributes that modify a logical name string. The /TRANSLATION_ATTRIBUTES qualifer is positional and must be specified before the logical name is specified.

You can specify the following keywords for translation attributes:

CONCEALED Indicates that the logical name is a concealed device name. If you specified the CONCEALED attribute, the logical name must be a device name of the form ddnnn: and the equivalence name must be of the form _ddnnn:[directory.]. The device name in the equivalence name must be present. The [directory.] portion of the equivalence name is optional.

TERMINAL Indicates that the equivalence name string should not be translated iteratively; logical name translation should terminate with the current equivalence string.

If you specify only one keyword, you may omit the parentheses. Only the attributes you specify are set.

Examples

```
$ DEFINE CHARLIE: DW001:[CHARLES]

$ PRINT CHARLIE:TEST.DAT
```

The DEFINE command associates the logical name CHARLIE: with the directory name [CHARLES] on the device DW001. Subsequent references to the logical name CHARLIE: result in the correspondence between the logical name CHARLIE: and the disk and directory specified. The PRINT command queues a copy of the file DW001:[CHARLES]TEST.DAT to the default print queue.

```
$ DEFINE CHARLIE: DW001:[CHARLES] /USER
```

The DEFINE command associates the logical name CHARLIE: with the directory name [CHARLES] on the device DW001: and makes the assignment in the user logical name table.

```
$ DEFINE/TRANSLATION_ATTRIBUTES:CONCEALED PD001: -
$-_DW002:[CHARLES.]

$ DIRECTORY PD001:[USERFILES]

    Directory PD1:[USERFILES]
    28-JUN-85 11:20

    TEST.DAT       4       27-JUN-85 17:21

    Total of 4./4. blocks in 1. files
```

The DEFINE command associates the logical name PD001: with the directory name [CHARLES] on the device DW002.

Subsequent references to the device PD001: result in the device and directory specified as being used as the MFD for all file operations.

```
$ DEFINE/TRANSLATION_ATTRIBUTES:TERMINAL TEMP: XX001:
```

The DEFINE command associates the logical name TEMP: with the device XX001. Subsequent references to the logical TEMP: will result in the device XX001: being used without any further translation attempts on the equivalence name XX001:.

3.17 DELETE

DELETE removes a file specification entry from a directory, releases the corresponding storage space occupied by those files, and causes any data in the corresponding file or files to be inaccessible.

Syntax

DELETE[/qualifier[s]] filespec[,s]

qualifier

Can be one or more of the following:

/LOG
/QUERY or /CONFIRM
/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yyy
/TODAY
/EXCLUDE:filespec
/[NO]WARNINGS

filespec

Specifies the file or files to be deleted. You must specify the name, file type, and version number fields of the filespecs you want to delete. Device and directory fields default to your current device and directory. DELETE accepts wildcards in any field except the device field. You need not supply a file type to delete a file with a null file type.

Prompts

Files(s)? filespec[,s]

Qualifiers

/LOG

Specifies that a list of the files deleted be displayed on your terminal.

/QUERY or /CONFIRM

Specifies that you want to decide which files should be deleted on an individual basis. Each file that is specified in the command is named. You may enter one of four characters:

Y - Deletes file named and goes on to next file.
N - Does not delete file named and goes on to next file.
G - (Go) Deletes the file and goes on to delete all other files specified.
Q - (Quit) Does not delete the file and exits the DELETE command. No more files are deleted.

Remember that you can specify files by default or wildcard. If you do not specify a version number, /QUERY is the default.

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the DELETE command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the DELETE command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the DELETE command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the DELETE command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the DELETE command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the DELETE command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/[NO]WARNINGS

The /NOWARNINGS qualifier specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

3.18 DELETE/DIRECTORY

This command deletes an empty directory. An error message appears if you attempt to delete a directory that is not empty.

Syntax

DELETE/**DIR**ECTORY [device:][dirspec]

device:

Is an optional device name, such as DW1: or SY:. If you do not specify a device name, the system searches the current default device for the directory to delete. See the *Tool Kit User's Guide* for further details on device names.

dirspec

Is a directory name enclosed in square brackets. An example is [USERFILES].

Prompts

Directory? [device:][dirspec]

3.19 DIBOL

This command invokes the PRO/Tool Kit DIBOL compiler to compile DIBOL source code files.

NOTE

Please refer to the language documentation for additional information.

Syntax

DIBOL[/qualifier[s]] filespec[,s][/qualifier[s]]

qualifier

Can be one or more of the following:

/[NO]BUILD
/[NO]CROSS_REFERENCE
/[NO]DEBUG
/[NO]LIST[:filespec]
/[NO]OBJECT[:filespec]
/[NO]OPTIMIZE
/PAGE_SIZE:number (D:system default)
/[NO]STANDARD
/[NO]SHOW
/[NO]WARNINGS

filespec

Is the name of a DIBOL source code file.

NOTE

If you want to reference a universal library containing source code modules, use the following format:

DIBOL[/qualifiers[s] filespec[,s][/qualifier[s] libr_spec/LIBR

libr_spec

Is the name of the universal (default) library.

/LIBR

Signals the DIBOL compiler that libr_spec is a universal (default) library. You may include only one such library in each compilation.

Prompts

File(s)? filespec[,s]

Qualifiers

/[NO]BUILD

/BUILD causes the build option to be invoked automatically after the compilation phase. This qualifier creates .CMD and .ODL files for direct input to the Professional Application Builder (PAB).

/[NO]CROSS_REFERENCE

The /CROSS_REFERENCE qualifier causes the compiler to produce a list file and to add a cross reference table to the end of the listing file.

/[NO]DEBUG

/DEBUG includes extra information for the DIBOL debugger. The default is /NODEBUG which prevents the inclusion of this information.

/[NO]LIST[:filespec]

/LIST causes the DIBOL to output a file with the same name as the source listing. /LIST creates this output. You may change the name of this output file by typing:

/LIST:new_filespec

Refer to NOTE in /[NO]OBJECT.

/[NO]OBJECT[:filespec]

/OBJECT creates an object module with the same name as the source listing and is the default value. /OBJECT: new_filespec creates an object file with a filename different than the source filename. /NOOBJECT suspends the creation of an object file. (When you list more than one source file on the command line and you specify /OBJECT, DIBOL creates an object file with the name of the first source file listed.)

NOTE

Use of the qualifiers /NOOBJECT and /NOLIST allows the DIBOL command to check for errors.

/PAGE_SIZE:n

The /PAGE_SIZE qualifier specifies the number of lines per page in the listing file.

/OPTIMIZE

/OPTIMIZE causes DIBOL to optimize the object file. The default is /NOOPTIMIZE which does not produce an optimized object file.

/[NO]STANDARD

/STANDARD flags all occurrences of P/OS (RSX-11M-PLUS) extensions to DIBOL-83. This is the default.

/SHOW:(arg[,...])

Table 3-1 shows the value of arg. In the table, the letter "D" indicates the default argument for this qualifier.

Table 3-1: Argument Values for /SHOW Qualifier

Argument	Default	Description
ALL		Shows everything.
NONE		Shows only source (nothing else).
COND	D	Shows non-compiled conditionals.
NOCOND		Does not show non-compiled conditionals.
INCLude	D	Shows included files.
NOINCL		Does not show included files.
TABLes		Shows symbol and label tables.
NOTABL	D	Does not show symbol and label tables.
SOURce	D	Shows DIBOL source.
NOSOUR		Does not show DIBOL source.

/[NO]WARNINGS

/WARNINGS displays (or prints if accompanied by /LIST) warning messages produced by the DIBOL compiler; /WARNINGS is the default value. /NOWARNINGS suppresses the display (listing) of warning messages.

3.20 DIFFERENCES

This command invokes the Compare (CMP) utility (refer to Chapter 5), which compares two ASCII (text) files, line by line, to determine if parallel records (lines) are identical. After this comparison, DIFFERENCES produces a listing of any differences between the two files.

Syntax

DIFFERENCES[/qualifier[s]] input_file1 input_file2

qualifier

Can be one or more of the following:

/CHANGE_BAR[:n]
/IGNORE:(arg[,s])
    BLANK_LINES
    COMMENTS
    FORM_FEEDS
    SPACING
    TRAILING_BLANKS
/LINES:n
/NONUMBERS
/OUTPUT:filespec
/SLP[:au]

input_file1

Specifies the first of two files to be compared. The filename must be included, and the default file type is .MAC.

input_file2

Specifies the second of two files to be compared. The filename must be included, and the default file type is .MAC.

Prompts

Filespec1? input_file1
Filespec2? input_file2

Qualifiers

/CHANGE_BAR[:nnn]

Specifies that the output consists of a listing of infile2 with change bars applied to each line in infile2 that does not have a corresponding line in infile1. The nnn argument is the number of the ASCII character to be used for the change bar. The default is 041, the exclamation point (!). The following are codes for some commonly used ASCII characters:

!	041
#	043
$	044
*	052
+	053
<	074
>	076
?	077
|	074

/IGNORE:(arg[,s])

BLANK_LINES
COMMENTS
FORM_FEEDS
SPACING
TRAILING_BLANKS

Specifies that you want certain defaults to be ignored in making the comparison. Without the /IGNORE qualifier, the comparison is strictly line-by-line. If you specify only one argument, you do not need the parentheses. If you specify more than one argument, enclose them in parentheses and separate them with commas (,).

The default is to include all these factors in the comparison.

The BLANK_LINES argument specifies that blank lines are to be excluded from comparison. The default is to include blank lines in the comparison.

The COMMENTS argument specifies that comments, that is, text preceded by a semicolon (;) not be included in the comparison. Otherwise, comments are compared like all other text.

The FORM_FEEDS argument specifies that lines beginning with a single form feed character (CTRL/L) are to be excluded from the comparison. The default is to include such lines in the comparison.

The SPACING argument specifies that any sequence of blank and tab characters is to be interpreted as a single blank for the comparison. This argument is useful when comparing source files that differ only in their spacing. The output listing includes blanks and tabs as they are found in the input files, but the blanks and tabs are ignored. The default is to compare sequences of blanks and tabs like any other characters.

The TRAILING_BLANKS argument specifies that all blanks following the last nonblank character in the line be ignored in the comparison. The default is to compare trailing blanks like all other characters. If you specify both TRAILING_BLANKS and COMMENTS in your /IGNORE qualifier, any blanks preceding a semicolon (;) are also ignored.

The DCL DIFFERENCES command does not have quite the same defaults as CMP, the File Compare Utility, which DIFFERENCES invokes. Specify /IGNORE:(BLANK_LINES,FORM_FEEDS) to duplicate the CMP defaults for blank lines and form feeds. Other CMP defaults need not be specified.

/LINES:n

Specifies that n lines must be identical before a match is recognized. The default value for n is three identical lines.

/[NO]NUMBERS

/NONUMBERS specifies that lines in the output file not be preceded by line numbers. The standard output listing automatically includes line numbers. Line numbers start with 1 in increments of one. All lines are numbered, including blanks.

/OUTPUT:filespec

Specifies that the output listing be written to the file named in the qualifier. Normally, this output appears on your terminal.

/SLP[:au]

Specifies that the output listing take the form of an SLP indirect command file. When you include this qualifier, the output listing is an SLP indirect command file that makes infile1 identical to infile2. The optional au argument, preceded by a colon, is a 1- through 8-character alphanumeric audit trail symbol. For more information on the Source Language Input Program (SLP), see Chapter 10.

3.21 DIRECTORY

DIRECTORY displays information for an individual file or a group of files.

Syntax

DIRECTORY[/format-qual][/other-qual[s]] [filespec[,s]]

format-qual

Controls the appearance of the directory list and can be one of the following:

/FULL
/BRIEF
/FREE[ddnnn:]
/SUMMARY
/ATTRIBUTES

other-qual

Modifies the list of files by creation date or exclusion and may be one or more of the following:

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/OUTPUT:
/NOWARNINGS

With PRO/DECnet DCL extensions, the following qualifier is also available:

/WIDTH:n

filespec[,s]

Specifies the file or files for which information should be displayed. If you do not supply a filespec, the complete default directory is displayed.

You can supply one or more filespecs, separated by commas, to display directory information on the files you name.

You can use wildcards in place of any filespec field except the device field. If you do not supply a version number, only information on the most recent versions is displayed. However, if you do not supply a file type, a null file type is assumed. If you do not know the file type, use a wildcard.

You can display another directory by supplying the directory name in this field. You can also specify device names in the form ddnnn: in this field.

If you name files in two directories, you should name files for the default directory first. If you name files from another device or directory first, the defaults are canceled.

Prompts

None

Qualifiers

Qualifiers are in two groups:

1. Format qualifiers control the appearance of the directory.

2. Other qualifiers modify the list of files by creation date or exclusion.

If you do not supply a qualifier from the format-qual list, the display is in standard format, which shows: the filename, type, and version number; the number of blocks the file occupies; and the date and time of creation.

If you do not include any qualifier from the other-qual list, the display includes all files that otherwise qualify.

Format Qualifiers

/BRIEF

Specifies that the display give filenames, types, and version numbers only.

/FULL

Specifies that the complete directory entry be displayed, including file ID number, blocks used and allocated, the owning UIC, protection status of the file, in addition to all the information in the standard display.

/SUMMARY

Specifies that the display give only the total number of blocks allocated and used for the specified files. If you give no filespecs in the command, the display shows the total blocks allocated and used for the default directory.

/FREE [ddnnn:]

Specifies that the display give the free space and number of free file headers on the default device or a specified device.

/ATTRIBUTES

Displays the Record Management Services (RMS-11) attributes of a file or files. This display includes filename and type, creation date and time, file organization, protection status, allocation information, last access date and time, record format, record size, primary and alternate key definitions for indexed files, and bucket size for indexed and relative files. This qualifier invokes the RMS-11 DSP utility.

NOTE

This display is produced for any file you specify in a DIRECTORY command, whether or not the file is an RMS-11 file.

Some display information may not be meaningful for non-RMS files.

Other Qualifiers

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the DIRECTORY command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the DIRECTORY command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the DIRECTORY command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the DIRECTORY command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the DIRECTORY command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the DIRECTORY command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/OUTPUT:filespec

The OUTPUT qualifier specifies that the output of the DIRECTORY command be placed in a file having the file specification you supply.

/[NO]WARNINGS

The /NOWARNINGS qualifier specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

Additional Qualifiers Available with PRO/DECnet

Command Qualifiers

/WIDTH:n

The /WIDTH qualifier specifies the width of the display for the default and /FULL format displays only. The default value is 72. At least one field is always displayed per line, even if that field exceeds the maximum width specified.

3.22 DISMOUNT

This command marks the volume mounted on the specified device to be logically off-line and disconnected from the file system.

Syntax

DISMOUNT device: [volumelabel]

device:

Is the name of the device on which the magnetic medium containing the volume is currently mounted. This argument can be in the form ddnnn:, or it can be a logical name previously assigned to the device by the MOUNT command.

volumelabel

Is the name of the volume. If the volume currently mounted in devname does not match the volume label you specify, the system does not dismount the device.

Prompt

Device? device: [volumelabel]

3.23 DUMP

DUMP displays or prints the contents of a file or volume in ASCII, hexadecimal, octal, or decimal format. This command invokes the File Dump (DMP) Utility. (Refer to Chapter 6.)

Syntax

DUMP[/qualifier[s]] filespec

qualifier

Can be one or more of the following:

/ASCII
/BASE_ADDRESS:n:m
/BLOCKS:n:m
/BYTE
/DECIMAL
/FILE:file-number:sequence-number
/FILE_HEADER[:[NO]FORMATTED]
/HEADER
/HEXADECIMAL
/IDENTIFICATION
/LOGICAL_BLOCK
/LONGWORD
/LOWERCASE
/NUMBER[:n]
/OCTAL
/OUTPUT:file-name
/RADIX_50
/RECORD
/WORD

filespec

Is the file specification to be displayed (dumped).

Prompts

File? filespec

Qualifiers

/ASCII

/ASCII specifies that the data should be dumped one byte at a time in ASCII mode. This qualifier displays the full DEC Multinational Character Set. The control characters (0-37) are printed as a circumflex (^), followed by the alphabetic

character corresponding to the character code plus 100. For example, bell (code 7) is printed as ^G (code 107). The control characters from 200 (octal) through 240 (octal) are displayed as Chr, where Chr is the control character + 100 (octal).

Lowercase characters (140-177) are printed as a percent sign (%), followed by the corresponding uppercase character (character code minus 40), unless the /LOWERCASE qualifier is specified. The /ASCII and /OCTAL qualifiers are mutually exclusive when dumping bytes.

/BASE_ADDRESS:n:m

This qualifier specifies a two-word base block address (the initial base address is 0,0), where n is the high-order base block address (octal), and m is the low-order base block address (octal). The address may also be specified in decimal by using a period after the number. All future block numbers specified by the /BLOCKS qualifier will be added to this value to obtain an effective block number. This qualifier is useful for specifying block numbers that exceed 16 bits. For example:

```
/BASE_ADDRESS:1:0
```

specifies that all future block numbers will be relative to 65536 (decimal) (200000 octal). Also, the following clears the base address:

```
/BASE_ADDRESS:0:0
```

When the /BASE_ADDRESS appears in a command line, no blocks are dumped. The only result of the command line is to set the base address.

NOTE

The following pertains if you run DUMP directly as a utility.

Once this qualifier is specified, it remains in effect until it is used again to set a new base address. The /BASE_ADDRESS qualifier would be the /BA:n:m qualifier (refer to Chapter 6 for further information).

/BLOCKS:n:m

Specifies the range of blocks to be dumped, where n is the first block and m is the last block. The values of n and m must not exceed 16 bits. In file mode only, the /BLOCKS:n:m qualifier is not required. If the /BLOCKS:n:m qualifier is not specified, DMP will dump all blocks of the specified file, relative to the current base address.

If /BLOCKS:n:m is specified in file mode, it specifies the range of virtual blocks to be dumped. If /BLOCKS:n:m is specified as /BLOCKS:0 file mode, no virtual blocks are dumped. This is useful for dumping only the header portion of the file (see /HEADER). The /BLOCKS:n:m qualifier and the /RECORD qualifier are mutually exclusive.

The /BLOCKS:n:m qualifier is a required parameter in device mode. When used in device mode, it specifies the range of logical blocks to be dumped.

The value n represents the block number of the first block dumped. Successive blocks are labeled with a block number one higher than the preceding block number. The dump will continue until the block labeled m is dumped.

/BYTE

Specifies that the data be dumped in octal byte format.

/DECIMAL

Specifies that the data be dumped in decimal word format.

/FILE:file-number:sequence-number

In file mode, the file number can be used instead of a filename as a file specification for input.

/FILE_HEADER[:[NO]FORMATTED]

This qualifier is an optional parameter used in file mode. In addition, this qualifier has two options. FORMATTED, the default, causes a FILES-11 formatted dump of the header. NOFORMATTED specifies an unformatted octal dump. An octal dump occurs when DUMP is used on non-FILES-11 headers. If specified, the /HEADER[:[NO]FORMATTED] qualifier causes the file header as well as the specified or implied portion of the file to be dumped.

/HEADER

Specifies the format for data blocks that have the FILES-11 header structure. Other blocks are output as an unformatted octal dump.

/HEXADECIMAL

Specifies that the data be dumped in hexadecimal byte format. Note that a hexadecimal dump reads from right to left. (See also the /LONGWORD and /WORD qualifiers.)

/IDENTIFICATION

Causes DMP's version to be identified. This qualifier may be specified on a command line by itself at any time.

/LOGICAL_BLOCK

Requests logical block information for a file. The starting block number and a contiguous or noncontiguous indication for the file are displayed.

/LONGWORD

Specifies that the data be dumped in hexadecimal double-word format.

/LOWERCASE

Specifies that the data should be dumped in lowercase characters.

/NUMBER[:n]

Specifies a memory dump and allows control of line numbers. Line numbers are normally reset to zero whenever a block boundary is crossed. The /NUMBER[:n] qualifier allows lines to be numbered sequentially for the full extent of the file, that is, the line numbers are not reset when block boundaries are crossed. The optional value (:n) specifies the value of the first line number. The default is 0. The /NUMBER[:n] qualifier is used with the output file specification.

/OCTAL

Specifies that the data should be dumped in octal format in addition to other formats specified. If no format qualifiers are specified, the default is octal. The /ASCII qualifier and the /OCTAL qualifier are mutually exclusive when dumping bytes.

/RADIX_50

Specifies that data be dumped in Radix-50 format words.

/RECORD

Specifies that data be dumped a record at a time (rather than a block at a time). The data format is determined by setting any of these format qualifiers: /ASCII, /DECIMAL, /HEXADECIMAL, /LONGWORD, /RADIX50, or /WORD.

The largest record that DUMP can process is 512 (decimal) bytes.

The /RECORD qualifier and the /BLOCK qualifier are mutually exclusive.

/WORD

Specifies that the data be dumped in hexadecimal word format.

3.24 EDIT OR EDIT/EDT

EDIT starts an interactive editing session with the EDT editor to create or modify a file.

Syntax

EDIT[/qualifier[s]] filespec

qualifier

Can be one or more of the following:

/[NO]COMMAND[:filespec]
/[NO]JOURNAL[:filespec]
/[NO]OUTPUT[:filespec]
/[NO]RECOVER
/[NO]READ_ONLY

filespec

Is the name of an existing file or the name of a file to be created. You must specify a filename, but the file type may be null. For example, the command:

```
EDIT filename
```

causes EDT to edit filename.;

Prompts

File? filespec

Qualifiers

/[NO]COMMAND[:filespec]

Controls whether an EDT initialization file is read by EDT before editing begins. These files contain commands that alter the default setup for EDT, such as custom line-mode commands and change-mode key definitions.

The default is /COMMAND:LB:[1,2]EDTSYS.EDT. If you use this qualifier and EDTINI.EDT or some other file you name does not exist, EDT issues no error message and continues with the editing session.

If you do not want to use LB:[1,2]EDTSYS.EDT, use the /NOCOMMAND qualifier.

/[NO]JOURNAL[:filespec]

Controls whether EDT creates a journal file for the editing session. The default is to create a journal file with a filename the same as that of the input file with the type .JOU. You can specify a different name by including a filespec.

The journal file consists of all editing commands and text entered during the session. If the editing session ends abnormally, such as through a system crash, or your inadvertently typing three CTRL/Zs in succession, the journal file is saved. In such a case, you invoke EDT again, with the same command line as before plus the /RECOVER qualifier. Your editing session is repeated and all your editing is restored. If the editing session ends normally, the journal file is deleted.

If you specify /NOJOURNAL, no journal file is created and no recovery is possible.

/[NO]OUTPUT[:filespec]

If you do not specify this qualifier, the default is to create a file of the same name and type as the input file with a version number one higher than the input file. If the file is new, EDT creates version number 1. You can alter the name of the output file by including a filespec with the /OUTPUT qualifier. Otherwise, the qualifier need not be included.

If you specify /NOOUTPUT, you cannot exit EDT without including a filespec in your EDT EXIT command.

/[NO]READ_ONLY

Specifies whether you want simply to read the file or to edit it. If your command line includes /READ_ONLY, you can use the full facilities of EDT, but you cannot exit without including a filespec in your EDT EXIT command. Normally, you would use the EDT QUIT command if you had specified /READ_ONLY. The /READ_ONLY qualifier is equivalent to a combination of /NOOUTPUT and /NOJOURNAL. You can use /READ_ONLY to look at files to which you have no write access.

The default is /NOREAD_ONLY, which need never be specified.

/[NO]RECOVER

Specifies whether EDT reads commands from a journal file prior to starting the editing session. With a journal file, your editing session can be restored if interrupted by a system crash or other problem.

The default is /NORECOVER, which need never be specified.

The /RECOVER qualifier requests EDT to open the input file and then read EDT commands and text from the file with the same filename as the input file and the file type .JOU. The command line with /RECOVER added to it must be identical to the command line that initiated the original failed editing session. This means that if you specified an EDT initialization file, you must specify the same file in the /RECOVER command line. If you specified a name for the journal file other than infile.JOU, you must include the /JOURNAL qualifier with the appropriate filespec.

If journaling was not enabled on the original command line, you cannot recover the editing session.

3.25 EDIT/PROSE

EDIT/PROSE invokes the PROSE editor from DCL.

NOTE

This section only describes the DCL command that accesses the PROSE editor. For further information on PROSE you should turn to the *Prose User's Guide*.

Syntax

EDIT/**PR**OSE/qualifier filespec

qualifier

/OUTPUT:filespec

filespec

Is the name of an existing input file that PROSE will edit or a new file spec that PROSE will first create and then edit.

NOTE

Whenever you leave a PROSE editing session, you cannot discard your edits from that session. If you do not want the output file, you have to delete it after exiting the editing session.

Prompts

File? filespec

Qualifier

/OUTPUT:filespec

Causes EDIT/PROSE to name the output file with a different filename than the input filename.

3.26 EDIT/SLP

This command invokes the Source Language Input Program (SLP), which is an editor designed for maintaining and updating source files. (Refer to Chapter 10.)

Syntax

EDIT/**SL**P[/qualifier[s]] filespec

qualifier

Can be one or more of the following:

```
/[NO]AUDIT[:arg]
           POSITION:n
           SIZE:n
/[NO]CHECKSUM[:arg]
/[NO]LIST[:filespec]
/[NO]OUTPUT[:filespec]
/[NO]REPORT
/[NO]TAB
/[NO]TRUNCATE[:n]
```

filespec

Is the name of the source program to be updated by SLP.

Prompts

```
File? filespec
```

Qualifiers

```
/[NO]AUDIT[:(arg[s])]
        POSITION:n
        SIZE:n
```

Controls whether the output file includes an audit trail, and optionally allows you to specify the location and size of the audit trail. You can specify one or both of these values. If you specify only one, you can omit the parentheses, but the parentheses are required syntax if you specify both POSITION and SIZE. Separate the two arguments within the parentheses by a comma (,).

The POSITION: argument sets the starting position of the audit trail. The value of n can be from 0 through 132, representing the column at which the first character in the audit trail is to appear. This value is rounded up to the

next highest tab stop. The default is to start the audit trail at column 80. Note that this default causes audit trails of more than a single character to wrap around when displayed on standard video terminals.

The SIZE: argument sets the length of the audit trail. The value of n can be from 0 through 14. The default is an audit trail of 8 characters. The audit trail itself is defined from within SLP. See Chapter 10.

/[NO]CHECKSUM[:arg]

Controls whether a checksum is calculated for the SLP commands. If you specify checksum without an argument, SLP calculates the checksum value and prints it on your terminal. If you specify an argument, SLP calculates the checksum and compares it to what you have specified. If the numbers differ, a warning message is displayed, but the execution of SLP is not interrupted.

The default is /NOCHECKSUM.

/[NO]LIST[:filespec]

The /LIST qualifier creates a listing of a file with line numbers. If you do not give a filespec, the default filespec is filename.LST.

The default is /LIST. /NOLIST suppresses creation of the listing file.

/[NO]OUTPUT[:filespec]

Use this qualifier to change the name of the output file. The default output filespec is the same name and type as the input file and a version number one higher than the highest existing version of the file. If you do not want to override this default, you do not need this qualifier.

The /NOOUTPUT qualifier suppresses creation of an output file.

/[NO]REPORT

Controls whether line truncations that result from audit trails are reported. If you specify /REPORT, you receive warning messages on your terminal and the affected lines are marked with a question mark (?) in place of the period (.) in the line number in the listing file.

The default is /NOREPORT.

/[NO]TAB

Controls whether SLP replaces tabs or spaces at the end of each record containing an audit trail. If you specify /TAB, tabs are inserted. If you specify /NOTAB, spaces are inserted. The default is /NOTAB.

/[NO]TRUNCATE[:arg]

The TRUNCATE qualifier requests SLP to truncate each record in the input file when it creates the output file. This qualifier allows you to delete an audit trail from a file previously updated with SLP. If you specify /TRUNCATE without a number, SLP truncates input records at the beginning position of the audit trail. If you specify a number, SLP truncates the records beginning at the column. The value of n can be from 0 through 132.

The default is /NOTRUNCATE.

3.27 EXIT

Invoking this command, or pressing the EXIT key, causes you to exit PRO/Tool Kit.

You should not exit PRO/Tool Kit if any tasks are still active. This includes active tasks submitted to background mode by the SPAWN command. (Refer to Section 2.4.) You should either allow all tasks to finish, or you should abort them.

Syntax

EXIT

Prompts

None

3.28 FIX

This command loads and locks a task into its partition. Subsequent requests for running the task are serviced more quickly because the task is memory-resident and does not have to be loaded from the disk before it can run.

Fixed tasks remain physically in memory even after they exit. Therefore, they do not have to be reloaded when a request is made to run them. (Note that reexecuting fixed tasks is not always practical; to reexecute, the task must initialize all impure data and LUN assignments at run time.) Only a REMOVE command can free the occupied memory partition.

Note the following requirements:

- You must install a task before you fix it.
- You cannot fix a task whose name is in the form ...xxx or xxx$$$. This is because the copy of the task with that form of name is a nonexecutable, prototype task.
- You cannot fix an active task.
- You should not fix a checkpointable task because doing so makes the task noncheckpointable.

See also the INSTALL command with the /FIX qualifier.

Syntax

FIX taskname [/qualifier]

taskname

Is the task or region to fix in memory.

qualifier

/REGION
/READONLY_SEGMENT

Prompt

Taskname? taskname

Qualifier

/REGION

Indicates that the task to be fixed is a region, such as a read-write common.

/READONLY_SEGMENT

Fixes only the read-only segment of a multiuser task.

3.29 FORMAT

This command sets up a hard disk so that that it is recognizable as a FILES-11 volume by its controller. Once you have formatted a disk, you can then use the INITIALIZE command to initialize it.

Normally, the P/OS installation procedure formats the system disk. However, to connect additional hard disks to a Professional you must format each one by invoking the FORMAT command.

Syntax

FORMAT[/qualifier] device:

qualifier

/[NO]CONFIRM

device:

Is the name of the device on which the magnetic medium containing the volume is currently mounted. This argument can be in the form ddnnn[:], or it can be a logical name created with the ASSIGN or DEFINE command.

Prompt

Device? device:

Qualifier

/[NO]CONFIRM

The /NOCONFIRM qualifier specifies that P/OS should not verify that the volume is already in FILES-11 format. The /CONFIRM qualifier asks you to verify the format request if the volume is already in FILES-11 format. The /CONFIRM qualifier is the default.

3.30 FORTRAN

FORTRAN invokes the PRO/Tool Kit FORTRAN-77 compiler to compile FORTRAN language source files.

NOTE

Please refer to the language documentation for additional information.

Syntax

FORTRAN[/qualifier[s]] filespec[,s]

qualifier

Can be one or more of the following:

```
/[NO]CHECK
/CONTINUATIONS:n
/[NO]DEBUG
/[NO]DLINES
/IDENTIFICATION
/[NO]I4
/[NO]LIST[:filespec]
/[NO]MACHINE_CODE
/[NO]MAP
/[NO]OBJECT[:filespec]
/[NO]OPTIMIZE
/[NO]SHAREABLE
/SOURCE
/[NO]STANDARD:arg
              ALL
              NONE
              SOURCE
              SYNTAX
/[NO]TRACEBACK:arg
               ALL
               BLOCKS
               LINES
               NAMES
               NONE
/WORK_FILES:n
```

filespec

Is the name of the FORTRAN source file.

Prompts

File(s)? filespec[,s]

Qualifiers

/[NO]CHECK

The /CHECK qualifier specifies that you want the compiler to check that all array references are within bounds.

The default is /NOCHECK.

/CONTINUATIONS:n

The /CONTINUATIONS qualifier specifies the maximum number of continuation lines permitted in the code. The value of n can be from 0 through 99. The default is 5.

/[NO]DEBUG

Specifies that the compiler is to provide symbol table information for use by the PRO/TOOL KIT FORTRAN-77 symbolic debugger. When you use the /DEBUG command qualifier, you should also use the /NOOPTIMIZE command qualifier. /NODEBUG is the default.

/[NO]DLINES

The /DLINES qualifier specifies that lines with a "D" character in column 1 are to be compiled. The default is /NODLINES, meaning these lines are treated as comment lines. Use this feature for debugging.

/IDENTIFICATION

The /IDENTIFICATION qualifier causes the compiler to print its identification and version number on your terminal.

/[NO]I4

The /I4 qualifier causes a two-word default allocation for integer variables. The default is /NOI4, meaning the compiler causes a single-word allocation for any integer variables not given an explicit length specification.

/[NO]LIST[:filespec]

Specifies whether a compiler listing should be generated. The default is /NOLIST, meaning no compiler listing is generated.

If you do not supply a file specification for this qualifier, the listing has a filename derived from the name of the first source file in the FORTRAN command with the file type .LST. If you want the listing to have a different name, supply the name as an argument to the /LIST qualifier. If you do supply a name, the listing file appears in your directory but is not printed on the printer.

The /LIST qualifier behaves in a different manner depending on whether it is given as a command qualifier or a filespec qualifier. If /LIST:LP: is used as a command qualifier, the listing file is placed in your directory and printed on the printer. If /LIST is used as a filespec qualifier, the listing file appears in your directory but is not printed on the printer. If /LIST is used as a filespec qualifier, the listing file takes the name of that file.

NOTE

The /MACHINE_CODE, /MAP, and /SOURCE, qualifiers all affect the contents of the compiler listing file.

/[NO]MACHINE_CODE

The /MACHINE_CODE qualifier specifies that you want the compiler listing to include binary machine code and diagnostics. The default is /NOMACHINE_CODE.

/MACHINE_CODE implies /LIST. You do not need to include the /LIST qualifier unless you want to use it to establish a name for the listing file different from any of the filenames included in the compilation.

The /MACHINE_CODE qualifier behaves in a different manner depending on whether it is given as a command qualifier or a filespec qualifier. If /MACHINE_CODE is used as a command qualifier, the listing file is placed in your directory and printed on the printer. If /MACHINE_CODE is used as a filespec qualifier, the listing file appears in your directory but is not printed on the printer. If /MACHINE_CODE is used as a filespec qualifier, the listing file takes the name of that file.

/[NO]MAP

The /MAP qualifier specifies that you want the compiler listing to include a storage map and diagnostics.

/MAP implies /LIST. You do not need to include the /LIST qualifier unless you want to use it to establish a name for the listing file different from any of the filenames included in the compilation.

The /MAP qualifier behaves in a different manner depending on whether it is given as a command qualifier or a filespec qualifier. If /MAP is used as a command qualifier, the listing file is placed in your directory and printed on the printer. If /MAP is used as a filespec qualifier, the listing file appears in your directory but is not printed on the printer. If /MAP is used as a filespec qualifier, the listing file takes the name of that file.

/[NO]OBJECT[:filespec]

Determines whether or not an object module is generated by the compiler. The default is /OBJECT, which does create a module. The default name of the object file created by FORTRAN is the last-named source file with the file type .OBJ. If you want the object file to have a different name, supply the name as an argument to the /OBJECT qualifier.

/NOOBJECT specifies that no object module is created. You can use the /NOOBJECT qualifier to get a compiler listing file to check for errors without generating object code.

/[NO]OPTIMIZE

Controls whether the compiler optimizes the compiled program to generate more efficient code.

Use /NOOPTIMIZE in conjunction with the /DEBUG qualifier to link a P/OS FORTRAN-77 program with the debugger so that variables always contain their updated values.

/[NO]SHAREABLE

The /SHAREABLE qualifier states that you want the compiler to generate pure code and pure data sections as read-only. This takes advantage of code sharing in multiuser tasks on P/OS.

/SOURCE

The /SOURCE qualifier specifies that you want the compiler listing to include the source code.

/SOURCE implies /LIST. You do not need to include the /LIST qualifier unless you want to use it to establish a name for the listing file different from any of the filenames included in the compilation.

The /SOURCE qualifier behaves in a different manner depending on whether it is given as a command qualifier or a filespec qualifier. If /SOURCE is used as a command qualifier, the listing file is placed in your directory and printed on the printer. If /SOURCE is used as a filespec qualifier, the listing file appears in your directory but is not printed on the printer. If /SOURCE is used as a filespec qualifier, the listing file takes the name of that file.

/[NO]STANDARD:arg

```
ALL
NONE
SOURCE
SYNTAX
```

The /STANDARD qualifier directs the compiler to look in your source code for extensions to ANSI standard (X3.9-1978) FORTRAN at the full-language level. If the compiler finds extensions, it flags them and produces informational diagnostics.

The ALL argument produces informational diagnostics for all detected extensions.

The NONE argument produces no informational diagnostics.

The SOURCE argument produces informational diagnostics for lowercase letters and tab characters in the source code.

The SYNTAX argument is the same as /STANDARD with no argument.

The default is /NOSTANDARD.

/[NO]TRACEBACK:arg

```
ALL
BLOCKS
LINES
NAMES
NONE
```

/TRACEBACK controls the amount of extra code included in the compiled output for use by the OTS during error traceback. This code is used in producing diagnostic information and in identifying which statement in the FORTRAN source program caused an error condition to be detected during execution.

The ALL argument states that error traceback information is to be compiled for all source statements, and function and subroutine entries.

The LINES argument is the same as ALL.

The BLOCKS argument states that traceback information is to be compiled for subroutine and function entries and initial statements in sequences called blocks. This is the default.

The NAMES argument states that traceback information is to be compiled only for subroutine and function entries.

The NONE argument states that no traceback information is to be produced.

/WORK_FILES:n

The /WORK_FILES qualifier specifies the number of temporary on-disk files you want used during the compilation. The argument n can range from 0 through 3; the default is 2. Increasing the number of work files increases the maximum possible size of your program but decreases the speed of compilation.

3.31 HELP

HELP displays information about the commands and utilities.

Syntax

HELP[/qualifier] [topic]...

qualifier

Can be one or more of the following:

/OUTPUT:filespec
/FILE:filespec
/filename
/LOCAL

topic...

Are up to four optional topics, separated by spaces, for which you want help.

Prompts

None

Qualifiers

/OUTPUT:filespec

Permits you to name an output file where the requested help text is to be saved. The default is /OUTPUT:TI:.

/FILE:filespec

Specifies any file where help text is located. If you do not give a complete file specification, the defaults are LB:[1,2]filename.HLP.

/filename

Specifies that the help text begins with LB:[1,2]filename.HLP.

/LOCAL

Specifies that the help text is in the file HELP.HLP in the default directory on the default device.

3.32 INITIALIZE

This command deletes all existing files on a volume and imparts the standard FILES-11 file structure. INITIALIZE can also optionally check for bad blocks. Note that the volume must be formatted (see the FORMAT command) prior to being initialized.

Syntax

INITIALIZE[/qualifier[s]] device: volumelabel

qualifiers

/[NO]BAD_BLOCKS
/CHECKPOINT_SPACE:n
/[NO]CONFIRM

device

Is the name of the device on which the magnetic medium containing the volume is currently mounted. This argument can be in the form ddnnn[:], or it can be a logical name previously assigned to the device.

volumelabel

is the label that names the volume after initialization. It can be from 1 to 12 characters long.

Prompts

Device? device
Volume name? volumelabel

Qualifiers

/[NO]BAD_BLOCKS

/BAD_BLOCKS, the default, performs a check for bad blocks on the volume you are initializing. If any bad blocks are found, P/OS will not use them for storing information. If you specify /NOBAD_BLOCKS, P/OS does not search for bad blocks on the volume.

/CHECKPOINT_SPACE:n

Allows you to control the size of a system checkpoint file on the volume being initialized.

/[NO]CONFIRM

The /NOCONFIRM qualifier specifies that P/OS should not verify that the volume is already in FILES-11 format. The /CONFIRM qualifier asks you to verify the initialization request if the volume is already in FILES-11 format. The /CONFIRM qualifier is the default.

3.33 INSTALL

INSTALL includes a specific task in the System Task Directory, thus making it known to the system.

An installed task is dormant until it is requested to run by the Executive. You can request an installed task to run through the RUN command or through a variety of Executive directives under program control. Examples of such directives are: RQST$ (Request), RUN$ (Run), and SPWN$ (Spawn). See the *P/OS System Reference Manual* for details on the directives.

Syntax

INSTALL[/qualifier[s]] [$]filespec

qualifier

Can be one or more of the following:

/FIX
/NOREMOVE
/READONLY_COMMON
/TASK_NAME:taskname
/[NO]WRITEBACK

[$]filespec

Specifies the name of the task image file containing the task you want to install. The default file type is .TSK. The dollar sign ($), if present, directs the system to search for the file in APPL$DIR. If you do not include the /TASK_NAME qualifier, the system installs the task under a name based on the first six characters of the filename unless another name was assigned through the TASK= option of the Task Builder.

Prompts

File? filespec

Qualifiers

/FIX

Loads and locks a task into memory after installing it. P/OS can service requests for a fixed task much more quickly because it is memory-resident and does not have to be loaded from disk.

You can use the FIX command to fix a task that is already installed. See Section 3.28.

/NOREMOVE

When the DCL application exits, P/OS will not remove tasks or regions that have been installed with the /NOREMOVE qualifier.

/READONLY_COMMON

Specifies that a common or region is to be installed as read-only.

/TASK_NAME:taskname

Specifies the name by which the task is to be referenced. The default is set at link time. This qualifier overrides the link-time specification.

/[NO]WRITEBACK

Applies only when you install a read-write common. This qualifier controls whether or not the system writes the common back to its task image either when you remove the common from memory or when the system checkpoints the common.

If you use /WRITEBACK, then any changes made to the common while installed are copied to the common's task image when the common is checkpointed or you remove it.

If you specify /NOWRITEBACK, then changes made to the common are not copied back to the image. The default (that is, if you do not specify either), is /NOWRITEBACK.

3.34 LIBRARY

LIBRARY creates and maintains user-written library files, or libraries. Libraries can contain macro definitions, object modules, or, in the case of universal libraries, anything. LIBRARY invokes the Librarian Utility program (LBR). (Refer to Chapter 7.)

Syntax

LIBRARY[/operation][/qualifier[s]] libr_spec input_file

operation

Represents a specific subcommand form of LIBRARY (refer to each separately listed LIBRARY command).

qualifier

Refer to each separately listed LIBRARY command.

libr_spec

Specifies the name of the library. The default type is .OLB, specifying an object module library.

input_file

Specifies input files used as input for a new library file created with the /CREATE qualifier.

Prompts

Operation? operation

3.35 LIBRARY/COMPRESS

This command physically deletes modules that have been logically deleted through the LIBRARY/DELETE command. You can rename the resulting compressed library with this command.

Syntax

LIBRARY/**CO**MPRESS[:(arg[,s])] libr_spec [new_libr_spec]

arg

Can be one or more of the following:

GLOBAL:n
MODULES:n
BLOCKS:n

libr_spec

Is the name of the library to be compressed.

new_libr_spec

This optional parameter specifies a name for the newly compressed library. If you do not specify a name, the new, compressed file has the same name as the old one. The old file is not deleted after you create (compress) a new one.

Prompts

Library? libr_spec [new_libr_spec]

Arguments

If you are specifying more than one argument, the arguments must be enclosed in parentheses and separated by commas. If you are specifying only one argument, the parentheses are not necessary.

GLOBALS:n

The GLOBALS argument specifies the number of global symbols (entry point table entries) to allocate. The default n is the number of global symbols allocated in the old library. The maximum n is 4096. The value of n is always forced to zero for macro and universal libraries.

MODULES:n

The MODULES argument specifies the number of entries to allocate in the module name table. The default value is the number of entries in the old library. The maximum number of module names is 4096.

BLOCKS:n

The BLOCKS argument specifies the size of the library in 256-word blocks. The default size is the size of the old library.

3.36 LIBRARY/CREATE

LIBRARY/CREATE creates a library. This command can optionally insert one or more modules into the newly created library.

Syntax

```
LIBRARY/CREATE[:(arg[,s])][/qualifier[s]] libr_spec
                          [input_file[,s]]
```

arg

Can be one or more of the following:

```
GLOBALS:n
MODULES:n
BLOCKS:n
```

qualifier

Can be one or more of the following:

```
/[NO]GLOBALS
/MACRO
/OBJECT
/SQUEEZE
/UNIVERSAL
```

libr_spec

Is the name of the library to be created.

input_file

Specifies the file or files to be used as input to the new library file. If no input files are specified, an empty library file is created. The default file types are .OBJ when creating object module libraries, .MAC when creating macro libraries, and .UNI when creating universal libraries.

Prompts

```
Library?   libr_spec
Module(s)? input_file
```

Arguments

If you are specifying more than one argument, the arguments must be enclosed in parentheses and separated by commas. If you are specifying only a single argument, the parentheses are not necessary.

GLOBALS:n

The GLOBALS argument specifies the number of global symbols (entry point table entries) to allocate. The default is 512 for object libraries. The value n is always forced to 0 for macro and universal libraries; it can range from 0 through 4096.

MODULES:n

The MODULES argument specifies the number of entries to allocate in the module name table. The default value is 256. n can be from 0 through 4096.

BLOCKS:n

The BLOCKS argument specifies the size of the library in 256-word blocks. The default size is 100 blocks.

Qualifiers

/[NO]GLOBALS

/NOGLOBALS specifies that global symbols are not to be included in the entry point table.

Use this qualifier if you want to use the same global symbols in more than one module. /GLOBALS is the default but is a no-op.

/SQUEEZE

The /SQUEEZE qualifier reduces the size of macro definitions by eliminating all trailing blanks and tabs, blank lines, and comments from macro text. You must specify an input file or files.

This qualifier has meaning for macro libraries only.

Macros that have been squeezed not only take up less room in the macro library file, but also take up less memory.

This /SQUEEZE qualifier is the equivalent of the LBR /SZ qualifier applied to the output library file when using the /CR qualifier.

/MACRO

The /MACRO qualifier specifies that the library being created is a macro library.

/OBJECT

The /OBJECT qualifier specifies that the library being created is an object module library. This is the default and need not be specified.

/UNIVERSAL

The /UNIVERSAL qualifier specifies that the library being created is a universal library.

3.37 LIBRARY/DELETE

LIBRARY/DELETE deletes modules from a library. (Refer to LIBRARY/REMOVE to remove global symbols (entry points) from a library.)

Syntax

LIBRARY/**D**ELETE libr_spec module[,s]

libr_spec

Is the name of the library that contains modules for deletion.

module

Is the name of the module for deletion. (The LIBRARY/DELETE command will delete up to 15 modules.)

Prompts

```
Library? libr_spec
Module(s)? module
```

3.38 LIBRARY/EXTRACT

LIBRARY/EXTRACT reads one or more modules from a library and writes them to a specified output file. This command can extract as many as eight modules with each execution. If you specify more than one module to be extracted, LIBRARY/EXTRACT concatenates those modules in the output file.

Syntax

LIBRARY/**E**XTRACT[/qualifier] libr_spec modules[,s]

qualifier

/OUTPUT[:filespec]

libr_spec

Is the name of the library for the extract (read) operation.

module

Specifies the modules that are to be extracted. If you do not include a list, all modules in the library are extracted and concatenated in the output file in alphabetical order. You can specify up to eight modules, separated by commas.

Prompts

```
Library?   libr_spec
Module(s)? module
To?   filespec
```

Qualifier

/OUTPUT[:filespec]

The /OUTPUT qualifier specifies the file to which the extracted modules or macros are to be written. If you specify /OUTPUT without a filespec, the default is to write the modules to your terminal. This makes sense only for macro libraries or universal libraries containing text modules.

If you do not include the qualifier, you will be prompted To?, to which you are to reply with a filespec. You can reply TI: to have the output printed on your terminal.

3.39 LIBRARY/INSERT

This command inserts modules from one or more files into a library.

Syntax

LIBRARY/**I**NSERT[/qualifier[s]] libr_spec input_filespec[,s]

qualifier

Can be one or more of the following:

/[NO]GLOBALS
/SELECTIVE_SEARCH
/SQUEEZE

libr_spec

Is the name of the library in which modules will be inserted. Any number of files can be specified and each file can contain any number of concatenated modules.

input_filespec

Is the name of the file that contains the module or modules for insertion.

Prompts

```
Library?   libr_spec
Module(s)? module
```

Qualifiers

/[NO]GLOBALS

/NOGLOBALS specifies that entry points for the specified modules are not to be included in the entry point table.

Use this qualifier if you want to insert global symbols having the same name as symbols already in the library file. The default, /GLOBALS, does not permit this operation.

/SELECTIVE_SEARCH

Sets the selective search attribute bit in the module header of object modules as they are inserted into an object library. You must specify an input file or files.

Object modules with this attribute are given special treatment by the Task Builder. Global symbols defined in modules with the selective search attribute are not included in the Task Builder's symbol table unless they were previously referenced by other modules.

/SQUEEZE

Reduces the size of macro definitions by eliminating all trailing blanks and tabs, blank lines, and comments from macro text. You must specify an input file or files.

Macros that have been squeezed not only take up less room in the macro library file, but also take up less memory.

This qualifier is the equivalent of the LBR /SZ qualifier applied to the output file.

3.40 LIBRARY/LIST

This command lists the name of all modules which reside in a library. LIBRARY/LIST either displays these names on your terminal or writes them in an output file, depending on whether you supply a filespec or not.

Syntax

LIBRARY/**L**IST [:filespec][/qualifier] libr_spec

qualifier

Can be one or more of the following:

/BRIEF
/FULL
/[NO]NAMES

filespec

Is the name of the optional output file. If you do not include a filespec as an argument to /LIST, the library is listed on your terminal.

libr_spec

Is the name of the library that contains the modules to be listed.

Prompts

Library? libr_spec

Qualifiers

/BRIEF

The /BRIEF qualifier specifies that you want the list to include only the module names. This is the default.

/FULL

The /FULL qualifier requests a listing of all module names, along with a module description including size, date of insertion, and module-dependent information.

/[NO]NAMES

The /NAMES qualifier requests a list of modules in the library, along with their entry points. The default is /NONAMES.

3.41 LIBRARY/REMOVE

This command removes global symbols (entry points) from a library. (Refer to LIBRARY/DELETE to delete object modules from a library.)

Syntax

LIBRARY/**REM**OVE libr_spec global[,global[,s]]

libr_spec

Is the name of the library that contains global symbols to be deleted.

global

Is the name of a global symbol for deletion. You can specify as many as 15 global symbols.

Prompts

```
Library?   libr_spec
Symbol?    globals
```

3.42 LIBRARY/REPLACE

This command replaces a module in a library with a new module of the same name and deletes the old module. When a match occurs on a module name, the existing module is logically deleted and all its entries are removed from the global symbol table.

Syntax

LIBRARY/**REP**LACE[/qualifier[s]] libr_spec filespec[,s]

qualifier

Can be one or more of the following:

/[NO]GLOBALS
/SELECTIVE_SEARCH
/SQUEEZE

libr_spec

Is the name of the library that contains the module for replacement.

filespec

Is the name of the file or files that contain the new modules. If the module to be replaced does not exist in the library, LIBRARY performs an insert operation.

Prompts

```
Library?   libr_spec
Module(s)?  module
```

Qualifiers

/[NO]GLOBALS

/NOGLOBALS specifies that entry points for the specified modules are not to be included in the entry point table. The default is /GLOBALS.

/SELECTIVE_SEARCH

Sets the selective search attribute bit in the module header of object modules as they are inserted into an object library.

Object modules with the selective search attribute are given special treatment by the Task Builder. Global symbols defined in modules with the selective search attribute are only included in the Task Builder's symbol table if they were previously referenced by other modules.

/SQUEEZE

Reduces the size of macro definitions by eliminating all trailing blanks and tabs, blank lines, and comments from macro text.

Macros that have been squeezed not only take up less room in the macro library file, but also take up less memory in the assembler when they are invoked.

This is the equivalent of the LBR /SZ qualifier applied to the input file.

3.43 LINK

LINK invokes the Professional Application Builder (PAB), which links object modules and routines from user and system libraries to form an executable task.

Syntax

```
LINK[/qualifier[s]]
       filespec[/file-qualifier[s]][,filespec[,s]]
```

qualifier

Can be one or more of the following:

```
/ANCILLARY_PROCESSOR[:n]
/[NO]CHECKPOINT:arg
                SYSTEM
                TASK
/CODE:(arg[,s])
        EAE
        FPP
        PIC
        POSITION_INDEPENDENT
/COMPATIBLE
/[NO]CROSS_REFERENCE
/[NO]DEBUG[:filespec]
/ERROR_LIMIT:n
/[NO]EXECUTABLE[:filespec]
/FAST_MAP
/FULL_SEARCH
/[NO]HEADER
/[NO]IO_PAGE
/LONG
/MAP[:options-spec]
/[NO]MEMORY_MANAGEMENT[:n]
/OPTIONS[:options-spec]
/[NO]PRIVILEGED[:n]
/[NO]RECEIVE
/[NO]RESIDENT_OVERLAYS
/SAVE
/[NO]SEGREGATE
/SEQUENTIAL
/SHAREABLE[:arg]
            COMMON
            LIBRARY
            TASK
```

```
/SLAVE
/SYMBOL_TABLE[:filespec]
/[NO]SYSTEM_LIBRARY_DISPLAY
/[NO]TASK[:filespec]
/TKB
/TRACE
/[NO]WARNINGS
/WIDE
```

filespec

Is the name of the file containing the object module.

file-qualifier

```
/[NO]CONCATENATE
/DEFAULT_LIBRARY
/[NO]GLOBALS
/LIBRARY
/INCLUDE:(module1[:...:modulen])
/SELECTIVE_SEARCH
/OVERLAY_DESCRIPTION
```

Prompts

File(s)? filespec[,s]

Qualifiers

/ANCILLARY_PROCESSOR[:n]

Identifies the task as an Ancillary Control Processor (ACP). The parameter n specifies the base relocation register. Acceptable values are 0, 4, or 5. The default is 5.

In TKB format, apply the /AC qualifier to the .TSK filespec.

/[NO]CHECKPOINT:arg

SYSTEM
TASK

Specifies that the task is to be (or is not to be) checkpointable. Checkpointability of tasks is an important part of the operating system's ability to share resources. When a higher priority task seeks access to system memory, a checkpointable task of lower priority is checkpointed, or rolled out to the disk to be stored in its current state until the higher priority task exits, whereupon the lower priority task returns and takes up where it left off.

If you do not use the /CHECKPOINT qualifier, your task is built noncheckpointable. The default is /NOCHECKPOINT. A noncheckpointable task cannot be dislodged by a task of higher priority. Therefore, you should always build your tasks checkpointable unless you have some important reason for not doing so.

You can specify how the checkpointing of the task is handled through the arguments to the /CHECKPOINT qualifier. Your task can be checkpointable to the system checkpoint file with the SYSTEM option. This is the default. LINK/CHECKPOINT and LINK/CHECKPOINT:SYSTEM are equivalent commands.

Tasks built with system checkpointing cannot be checkpointed if the system checkpoint file is full.

You can also reserve checkpoint space for the task as part of its own task image file by using LINK/CHECKPOINT:TASK. Such tasks are always checkpointable. (If there is no system checkpoint file, you can only run one copy of such tasks.)

It should be apparent that the checkpointability of a task has an impact on the efficient operation of the entire system as well as on the task itself. If the task is built noncheckpointable, it can block more important tasks from running. If it is built with task checkpointability, mass-storage space is reserved that may never be used. If it is built with system checkpointability, there may be no room for it in the system checkpoint file.

In TKB format, for /CHECKPOINT:SYSTEM, apply the /CP qualifier to the task image file. For /CHECKPOINT:TASK, apply the /AL qualifier to the task image file. For /NOCHECKPOINT, apply the /-CP qualifier to the task image file.

/CODE:(arg[,s])
EAE
FPP
PIC
POSITION_INDEPENDENT

The /CODE qualifier specifies that the code for the task relies on certain hardware elements or employs certain techniques. See the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual* for more information.

The EAE argument specifies that the task uses the Extended Arithmetic Element. In TKB format, apply the /EA qualifier to the .TSK filespec.

The FPP argument specifies that the task uses the Floating Point Processor. This hardware is optional and may not be part of your system. In TKB format, apply the /FP qualifier to the .TSK filespec.

The PIC and POSITION_INDEPENDENT arguments are identical and specify that the resident common or library being built is position-independent. In TKB format, apply the /PI qualifier to the .TSK or .STB filespec.

If you want to use more than one argument, enclose them in parentheses, separated by commas.

/COMPATIBLE

Specifies that the task be built in compatibility mode. This means that memory-resident overlay segments are aligned on 256-word boundaries for compatibility with other implementations of the mapping directives. Without this qualifier, overlay segments are aligned on 32-word boundaries.

In TKB format, apply the /CM qualifier to the .TSK filespec.

/[NO]CROSS_REFERENCE

Specifies that a listing of symbol cross-references is to be appended to the Task Builder map file.

If you include this qualifier, you automatically include the /MAP qualifier as well. You do not need the /MAP qualifier unless you want to supply a name for the map file. If you supply a name, the map file appears in your current directory.

In TKB format, apply the /CR qualifier to the .MAP filespec.

/[NO]DEBUG[:filespec]

Specifies the inclusion of a debugging aid in the task image. A debugging aid permits you to interrupt the running of a task and inspect registers and other memory locations at various stages. If you give no filespec, the default is LB0:[1,5]ODT.OBJ, which is ODT, the On-line Debugging Tool, a DIGITAL-supplied utility. ODT is incorporated in the task you are building. ODT can be used only on tasks written in the MACRO-11 Assembly Language.

See the *IAS/RSX-11 ODT Reference Manual* for further information.

If you have a user-written debugger, name the file it is in when you use this qualifier. This file should be an object module.

In TKB format, apply the /DA qualifier to the .TSK filespec if you want to use ODT. If you want to use a user-written debugger, apply the /DA qualifier to the input filespec naming the debugger.

/ERROR_LIMIT:n

Directs the Task Builder to abort LINK after n diagnostics errors have been produced.

In TKB format, apply the /XT:n qualifier to the .TSK filespec.

/[NO]EXECUTABLE[:filespec]

See /[NO]TASK.

/FAST_MAP

Use this qualifier if you want P/OS to interpret IOT instructions in your task as requests to perform fast remap operations. When you use the /FAST_MAP qualifier to link your task, P/OS interprets all IOT instructions as fast remap requests. See the *P/OS System Reference Manual* for details on fast remap operations.

In TKB format, apply the /FM qualifier to the .TSK filespec.

/FULL_SEARCH

Specifies that when processing modules from the default object module library, the Task Builder should search all co-tree overlay segments for matching definitions or references.

Without this qualifier, unintended global references between co-tree overlay segments are eliminated. Definitions of global symbols from the default library are restricted in scope to references in the main root and the current tree.

In TKB format, apply the /FU qualifier to the .TSK filespec.

/LONG

Specifies that the map file produced by the Task Builder include additional file information on modules used in the task build. The long map does not include file information on modules from the system library.

If you include this qualifier, you automatically include the /MAP qualifier as well. You do not need the /MAP qualifier unless you want to supply a name for the map file. If you supply a name, the map file appears in your current directory.

In TKB format, apply the /-SH qualifier to the .MAP filespec.

/MAP[:options-spec]

Specifies that you want to produce a memory allocation, or map file. If you use /MAP as a command qualifier, without a options-spec argument, the map appears in your directory; its filename is derived from the name of the first input file named in the command line. Its file type is .MAP.

If you use /MAP with a options-spec argument, either as a command qualifier or a filespec qualifier, the map appears in your directory with a filename you have given.

The following qualifiers automatically direct the Task Builder to produce a map with special characteristics:

/CROSS_REFERENCE
/LONG
/SYSTEM_LIBRARY_DISPLAY
/WIDE

You do not need the /MAP qualifier with these qualifiers unless you want to add a filespec argument to /MAP. These other map-related qualifiers can be used as either command or filespec qualifiers, but they have no affect on filenames.

In TKB format, the /MAP qualifier corresponds to the second position in the list of TKB output files and has the default file type .MAP.

/[NO]HEADER

/NOHEADER sets the Task Builder STACK option to 0. If you are building a shared region or a driver, you should specify this qualifier. If you specify this qualifier, you cannot use the STACK option. The default is /HEADER.

In TKB format, apply the /-HD qualifier to either the .TSK or .STB filespec and specify the STACK=0 option. See /SHAREABLE.

/[NO]IO_PAGE

/NOIO_PAGE indicates to the Task Builder that the task is over 12K and purposely does not map to the I/O page. This qualifier is for building privileged tasks only. The default is /IO_PAGE.

/[NO]MEMORY_MANAGEMENT[:n]

Specifies that the task is being built for a system with (or without) memory management hardware. Use the /NOMEMORY_MANAGEMENT qualifier when you are building a task on a system with memory management to be run on an RSX-11M (or RSX-11S) system without the memory management hardware. The value n specifies the highest physical address of a task on the target system; it can be 28 (the default) or 30. The default is /MEMORY_MANAGEMENT.

In TKB format, apply the /-MM qualifier to the .TSK filespec.

/SHAREABLE[:arg]

COMMON
LIBRARY
TASK

/SHAREABLE:TASK is the default value and identifies the task as a multiuser task. Such tasks permit more than one user to share the read-only partition of a single task. When you specify /SHAREABLE:TASK, the Task Builder divides the task into two regions: region 0 contains the read-write portion of the task and region 1 contains the read-only portion of the task. When multiuser tasks are installed, multiple requests for the task to run cause the system to duplicate only the read-write portion of the task for each request after the first. The ROPAR option permits you to name the portion in which region 1 is to reside. In TKB format, apply the /MU qualifier to the .TSK filespec.

/SHAREABLE:COMMON informs the Task Builder that a shareable common is being built. You should always use the /NOHEADER qualifier with /SHAREABLE:COMMON. If you do not specify CODE:PIC or CODE:POSITION_INDEPENDENT, TKB builds an absolute shared common. All program sections in the common are marked absolute. If you specify CODE:PIC or CODE:POSITION_INDEPENDENT, all program sections in the common are marked relocatable. In either case, the .STB file contains all the program section names, attributes, lengths, and symbols. The .STB file of a common built /SHAREABLE contains all defined program sections. In TKB format, apply the /CO qualifier to the .TSK or .STB filespec. /SHAREABLE:TASK defaults to /SEGREGATE.

/SHAREABLE:LIBRARY informs the Task Builder that a shareable library is being built. You should always use /NOHEADER with /SHAREABLE:LIBRARY. TKB includes only one program section in the .STB file. If you do not specify CODE:PIC or CODE:POSITION_INDEPENDENT, TKB names the program section .ABS, makes the library position-dependent, and defines all symbols as absolute. If you specify CODE:PIC or CODE:POSITION_INDEPENDENT, TKB gives the program section the same name as the root segment of the library. TKB forces this name to be the first and only declared program section in the library. TKB declares all global symbols in the .STB file relative to that program section. In TKB format, apply the /LI qualifier to the .TSK or .STB filespec.

/OPTIONS[:options-spec]

Specifies one or more Task Builder options. Use this qualifier if you need to use any of the Task Builder options. For full information, on options, see the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual*.

You can supply options for this qualifier in two ways: supply the filespec of a file containing the options, or specify the options themselves. If you do not include a filespec with this qualifier, you are prompted for options. If you want to enter multiple options, you use a comma after each option listed. To list multiple options on individual lines, end each line with a comma.

If you include a filespec with this qualifier, this file should contain only the option names, comments, and the arguments associated with the options, as shown:

```
UNITS=8
ASG=TT0:7:8
; COMMENTS, PRECEDED BY SEMICOLON, ALLOWED
PAR=KROBAR:50000:40000
```

This file should only the option statements and comments preceded by the semicolon (;). Comments preceded by the exclamation point (!) are not accepted in this instance. Do not put any slashes in the file.

Note that the ABORT option is the only means of preventing a task build once you have begun issuing commands to the Task Builder. A CTRL/Z simply directs the Task Builder to begin the task build based on whatever instructions you have issued up to that point. Thus, if you want to be sure of being able to stop a task build in this way, you must include the /OPTIONS qualifier in the LINK command line. This assures the availability of the ABORT option. If you are typing the command line and make a mistake, specify the /OPTIONS qualifier, press RETURN, and enter the ABORT=0 option. Then press RETURN and the task build aborts.

/PRIVILEGED[:n]

Indicates that task is privileged. The argument n specifies the base relocation register. Acceptable values are 0, 4, or 5. The default is 5.

In TKB format, apply the /PR qualifier to the .TSK filespec.

/[NO]RECEIVE

Indicates that the task may (or may not) receive messages by means of the Executive directive SEND. /RECEIVE is the default.

In TKB format, apply the /SE qualifier to the .TSK filespec.

/[NO]RESIDENT_OVERLAY

Enables (or disables) recognition of the memory-resident overlay operator (!) in an overlay description file. The qualifier is used with the /OVERLAY_DESCRIPTION qualifier when the task has memory-resident overlays. The default is /RESIDENT_OVERLAY.

In TKB format, apply the /RO qualifier to the .TSK filespec.

/SAVE

The /SAVE qualifier specifies that you want to retain the indirect command file created by DCL to pass your LINK command to TKB. If you include /SAVE in your LINK command line, a file named ATLNK.TMP appears in your directory after the task build completes. Since this file always has the same name, you should give it a name related to the task it

builds, such as SHEMPBLD.CMD. Then you can issue a command in the following form and duplicate the task build that originally produced the command file:

```
>LINK @SHEMPBLD
```

This file is also useful for comparing LINK command syntax with TKB syntax because it includes the full translation of the LINK command into TKB format.

/[NO]SEGREGATE

/SEGREGATE causes the Task Builder to order program sections alphabetically by name within access code (RO followed by RW). If you also specify /SEQUENTIAL, TKB orders program sections in their input order by access code.

/NOSEGREGATE is the default. TKB interleaves RO and RW program sections. When combined with /SEQUENTIAL, /NOSEGREGATE results in a task with program sections allocated in input order with its RW and RO sections interleaved. If you use /NOSEQUENTIAL and /NOSEGREGATE together, which is the default for both, TKB orders program sections alphabetically with RW and RO sections interleaved.

In TKB format, use the /[-]SG qualifier on the .TSK filespec.

/SHAREABLE:TASK defaults to /SEGREGATE.

/SEQUENTIAL

Directs Task Builder to construct a task image from program sections in the order in which they appear. Normally, the Task Builder finds all program sections referenced in all modules in an overlay segment and then builds the task with those program sections in alphabetical order. Do not use this qualifier to build tasks that rely on alphabetical allocation of program sections, such as FORTRAN I/O handling modules and FCS modules from SYSLIB. See also /SEGREGATE.

In TKB format, apply the /SQ qualifier to the .TSK filespec.

/SLAVE

Specifies that the task is to be slaved to a sending task. When a slaved task successfully executes the Executive directive Receive Data, it is given the [UIC] and TI: of the sending task. This qualifier applies only to systems with multiuser protection.

Slaved tasks cannot be run with a RUN command. They must be run by the sending task.

In TKB format, apply the /SL qualifier to the .TSK filespec.

/SYMBOL_TABLE[:filespec]

Directs that a symbol table file be produced. The default name is that of the first input file and the default type is .STB. The options-spec parameter overrides the defaults. This qualifier is used when building shared regions.

In TKB format, this qualifier corresponds to the third position in the list of TKB output files, called the .STB filespec.

/[NO]SYSTEM_LIBRARY_DISPLAY

Directs the Task Builder to produce a map that includes (or does not include) global symbols defined or referenced by the task. These symbols are found in LB0:[1,5]SYSLIB or in any shared regions linked to using TKB options. This map is usually considerably longer than the default map. The information displayed illuminates the contribution that SYSLIB or the shared regions make to the task.

If you include this qualifier, you include the /MAP qualifier as well. You do not need the /MAP qualifier unless you want to supply a name for the map file.

If you supply a name, the map file appears in your current directory. See the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual* for more information.

In TKB format, apply the /MA qualifier to the .MAP filespec.

/TASK[:taskspec]
/EXECUTABLE[:taskspec]

Specifies a name for the task image file different from that of the first input file plus the type .TSK. If used as a options-spec qualifier, the task name is derived from the name of the file to which the qualifier is attached. /EXECUTABLE is a synonym.

In TKB format, this qualifier corresponds to the first position in the list of TKB output files, called the .TSK filespec.

/NOTASK
/NOEXECUTABLE

Specifies that LINK produce no task image file. This qualifier is useful when you want to use some facility of the Task Builder without building a task, to check for unresolved symbol references or make a map, for instance. /NOEXECUTABLE is a synonym.

In TKB format, leave the first position in the list of TKB output filespecs blank, followed by a comma (,).

/TKB

Specifies that the default Task Builder be used to build the task. This is the default; the qualifier is included for completeness.

You can also invoke the default Task Builder from DCL with the command RUN $PAB. You must follow TKB format if you run the default Task Builder in this fashion. See the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual* for a description of the TKB command format.

/TRACE

Specifies that the task is to be traceable. When you use this qualifier, a trace trap occurs on the completion of each instruction when the task is run.

In TKB format, apply the /TR qualifier to the .TSK filespec.

/[NO]WARNINGS

/NOWARNINGS suppresses diagnostic messages issued by the the Task Builder. Two messages are suppressed:

n undefined symbols segment "segname"

and

Module "modulename" multiply defines P-section "psectname"

The default is /WARNINGS.

In TKB format, use the /[-]NM qualifier on the .TSK filespec.

/[NO]WIDE

Specifies that the Task Builder map be printed in 132-column format. The default is /NOWIDE.

If you include this qualifier, you include the /MAP qualifier as well. You do not need the /MAP qualifier unless you want to supply a name for the map file. If you supply a name, the map file appears in your current directory.

In TKB format, apply the /WI qualifier to the .MAP filespec.

Parameter Qualifiers

Any input file can have a parameter qualifier applied to it, identifying the kind of file that it is.

/[NO]CONCATENATE

Identifies the input file as a concatenated object file; this is the default. All modules in the file are processed to form the task image. /NOCONCATENATE specifies that only the first object module encountered is to be processed, regardless of how many are present.

In TKB format, apply the /CC qualifier to an input file containing concatenated object modules.

/LIBRARY

Identifies the file as an object module library. This qualifier is required for any input library file and is prohibited for any other type of file. The default file type for object libraries is .OLB.

The Task Builder searches the library file to resolve all undefined global symbol references from files appearing to the left of the library file in the LINK command line. The Task Builder then extracts any and all modules that resolve undefined references and includes them in the task image. See also /INCLUDE.

/INCLUDE:module1[:...modulen]

You can specify as many as eight module names from a library using /INCLUDE. You must specify at least one. If you use the optional module arguments, the Task Builder takes only those modules from the library. The module names are defined at assembly time.

If you want both to resolve undefined references to global symbols and to specify modules, you must use this qualifier twice.

In TKB format, apply the /LB qualifier to an input library file for both /LIBRARY and /INCLUDE.

/DEFAULT_LIBRARY

Specifies that the file to which it is appended replace the system object module library, LB0:[1,5]SYSLIB.OLB, as the default library that is searched (in order to resolve unresolved global references). This qualifier can be applied to only one file; that file must be an object module library having the file type .OLB.

In TKB format, apply the /DL qualifier to an input library file.

/[NO]GLOBALS

Specifies that global symbols referenced and defined by the input file are (or are not) to be included in the map output file. The default is /GLOBALS.

In TKB format, apply the /-MA qualifier to the input file.

/SELECTIVE_SEARCH

Instructs the Task Builder to search the file only for undefined references to global symbols. This qualifier is most useful when building an Ancillary Control Processor or other privileged task that maps into the Executive. If you do not specify this qualifier, all of the Executive's global symbol definitions are included in the task build, whether or not the file contains undefined references to the global symbol. The Executive contains a myriad of modules. In these and similar circumstances, this qualifier considerably shortens the symbol table search and improves system performance.

If you do not use this qualifier, all global symbols from the input file are included in the task image.

In TKB format, apply the /-SS qualifier to an input file.

/OVERLAY_DESCRIPTION

Specifies that the input file is an overlay description file (type .ODL) that controls the linking of the task. No other input file can be specified if you use this qualifier. The .ODL file specifies input files to LINK. /OVERLAY_DESCRIPTION can be either a command qualifier or a filespec qualifier.

In TKB format, use the .ODL file as the only input file to the right of the equal sign (=) with the /MP qualifier applied to the .ODL filespec.

3.44 LINK/C81

This section explains how you can link COBOL-81 object files to produce a task image (.TSK file).

Syntax

LINK/**C81**[/qualifier[s]] filespec[,s]

LINK/**COB**OL[/qualifier[s]] filespec[,s]

qualifier[s]

Although the format shows them as command qualifiers, you can append the following qualifiers to either the command or a file specification. When you use the LINK/C81 command, the two locations are equivalent.

```
/[NO]FMS
/FMS:NORESIDENT
/OTS:[NO]RESIDENT
/[NO]RMS:[NO]RESIDENT
/[NO]MAP
/[NO]DEBUG
```

filespec[,s]

Specifies the file or files to be linked. The default file type is .SKL.

The LINK/C81 command links object modules indirectly through their associated .SKL files. Therefore, if you include file types in any file specifications, you must use .SKL rather than .OBJ. Each .SKL file corresponds to an object module (.OBJ file) that is included in the task image. Except for the file type, the .SKL file and object module have identical file specifications.

You can list any number of .SKL files as input files in any order, separated by commas.

Prompts

```
File(s)? filespec[,s]
```

Qualifiers

/OTS:[NO]RESIDENT

The /OTS:RESIDENT qualifier includes memory-resident OTS in your task image. When you specify /RMS:RESIDENT as well as /OTS:RESIDENT, LINK/C81 clusters these two libraries by default. If you can take advantage of the clustering feature, the resulting task image is smaller and program execution speed is improved. You should use /OTS:RESIDENT, because the Professional 300 Series only supports memory-resident OTS and RMS.

The /OTS:NORESIDENT qualifier includes the disk-resident OTS library in your task image.

The default is /OTS:NORESIDENT.

/[NO]RMS:[NO]RESIDENT

The /RMS:RESIDENT qualifier creates a reference to the shared RMS-11 memory-resident library, RMSRES. This library includes input/output support for sequential, indexed, and relative file organizations.

When you specify /OTS:RESIDENT as well as /RMS:RESIDENT, LINK/C81 clusters these two libraries by default. If you can take advantage of the clustering feature, the resulting task image is smaller and program execution speed is improved.

The /NORMS qualifier is equivalent to /RMS:NORESIDENT.

/[NO]MAP

The /MAP qualifier causes LINK/C81 to produce a Task Builder map file with the file type .MAP. The /NOMAP qualifier tells LINK/C81 not to produce a memory map file.

The default is /NOMAP.

/[NO]DEBUG

The /DEBUG qualifier tells LINK/C81 to include the COBOL-81 Symbolic Debugger in your task image. To use this qualifier, you must also use the /DEBUG qualifier to the COBOL command. The /NODEBUG qualifier tells LINK/C81 not to include the COBOL-81 Symbolic Debugger in your task image.

The default is /NODEBUG.

/[NO]FMS
/FMS:NORESIDENT

The /FMS qualifier causes LINK/C81 to include Forms Management Services (FMS) library support in your task image. You must use this qualifier if you call FMS routines from your program.

The /NOFMS qualifier tells LINK/C81 not to include FMS support.

The default is /NOFMS.

The /FMS:NORESIDENT qualifier causes LINK/C81 to include support for a non-memory-resident FMS library in your task image. Since no resident FMS libraries are supported under P/OS, this qualifier is equivalent to /FMS.

For more detailed information about how to create a task image using LINK/C81, refer to the *COBOL-81 RSX-11M/M-PLUS User's Guide*.

3.45 LOAD

The LOAD command loads a device driver in the P/OS Operating System. Both the driver task image and symbol table file must reside in LB:[ZZSYS]. The filenames must be xxDRV.TSK and xxDRV.STB respectively, where "xx" is the logical device name mnemonic.

Syntax

LOAD device_mnemonic

filespec

Is the logical device name associated with the device driver.

Prompts

Device mnemonic? device_mnemonic

3.46 MACRO

MACRO invokes the Professional MACRO-11 Assembler (PMA) to assemble one or more MACRO-11 assembly language programs into a single relocatable object module suitable for processing by the Professional Application Builder (PAB).

Syntax

```
MACRO[/command-qual[s]]
       filespec[/file-qual[s]][,filespec[,s]]
```

command-qual

Can be one or more of the following:

```
/[NO]CROSS_REFERENCE
/DISABLE:(arg[,s])
           ABSOLUTE
           BINARY
           CARD_FORMAT
           GLOBAL
           LOCAL
           LOWERCASE
           REGISTER_DEFINITIONS
           TRUNCATION
/ENABLE:(arg[,s])
           ABSOLUTE
           BINARY
           CARD_FORMAT
           GLOBAL
           LOCAL
           LOWERCASE
           REGISTER_DEFINITIONS
           TRUNCATION
/[NO]LIST[:filespec]
/[NO]OBJECT[:filespec]
/[NO]SHOW[:(arg[,s])]
           ALL
           BINARY
           CALLS
           COMMENTS
           CONDITIONALS
           CONTENTS
           COUNTER
           DEFINITIONS
           EXPANSIONS
           EXTENSIONS
           LISTING_DIRECTIVES
           OBJECT_BINARY
           SEQUENCE_NUMBERS
```

SOURCE
SYMBOLS
/[NO]WIDE

filespec

Specifies one or more input files for the MACRO-11 Assembler. These input files must contain MACRO-11 source code. Multiple filespecs must be separated by commas. Filespecs must include a filename. If no file type is given, the default file type (.MAC) is applied. If the parameter qualifier /LIBRARY is used, .MLB is the default file type. No wildcards are accepted by MACRO.

file-qual

Can be one or more of the following:

/LIBRARY
/PASS:n

Prompts

File(s)? filespec[,s]

Command Qualifiers

/[NO]LIST[:filespec]

Specifies whether an assembly listing should be generated. The default is /NOLIST, meaning no assembly listing is generated.

If you do not supply a file specification for this qualifier, the listing has a filename derived from the name of the last source file in the MACRO command, with the file type .LST. If you want the listing to have a different name, supply the name as an argument to the /LIST qualifier. The listing file appears in your default directory.

If your command line includes the /CROSS_REFERENCE qualifier, /LIST is implied and need not be specified.

If your command line includes listing-control arguments to either /SHOW or /NOSHOW, /LIST is implied and need not be specified.

The only time you need to use /LIST with /CROSS_REFERENCE or /[NO]SHOW is when you want to give the listing file a filespec other than the default.

/[NO]OBJECT[:filespec]

Specifies whether an object module should be generated. The default is /OBJECT, meaning an object module is generated. If you do not supply a file specification, the object file has a name derived from the name of the last source file and the file type .OBJ. If you want the object file to have a different name, give the name as an argument to the /OBJECT qualifier.

You can name the object file after any of the source files listed in the MACRO command by using /OBJECT as a filespec qualifier. If used as a filespec qualifier, /OBJECT cannot take a filespec argument.

The qualifier /NOOBJECT specifies that no object module is generated. You can use this qualifier if you want to use other facilities of the assembler, to get an assembly listing, for instance, without creating an object module.

/[NO]CROSS_REFERENCE

Specifies whether a cross-reference listing should be generated and appended to the assembly listing. The default is /NOCROSS_REFERENCE.

The cross-reference listing locates all user-defined and MACRO symbols that appear in the source program.

When you specify this qualifier, you are also specifying the /LIST qualifier by implication. An assembly listing is generated in your directory. If you want to change the name of the listing file, you must use /LIST with a file specification in addition to /CROSS_REFERENCE.

/ENABLE:(arg[,s])
/DISABLE:(arg[,s])

These qualifiers override the .ENABL and .DSABL assembler directives included in the source program being assembled. The .ENABL and .DSABL directives invoke or inhibit various aspects of the assembly. Table 3-2 summarizes the arguments to /ENABLE and /DISABLE and gives their MACRO-11 equivalents. There is a default setting for each of these directives, even if you do not specify them in your code or command line.

These qualifiers affect the entire assembly. If, for example, your MACRO command includes the qualifier /ENABLE:LOWER_CASE, the assembler does not convert any lowercase source text to uppercase, regardless of any .DSABL

LC or .ENABL LC directives in the source code. The same goes for /DISABLE:LOWER_CASE. All .ENABL or .DISABL LC directives are ignored.

If you specify only one argument to /ENABLE or /DISABLE, you need not include the parentheses, but if you have more than one argument, they must be separated by commas and enclosed in parentheses.

Table 3-2: The Enable and Disable Qualifiers

Function	MACRO Syntax	Description
Assembly Functions Disabled by Default		
ABSOLUTE	AMA	Enabling this function causes relative mode addresses (mode 67) to be assembled as absolute addresses (mode 37).
BINARY	ABS	Enabling this function produces absolute binary output in FILES-11 format.
CARD_FORMAT	CDR	Enabling this function causes source columns 73 and greater to be treated as a comment.
LOCAL	LSB	Enabling this function permits the disabling or enabling of a local symbol block.
LOWERCASE	LC	Enabling this function causes MACRO-11 to accept lowercase ASCII input. The default is to convert it to uppercase.

Function	MACRO Syntax	Description
TRUNCATION	FPT	Enabling this function causes floating-point truncation. The default is floating-point rounding.
Assembly Functions Enabled by Default		
REGISTER_ DEFINITIONS	REG	Disabling this function inhibits the normal MACRO-11 default register definitions. The default is: R0=%0, R1=%1...SP=%6,PC=%7 Under most circumstances, you should use these defaults.
GLOBAL	GBL	Disabling this function causes MACRO-11 to treat all symbol references that are undefined at the end of assembly pass 1 as undefined symbols. The default is to treat all such symbols as global symbols.

/[NO]SHOW[:(arg[,s])]

These qualifiers override the .LIST and .NLIST assembler directives included in the source program being assembled. The .LIST and .NLIST directives control the content and format of the assembly listing. Table 3-3 summarizes the arguments to /SHOW and /NOSHOW and gives their MACRO-11 equivalents. There is a default setting for each of these directives, even if you do not specify them in your code or command line.

These qualifiers affect the entire assembly. If, for example, your MACRO command includes the qualifier /SHOW:COMMENTS, the assembly listing includes all comments, regardless of any .NLIST COM or .LIST COM directives in the source code. The same goes for /NOSHOW:COMMENTS. All .LIST COM or .DISABL COM directives are ignored.

If you specify only one argument to /SHOW or /NOSHOW, you need not include the parentheses, but if you have more than one argument, they must be separated by commas and enclosed in parentheses.

/SHOW implies /LIST, but if you want the listing file to have a name other than the default, you must still use /LIST.

Table 3-3: The /SHOW and /NOSHOW Qualifiers

Function	MACRO Syntax	Description
Listing Functions Disabled by Default		
EXPANSIONS	ME	Enabling this function causes MACRO-11 to include all macro expansions in the listing.
BINARY	MEB	Enabling this function causes MACRO-11 to list only those macro expansions that generate binary code. This is a subset of EXPANSIONS.
LISTING__DIRECTIVES	LD	Enabling this function causes MACRO-11 to list all listing control directives without arguments (these are the listing directives that alter the listing level count).
SEQUENCE__NUMBERS	SEQ	Disabling this function suppresses the inclusion of sequence numbers in the listing. Sequence numbers are replaced by tabs.
COUNTER	LOC	Disabling this function suppresses the location counter field and does not replace it with a tab.
OBJECT__BINARY	BIN	Disabling this function suppresses the listing of generated binary code and does not replace it with a tab.

Function	MACRO Syntax	Description
EXTENSIONS	BEX	Disabling this function suppresses the listing of binary extensions, that is, all binary code that will not fit on the first line. This is a subset of OBJECT__BINARY.
SOURCE	SRC	Disabling this function suppresses the listing of source lines.
COMMENTS	COM	Disabling this function suppresses the listing of comments. This is a subset of SOURCE.
DEFINITIONS	MD	Disabling this function suppresses the listing of macro definitions and repeat range expansions.
CALLS	MC	Disabling this function suppresses the listing of macro calls and repeat range expansions.
CONDITIONALS	CND	Disabling this function suppresses the listing of unsatisfied conditional coding.
CONTENTS	TOC	Disabling this function suppresses the listing of table of contents during assembly pass 1. The full assembly listing is still prepared during assembly pass 2.
SYMBOLS	SYM	Disabling this function suppresses the listing of the symbol table resulting from the assembly.

/[NO]WIDE

Specifies whether you want the assembly listing in wide or narrow format. As supplied, the default is /WIDE, also called line printer format. /NOWIDE is sometimes called teleprinter format.

This qualifier overrides any .LIST TTM or .NLIST TTM directives included in your source program.

Parameter Qualifiers

/PASS:n

Specifies that the file thus qualified is only to be assembled during the pass specified. The assembler makes two passes; n can be either 1 or 2.

/LIBRARY

Specifies that the file thus qualified is a macro library. The default file type is .MLB. A user macro library file must be specified in the command line before any source files that use the macros defined in the library. A library may not be the last file named in the command line. Remember that the system macro library has the type .SML. If you are referencing this library, you must explicitly state the type.

3.47 MAIL

With PRO/DECnet only, the MAIL command invokes the PRO/DECnet Mail Utility. This utility allows you to send or read messages to or from other users on your system, or to another system connected by PRO/DECnet. For a complete description of the Mail Utility see the *PRO/DECnet User's Guide*.

Syntax

MAIL

Prompts

None

Qualifiers

None

3.48 MOUNT

This command declares a volume (and therefore the files or data it contains) to be logically known to the system, on-line, and available for use.

Syntax

MOUNT[/qualifier] device: [volumelabel]

qualifer

Can be any combination of the following:

/FOREIGN
/SHOW

device:

Is the device on which you want to mount the volume.

volumelabel

Is the name of the volume. If the volume name does not match the volumelabel parameter, then DCL displays an error message and does not mount the device.

Prompts

Device? device: [volumelabel]

Qualifiers

/FOREIGN

Specifies that the volume being mounted is not in standard P/OS (FILES-11) format.

/SHOW

Displays the volume label after mounting the volume.

3.49 PASCAL

This command invokes the PRO/Tool Kit PASCAL compiler to compile one or more PASCAL source programs.

NOTE

Please refer to the language documentation for additional information.

Syntax

PASCAL[/qualifier[s]] filespec[,s]

qualifier

Can be one or more of the following:

```
/[NO]CHECK[:arg]
          ALL
          BOUNDS
          SUBRANGE
          POINTER
          STACK
          DIVIDE
          CASE
/CODE:arg
          FPP
          EIS
/[NO]DEBUG:TRACE_BACK
/[NO]LIST[:filespec]
/[NO]MACHINE_CODE[:filespec]
/[NO]OBJECT[:filespec]
/[NO]STANDARD
/[NO]XRAY
/[NO]LOG
```

filespec

Is the name of a PASCAL source file.

Prompts

```
Files?   filespec[,s]
```

Qualifiers

/[NO]CHECK[:arg]

The /CHECK option controls run-time checking.

Arguments:

ALL	enables all values (default)
BOUNDS	enables array bounds checking
CASE	enables CASE statement expression checking
DIVIDE	enables divide by zero checking
POINTER	enables NIL pointer checking
STACK	enables stack overflow checking
SUBRANGE	enables subrange checking

/CODE:arg

The /CODE option controls whether the code generated by Tool Kit PASCAL uses FPP instructions to perform floating point operations or EIS instructions to simulate floating point operations with subroutine calls.

Arguments:

FPP	generates FPP instructions (default)
EIS	generates EIS instructions

/[NO]DEBUG:TRACE_BACK

The /DEBUG option controls whether Tool Kit PASCAL generates code that tracks the relationship between the lines in the program source code and the current position in the executing program. When /DEBUG is enabled, run-time error messages indicate the line which caused the error and the currently active procedure and function invocations.

/[NO]LIST[:filespec]

The /LIST[:filespec] option controls whether Tool Kit PASCAL generates a program listing file. The default is /NOLIST; thus compiler error messages appear on your terminal. If you specify /LIST and do not supply a file specification, the default is file.LST, where file is the name of the first input file.

/[NO]MACHINE_CODE[:filespec]

The /MACHINE_CODE option controls whether Tool Kit PASCAL generates a MACRO-11 source file containing the assembly language code for the program. The default is /NOMACHINE_CODE.

The default name of the object file created by PASCAL is the last-named source file with the file type .MAC. If you want the object file to have a different name, supply the name as an argument to the qualifier. You can examine or assemble this file.

Also, you must not specify the /OBJECT and /MACHINE_CODE qualifiers together on the same command line.

/[NO]OBJECT[:filespec]

Determines whether or not an object module is generated by the compiler. The default is /OBJECT, which does create a module. The default name of the object file created by PASCAL is the last-named source file with the file type .OBJ. If you want the object file to have a different name, supply the name as an argument to the /OBJECT qualifier.

/NOOBJECT specifies that no object module is created. You can use the /NOOBJECT qualifier to get a compiler listing file to check for errors without generating object code.

You must not specify the /OBJECT and /MACHINE_CODE qualifiers together on the same command line.

/[NO]STANDARD

The /STANDARD option flags any uses of language features not supported by the ISO PASCAL standard. The default is /NOSTANDARD.

/[NO]XRAY or /[NO]LOG

The /XRAY or /LOG option causes Tool Kit PASCAL to display its progress as it compiles a program. The default is /NOXRAY and /NOLOG.

3.50 PHONE

With PRO/DECnet only, the PHONE command invokes the PRO/DECnet Phone Utility, which allows you to communicate with other users on your system or another system connected to your system by PRO/DECnet. For a complete description of the Phone Utility, including information on the PHONE command line, see the PRO/DECnet User's Guide.

Syntax

PHONE [phone-command]

Prompt

None

3.51 PRINT

This command allows printing to the local printer. The current default settings will be used to print the file(s).

Syntax

PRINT filespec[,s]

filespec

The name of the file(s) to print.

Prompt

File(s)? filespec[,s]

3.52 PRINT/REMOTE

PRINT/REMOTE is available only with PRO/DECnet. PRINT/REMOTE directs an existing file on a remote node to a line printer for printing. If the file you want to print is not on that node, you may want to use the COPY command with a /PRINT qualifier.

Syntax

PRINT**/R**EMOTE[/qualifier[,s]] filespec[,s]

qualifier

Can be one or more of the following:

/LOG
/[NO]WARNINGS

filespec

The file specification (including node specification) for each file to be queued for printing.

Prompt

File(s)? filespec[,s]

Qualifiers

/LOG

The /LOG qualifier specifies that the names of files printed are to be displayed as the operation is performed.

/[NO]WARNINGS

The /NOWARNINGS qualifier specifies that the "No such file" error messages should not be displayed when the input files do not exist.

Example

```
$ PRINT/REMOTE/LOG MYFILE.DOC <DO>
```

3.53 PURGE

PURGE deletes all but the latest versions of files, and releases the storage space that the deleted files occupied.

Syntax

PURGE[/qualifier[s]] filespec[,s]

qualifier

Can be one or more of the following:

/[NO]LOG
/KEEP:n
/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/NOWARNINGS

filespec

Is the name of the file, whose old versions are to be deleted. You must have delete access to the file.

Prompts

File(s)? filespec[,s]

Qualifiers

/[NO]LOG

Specifies that the files deleted by PURGE be listed on your terminal. The default is /NOLOG.

/KEEP:n

Specifies that the n latest versions of a file be retained.

If you do not use this qualifier, all versions but the most recent of a given file are deleted. That is, the default form of the command includes the qualifier /KEEP:1. With the qualifier explicitly stated, all but the n highest numbered versions are deleted. PURGE assumes that the version numbers of files are in sequential order, for example:

```
FILE1.LIS;4 FILE1.LIS;3 FILE1.LIS;2 FILE1.LIS;1 ...
```

and that there are no versions missing from the sequence, for example:

```
FILE1.LIS;20 FILE1.LIS;15 FILE1.LIS;14 ...
```

If more than one filespec is given with the /KEEP qualifier, all but the latest n versions of all files listed are deleted.

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the PURGE command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the PURGE command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the PURGE command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the PURGE command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the PURGE command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the PURGE command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/NOWARNINGS

The /NOWARNINGS qualifier specifies that the "No such file" error message should not be displayed when the input files do not exist.

3.54 REMOVE

This command removes the name of a task from the System Task Directory. The task is no longer installed. REMOVE/REGION takes the name of a region out of the Common Block Directory and the partition list.

Syntax

REMOVE[/qualifier] taskname

qualifier

/REGION

taskname

Is the name of the task to be removed from the System Task Directory. If you want to remove a region, specify the /REGION qualifier.

Prompts

Taskname? taskname

Qualifier

/REGION

Specifies that you want to remove a region from the Common Block Directory.

3.55 RENAME

This command changes the name, type, or version number of an existing file.

Syntax

RENAME[/qualifier[s]] old_filespec new_filespec

qualifier

Can be one or more of the following:

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/NOWARNINGS

With PRO/DECnet DCL extensions, the following qualifiers are also available:

/LOG
/NEW_VERSION

old_filespec

is the filename prior to renaming.

new_filespec

Is the desired new name of the file. If you want to specify a new directory for the file, you can only specify a directory that exists on the same volume. This is because the file is not actually copied.

Prompts

Old file name? old_filespec
New file name? new_filespec

Qualifiers

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the RENAME command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the RENAME command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the RENAME command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the RENAME command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the RENAME command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the RENAME command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/NOWARNINGS

Specifies that the "No such file" error messages should not be displayed when the old_filespec does not exist.

Additional Qualifiers Available with PRO/DECnet

Command Qualifiers

/LOG

Specifies that the names of files renamed are to be displayed as the operation is performed.

File Qualifiers

/NEW_VERSION

The NEW_VERSION qualifier specifies that the output files will be created with a version number higher than any existing files of the same name in the directory. /NEW_VERSION must appear after the file specification in the command line.

3.56 RUN uninstalled task

This command checks to see if the task is installed. If the task is not installed, RUN installs the task image, runs it, and removes the name of the task from the System Task Directory when task execution completes. In the case when a task image is not installed, RUN actually comprises the INSTALL, RUN, and REMOVE commands.

Syntax

RUN[/qualifier[s]] [$] filespec

qualifier

Can be one or more of the following:

/COMMAND:"taskcommand"
/TASK_NAME:taskname

[$] filespec

Is the name of the uninstalled task image. The dollar sign ($) directs the system to search first for the file in the PRO/Tool Kit application directory. You can display this directory by typing SHOW LOGICAL APPL$DIR.

Prompts

Task? [$]filespec

Qualifiers

/COMMAND:"taskcommand"

Use the /COMMAND qualifier to pass a command to the task you are running. The command must be inside the "quotes" and not more than 40 characters long. For example, the command

RUN/COMMAND:"PIP /LI" $PIP

runs PIP and then passes the /LI qualifier to it. This results in a directory listing.

To pass an MCR-format command (if the task will accept it), you must precede the command with the taskname followed by a space. For example:

RUN/COMMAND:"CMP TI:=file1.dat,file2.dat" $CMP

/TASK_NAME:taskname

Specifies the name under which the task is to be run. The default is to run the task under a name derived from the name of the terminal from which the RUN command was issued.

Task names are restricted to six Radix-50 characters. The Radix-50 character set consists of the 26 uppercase letters, the 10 numerals, the period (.), and dollar sign ($).

3.57 RUN installed task

If the task is installed, RUN initiates its execution. You can use RUN to initiate the execution of installed tasks on a schedule by creating entries in the system clock queue.

Syntax

RUN[/qualifier[s]] taskname

qualifier

Can be one or more of the following:

/COMMAND:"taskcommand"
/DELAY:nu
/INTERVAL:nu
/SCHEDULE:hh:mm:ss
/SYNCHRONIZE:u

taskname

Is the name of the task that RUN will execute.

Prompts

Task? taskname

Qualifiers

/COMMAND:"taskcommand"

Use the /COMMAND qualifier to pass a command to the task you are running. The command must be inside the "quotes" and not more than 40 characters long. For example, the command

RUN/COMMAND:"PIP /LI" $PIP

runs PIP and then passes the /LI qualifier to it. This results in a directory listing.

To pass an MCR-format command (if the task will accept it), you must precede the command with the taskname followed by a space. For example:

RUN/COMMAND:"CMP TI:=file1.dat,file2.dat" $CMP

The time-oriented qualifiers to RUN create entries in the system clock queue. The contents of the clock queue can be displayed with the command SHOW CLOCK_QUEUE.

/DELAY:nu

Specifies that the task be run after the stated amount of time passes. The argument nu specifies the amount of the delay as a number of units. The value n is the number of units and the value u is the time unit, as follows:

T - Ticks
S - Seconds
M - Minutes
H - Hours

Your system has a programmable clock. The frequency of this clock is 64 Hz, which results in a tick length of 1/64th of a second.

Acceptable values for these units are as follows:

T - Any positive value to a maximum of 15 bits, or 32767.

S - Any positive value to a maximum of 15 bits, or 32767.

M - The maximum value is 1440.

H - The maximum value is 24.

The system always waits at least one interval. If you specify 0, the system treats it as a 1.

/INTERVAL:nu

Specifies that the task is to be run at regular intervals. The argument nu specifies the interval as a number of units of time. The value n is the number of units and the value u is the unit, as follows:

T - Ticks
S - Seconds
M - Minutes
H - Hours

See the /DELAY qualifier for a detailed description of these units.

/SCHEDULE:hh:mm:ss

Specifies that the task be run at a particular time of day.

/SYNCHRONIZE:u

Specifies that the execution of the task be synchronized on the next occurrence of a particular clock unit. The value u is the time unit, as follows:

T - Ticks
S - Seconds
M - Minutes
H - Hours

3.58 SET [DAY]TIME

This command sets the current date and time.

Syntax

SET [**DA**Y]**TIME**:[dd-mmm-yy] [hh:mm]

dd-mmm-yy

Specifies the date. You can enter the date in either of two formats:

dd-mmm-yy	Where dd is the number of the day, mmm is the first three characters of the name of the month, and yy is the number of the year (relative to 1900).
mn/dd/yy	Where mn is the number of the month, dd is the number of the day, and yy is the number of the year (relative to 1900).

Regardless of the format you choose, the date is displayed in the first format.

hh:mm[:ss]

Is the time to which the system will be set (24-hour format) in the form 14:35.

Prompts

```
Time?   hh:mm
Date?   dd-mmm-yy
```

3.59 SET DEFAULT

The SET DEFAULT command establishes the default directory and/or device from the current default. You can set the default for the directory, the device, or both.

It is possible to set default to a nonexistent device or directory.; no error is reported when the SET DEFAULT command is executed. However, an error is reported when you try to perform an operation in the nonexistent directory or on the nonexistent device.

Syntax

SET DEFAULT [device:][dirspec]

device:

Is the device name.

dirspec

is the directory name enclosed in square brackets, for example: [MYDIR].

Prompts

Device and/or directory? [device:][dirspec]

3.60 SET DEVICE

SET DEVICE establishes certain attributes for the specified device.

Syntax

SET DEVICE:ddnnn:/device-attribute[s]

ddnn

Specifies the device for which attributes are to be set. When the one-line form of the command is used, the parameter is ddnnn: preceded by a colon (:) at the end of the word DEVICE.

device-attribute

Can be one or more of the following:

/[NO]LOWERCASE
/WIDTH:n

Prompts

Device? ddnnn:
Device attribute? qualifier

Device Attributes

/[NO]LOWERCASE

The /LOWERCASE attribute sets a terminal or line printer so that lowercase characters are not converted to uppercase for printing.

The /NOLOWERCASE attribute sets a terminal or line printer so that lowercase characters are converted to uppercase for printing. This is the default.

You can use the SET TERMINAL/LOWERCASE command to set TI: in this fashion.

/WIDTH:n

Sets the size of a device's I/O buffer. The value of n (octal) is the length in characters of a line on the device.

For terminals, n must be greater than 2 and not greater than 255. The terminal driver does not discard excess characters, but puts them in a record of their own. That is, excess characters appear one line below the line in which they should appear.

3.61 SET HOST

With PRO/DECnet only, SET HOST invokes the Remote DECnet Terminal application (available with PRO/DECnet). This allows your Professional node to communicate with a remote host, performing terminal operations as though it were local to the host. The host must support terminal emulation from remote nodes.

Syntax

SET **H**OST node_name

node-name

Is the name of the node to which you want to connect.

Prompts

Host? node_name

Qualifiers

None

To disconnect from a VMS or RSX host computer, either log off of the host, or press the <MAIN SCREEN> or <EXIT> key. The preferable method is to log off the host computer.

To disconnect from a TOPS-10 or TOPS-20 host computer, log off of the host, and then press the <MAIN SCREEN> or the <EXIT> key.

Once you have exited from the host computer, press <RESUME> to return control to your node.

Refer to the *PRO/DECnet User's Guide* for information about differences in processing during terminal emulation.

3.62 SET PRIORITY

This command alters the priority of an active task.

Syntax

SET PRIORITY:n taskname

n

Specifies the new priority you want to assign to the task. Priority numbers are in the range of from 1 through 250. The value n is octal or decimal. Append a decimal point to indicate that it is a decimal number.

taskname

Is the name of the task whose priority is to change.

Prompts

```
Priority number?   n
Taskname?   taskname
```

3.63 SET PROTECTION

SET PROTECTION establishes the protection status of a file. Protection status determines which categories of user may access a file and what each kind of user may do to the file.

Syntax

SET PROTECTION[/qualifier[s]] filespec[,s] (code)

qualifier

Can be one or more of the following:

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/[NO]WARNINGS

With PRO/DECnet DCL extensions, the following qualifiers are also available:

/OWNER[:uic]

filespec

Is the name of the file to be protected

(code)

There are four kinds of users:

SYSTEM	The operating system itself, and privileged users, those with group numbers of 10 or less.
OWNER	The user whose UIC the file was created under.
GROUP	All users whose group number the file was created under.
WORLD	All other users.

There are also four kinds of access to files:

READ	The user or user's tasks can read, copy, print, type or run the file (if it is a task image).
WRITE	The user or user's tasks can write new data to the file.
EXTEND	The user or user's tasks can increase the amount of disk space allocated to the file.
DELETE	The user or user's tasks can delete the file.

The system default protection code is expressed as follows:

```
(SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:RWED)
```

Under this code, the system and privileged users have full access to your files, You, as well as others with your group number, also have full access to your files. Other nonprivileged users can only read your files. If no other protection is specified, all files have this protection.

If you want to protect a particular file differently from the current system default, name in the protection code only the user group whose access rights you want to change and the access form you want to grant to that group. If you want to deny all access to a group, simply name the group and omit the colon (:) and the code for the access form.

If any of the parameters (System, Owner, Group, World) is eliminated from the code, SET PROTECTION does not change the value of that parameter. If the parameter is listed with no code (SYSTEM:R,OWNER:RWED,GROUP,WORLD), SET PROTECTION assigns no access to that parameter.

Prompts

File? filespec[,s]

Qualifiers

/DATE:dd-mmm-yy

The /DATE qualifier specifies that the SET PROTECTION command affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that the SET PROTECTION command affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that the SET PROTECTION command affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers combined specifies that the SET PROTECTION command affect only files created within the given range.

/TODAY

The /TODAY qualifier specifies that the SET PROTECTION command affect only files created on the day the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the SET PROTECTION command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/NOWARNINGS

This qualifier suppresses display of "No such file" error messages when the input files specified do not exist.

Additional Qualifiers Available with PRO/DECnet

Command Qualifiers

None

File Qualifiers

/OWNER:uic

Sets the ownership of a file to the specified UIC, in the form [ggg,mmm].

3.64 SET PROTECTION/DEFAULT

Establishes the default protection for all files created during the current login session. The protection for a file limits the type of access available to other system users.

Syntax

SET PROTECTION:protection_code/**DEF**AULT

protection_code

Is the protection code to be given to the files created during your current login session.

The code takes the following format:

([SYSTEM[:n]],[OWNER[:n]],[GROUP[:n]],[WORLD[:n]])

where:

SYSTEM	Is the system level protection code.
OWNER	Is the owner level protection code.
GROUP	Is the group level protection code.
WORLD	Is the world level protection code.
n	Is the type of access that is to be allowed.

The value for n can be any or all of the following values, in any order:

R	Read allowed. The user may read, copy, print or type (scroll through) the file, and if it is a task, run it.
W	Write allowed. The user (or the user's application) can add new information to the file.
E	Extend allowed. The user, or the user's tasks, can change the amount of disk space allocated to the file.
D	Delete allowed. The user (or the user's application) can delete the file.

Only those categories that are specified are allowed; all others remain protected. If you enter one of the categories, but do not

specify a value for n, no access is allowed at that level. In addition, if a category is not entered, then access is denied for that category. For example, SET DEFAULT PROTECTION:(OWNER:RWED) allows only owner access to the files created.

NOTE

Your default file protection is set during the login process. You may specify a default file protection that is restored during the login process with Environment Services.

Prompts

Protection code? protection_code

3.65 SET PROTECTION/NODEFAULT

Removes, your currently defined default file protection. (See SET PROTECTION/DEFAULT.) Any file created during the current login session will have the file protection of:

(SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:RWED)

Syntax

SET PROTECTION/NODEFAULT

Prompts

None

3.66 SET TERMINAL

This command sets various attributes of the terminal.

Syntax

SET TERMINAL[:ttnnn:] attribute[/attribute[s]]

ttnnn

Is the number of the terminal.

attribute

Can be one or more of the following:

Group 1: Common Use

/[NO]LOWERCASE	/[NO]UPPERCASE
/WIDTH:n	

Group 2: Terminal Setup

/VT100	/VT101
/VT102	/VT105
/VT125	/VT131
/VT132	/VT2XX
/LA50	/LA100
/LA210	/LQP02
/PC300	/[NO]FORM_FEED
/[NO]TAB	/[NO]HARDCOPY
/[NO]SCOPE	/PAGE_LENGTH:n

Group 3: Task Setup

/[NO]ECHO	/[NO]ESCAPE
/[NO]TYPEAHEAD[:n]	/[NO]WRAP
/[NO]EIGHT_BIT	

Prompts

Terminal Attribute? attribute[/attribute[s]]

You can set several attributes at once. If one of these commands fails, all others following it in the command line also fail.

Group 1: Common Use

The following parameters set terminal characteristics that are regularly needed by the average terminal user.

/[NO]UPPERCASE

/UPPERCASE is the default. All characters typed on the terminal appear in uppercase. /NOLOWERCASE is the same as /UPPERCASE.

/[NO]LOWERCASE

Characters typed on the terminal in lowercase appear in lowercase. Most system tasks will accept input in lowercase. Note that some terminal escape sequences use lowercase characters. If, for instance, the keypad commands in EDT do not seem to be working, you may have to set the /LOWERCASE attribute.

/WIDTH:n

The SET TERMINAL/WIDTH command sets the width of your terminal, that is, the length of a line. The value n can range from 0 through 132. Note that a line length of zero means no commands can be entered on the terminal.

Group 2: Terminal Setup

The following qualifiers set hardware characteristics of terminals. For certain common models, you can set a number of characteristics automatically simply by identifying the model. For other terminals, you must set these characteristics explicitly.

Here are the models for which setup is provided:

VT100	VT101
VT102	VT105
VT125	VT131
VT132	VT2XX
LA50	LA100
LA210	LQP02
PC300	

All of these are DIGITAL terminals. Those with "LA" designations are hard-copy terminals; those with "VT" and "PC" designations are video models.

Setting a terminal to a particular model does not mean that the terminal will behave like that model. It means only that the operating system will treat the terminal as if it were that model. This feature is intended primarily to deceive tasks that expect a certain terminal model or to identify a terminal as to what it is.

You can set hard-copy terminals as video terminals and you can set video terminals as hard-copy terminals. For the terminal user, the most noticeable difference will be in the way the DELETE key operates. Setting terminals from hard-copy to video may prove disruptive; setting terminals from video to hard-copy is less likely to cause trouble.

Setting a terminal to a particular model designation automatically sets a number of attributes for the terminal.

You can also set these attributes individually.

You can find out how your terminal is known by issuing a SHOW TERMINAL command.

NOTE

The DCL single line editor does not honor the ECHO, LOWER, WRAP, or WIDTH terminal attributes.

/NOHARDCOPY
/SCOPE

Sets the terminal as a video terminal. /SCOPE is the equivalent of /NOHARDCOPY.

/HARDCOPY
/NOSCOPE

Sets the terminal as a hard-copy terminal. /NOSCOPE is the equivalent of /HARDCOPY.

/[NO]FORMFEED

If the terminal hardware supports form feeds, the terminal should be set /FORMFEED. If form feeds are handled by the software providing line feeds, the terminal should be set /NOFORMFEED.

/[NO]TAB

If the terminal hardware supports horizontal tabs, the terminal should be set TAB. If tabs are handled by the software providing spaces, the terminal should be set NOTAB.

/PAGE_LENGTH:n

Defines the number of lines to a page. By convention, a page is usually considered the number of lines to a screenful on video terminals or the number of lines between perforations on hard-copy terminals. This attribute sets the length of the page.

Group 3: Task Setup

The following parameters set terminal characteristics that may be needed by system or user tasks. Most system tasks that require these attributes in a terminal will set the attributes when they attach the terminal. User tasks can also do this. This will be transparent to the user. These parameters are included for use in cases where the task does not do this setting.

/[NO]ECHO

Enables (or disables) echoing of characters typed on the terminal.

/ECHO is the default. Characters typed on the terminal are echoed on the terminal.

/NOECHO suppresses the echo.

/NOECHO changes nothing but echoing. Commands can still be passed to the system, but the system passes no echo back.

/[NO]ESCAPE

Enables (or disables) recognition of escape sequences from a terminal.

/NOESCAPE is the default. When you press the ESC key (equivalent to CTRL/3 on LK201 keyboards), it is interpreted as a line terminator (with a line feed but no carriage return).

ESCAPE enables the recognition of escape sequences from the terminal. When you press the ESC key, it is interpreted as the beginning of an escape sequence. The ESC key will no longer terminate a line.

This is a rarely used parameter. Most tasks that recognize escape sequences will attach the terminal so that escape sequences are passed without the user's knowledge, usually from the keypad.

/[NO]WRAP

Specifies that the terminal automatically wrap (or not wrap) lines longer than its line width.

/WRAP is the default. The terminal automatically issues a carriage return and a line feed when you type to whatever line width the terminal is set for.

/NOWRAP overrides this feature and permits unlimited line length.

/[NO]EIGHT_BIT

Enables (or disables) a terminal's /EIGHT_BIT attribute.

/EIGHT_BIT is the default, because the Professional 350 uses eight bit mode to display the DEC Multinational Character Set. /EIGHT_BIT allows the terminal to pass all eight bits of the ASCII character. This attribute is used when your terminal is communicating with some device that sends 8-bit ASCII.

3.67 SHOW ASSIGNMENTS

SHOW ASSIGNMENTS displays all the logical names in the logical name table(s), or it displays the current equivalence value assigned to a specific logical name.

Syntax

SHOW **AS**SIGNMENTS[/qualifier[s]] [logical_name]

qualifier

Can be any combination of the following:

/USER
/SESSION
/SYSTEM
/ALL

logical_name

Is the name of a unique logical assignment.

Prompts

None

Qualifiers

/ALL

Specifies that all logical names in the specified logical name table(s) be displayed.

/USER

Limits the search to the user logical name table.

/SESSION

Limits the search to the session logical name table. If you do not explicitly specify a logical name table, P/OS uses /SESSION by default.

/SYSTEM

Limits the search to the system logical name table.

3.68 SHOW CLOCK_QUEUE

SHOW CLOCK_QUEUE displays information about tasks that are currently in the clock queue. The displayed information comprises task names, the next time the task is scheduled to run, and each task's reschedule interval, if any.

Syntax

SHOW **CL**OCK_QUEUE

Prompts

None

3.69 SHOW COMMON

SHOW COMMON displays the names of resident commons installed in the system, their PCB addresses, the number of attached tasks, and the status of the common.

You can also display information about a single common; optionally, you can include a list of tasks attached to the common.

If you do not name a common, information about all commons in the Common Block Directory is displayed.

Syntax

SHOW **CO**MMON[:name][/qualifier]

qualifier

/TASK

name

Is the name of the common.

The display is in the following format:

commonname pcbaddr taskcount statusbits

In the format that SHOW COMMON displays at the terminal, commonname is the name of the common, pcbaddr is the address of the Partition Control Block (PCB), taskcount is the number of tasks mapped to the common, and statusbits is a list of common region status bits that are set. Usually, the status bits will indicate that the common region is either fixed in memory (FXD) or out of memory (OUT).

Table 3-4 lists the status bits and their meanings.

Prompts

None

Qualifier

/TASK

Specifies that you want a list of tasks attached to a particular common region showing the number of times each task is mapped to the common (mapping count).

Table 3-4: Status Bits

Status Bit	Meaning
CAF	Checkpoint allocation failure
-CHK	Not checkpointable
CKP	Checkpoint in progress
CKR	Checkpoint requested
COM	Library or common
DEL	Marked for delete on last detach
DRV	Driver common
FXD	Fixed in memory
LIO	Long I/O
LFR	Last load failed
NSF	Not shuffleable
OUT	Out of memory
PER	Parity error
PIC	Position independent
RON	Read-only common

3.70 SHOW DAYTIME or SHOW TIME

This command displays the current day and time.

Syntax

SHOW **DA**YTIME

or

SHOW **TIM**E

Prompts

None

3.71 SHOW DEFAULT

This command displays the current default device and directory name.

Syntax

SHOW **DEF**AULT

Prompts

None

3.72 SHOW DEVICES

This command displays information concerning the devices that are included in the system.

Syntax

SHOW **DEV**ICES[/qualifier]

qualifier

Can be one or more of the following:

/dd[nnn]:
/WIDTH:ddnnn:

Prompts

None

If you do not include a qualifier, SHOW DEVICES displays a list of all the devices on the system, including terminals.

Qualifiers

/dd[nnn]:

Displays information about all devices of a particular type on the system. The value of dd: is a two-letter device mnemonic terminated with a colon, which indicates the type of device controller. The display shows the devices under that type of controller by model name.

/WIDTH:ddnnn:

Displays the size of the I/O buffer (line length) for a particular device, including a terminal.

3.73 SHOW LOGICALS

SHOW LOGICALS displays all the current logical names in the logical name table(s), or it displays the current equivalence value of a specific logical name.

Syntax

SHOW **LOG**ICALS[/qualifier[s]] [logical_name]

qualifier

Can be any combination of the following:

/USER
/SESSION
/SYSTEM
/ALL

logical_name

Is the name of a unique logical assignment.

Prompts

None

Qualifiers

/ALL

Specifies that all logical names in the specified logical name table(s) be displayed.

/USER

Limits the search to the user logical name table.

/SESSION

Limits the search to the session logical name table. If you do not explicitly specify a logical name table, P/OS uses /SESSION by default.

/SYSTEM

Limits the search to the system logical name table.

3.74 SHOW MEMORY

SHOW MEMORY invokes the Resource Monitoring Display (RMD), a system utility program. This program displays in a graphic manner the status of much of the system.

The RMD program is useful for monitoring the general activity of the system. It is also useful for a new user to see how the operating system operates. However, you should understand that the graphic display is approximate and cannot be used for critical measurement. (Refer to Chapter 8 for a description of RMD.)

Syntax

SHOW **ME**MORY

Prompts

None

3.75 SHOW PROTECTION

SHOW PROTECTION displays the current file protection to be applied to all new files created during the current session. You can change the default protection at any time using the SET PROTECTION/DEFAULT command.

Syntax

SHOW **PROT**ECTION

Prompts

None

3.76 SHOW TASKS/ACTIVE

SHOW TASKS/ACTIVE displays information about active tasks in brief and full format.

Syntax (Brief Format)

SHOW **TA**SKS/**AC**TIVE[:ttnnn:][/qualifier[s]]

ttnnn:

If you name a terminal in the command, the display shows in brief form the tasks active at that terminal. If you do not name a terminal, the display shows in brief form the tasks active at your terminal.

qualifier[s]

Can be one or more of the following:

/BRIEF
/ALL

Qualifiers

/BRIEF

Specifies that you want to display information about active tasks in the brief format. This is the default and need not be specified.

The brief format includes task names and the originating terminal in parentheses next to each task name.

/ALL

Specifies that you want to display information about all tasks active on the system. The default is to show information about tasks active at your terminal only.

Syntax (Full Format)

SHOW **TA**SKS/**AC**TIVE/**FU**LL [taskname]

SHOW **TA**SK[:taskname]/**AC**TIVE/**FU**LL

taskname

If you include a task name, the display shows full information on that task. If you do not name a task, the display shows full information on all currently active tasks.

Qualifiers

/FULL

Requests the full format display for the SHOW TASK/ACTIVE command. This format includes detailed information on the state of one or all tasks active on the system. The format of the display is as follows:

```
taskname tcbaddr parname pcbaddr taskaddrlimits pri defpri
STATUS: statusflags
TI - ttnnn: IOC - iocount BOC -buffiocount EFLG -eventflags PS -pswv
PC - pcval REGS 0-6 r1val r2val r3val r4val r5val r6val
```

Table 3-5 describes the items that appear in the display. If the task is not in memory, the PC, PSW, and other registers are not displayed.

The display shows the state of the task at the time the command was issued. You can display similar information dynamically (in real time) for a single task with the /DYNAMIC qualifier, as explained in Section 3.78.

Table 3-5: Display for SHOW TASKS/ACTIVE

Item	Description
tcbaddr	The physical address of the Task Control Block (TCB).
parname	The name of the task's partition.
pcbaddr	The physical address of the Partition Control Block (PCB).
taskaddrlimits	The base and top of the task's dynamic subpartition as physical addresses.
pri	The priority at which the task is actually running.

defpri	The default priority at which the task was built or installed to run.
STATUS	The task's status flags. These are identified in Table 3-6.
TI	The initiating terminal.
IOC	The decimal I/O count for the task.
BIO	The decimal count of I/O buffered by the terminal driver and Executive.
EFLG	Local event flags.
pswval	The Processor Status Word.
pcval	The Program Counter.
REGS	The contents of the task's other registers. If the task was spawned by another task, the name of the parent task is also displayed.

Table 3-6: Task Status Flags

Status Flag	TCB Flag	Meaning
ABO	T2.ABO	Being aborted
ACP	T3.ACP	Ancillary Control Processor
AST	T2.AST	AST state
BLK	TS.STP	Blocked externally by CLI command
CAF	T2.CAF	Dynamic checkpointing allocation failure
CAL	T3.CAP	Checkpoint space allocated in task image
-CHK	TS.CHK	Not checkpointable
CIP	TS.CIP	Blocked for checkpoint in progress
CKD	TS.CKD	Checkpointing disabled
CKP	TS.CKP	Checkpointed
CKR	TS.CKR	Checkpoint request pending
CLI	T3.CLI	Command Line Interpreter
DST	T2.DST	ASTs disabled
-EXE	TS.EXE	Not executing
FXD	T2.FXD	Fixed in memory
GFL	T3.GFL	Task has own group global event flags locked

Status Flag	TCB Flag	Meaning
-PMD	T3.PMD	Suppress PMD on SST abort
OUT	TS.OUT	Out of memory
PRV	T3.PRV	Privileged
RDN	T3.RDN	I/O being run down
REM	T3.REM	Remove on exit
REX	T2.REX	Abort AST effected or in progress
ROV	T3.ROV	Resident overlays
RST	T3.RST	Restricted - used by layered software
SEF	T2.SEF	Stopped for event flag
SLV	T3.SLV	Slaved
SPN	T2.SPN	Being suspended
SPNA	T2.SPN	Suspended prior to AST
STP	T2.STP	Stopped
STPA	T2.STP	Stopped prior to AST
SWS	T3.SWS	Reserved for software services
WFR	T2.WFR	In a wait-for state
WFRA	T2.WFR	In a wait-for state prior to AST
DSP	T4.DSP	Task was built for user-mode I- and D-space
LDD	T4.LDD	Task's load device has been dismounted
MUT	T4.MUT	Task is multiuser task
PRO	T4.PRO	Task is (or should be) a prototype task
PRV	T4.PRV	Task was privileged but has cleared T3.PRV
SNC	T4.SNC	Task uses commons for synchronization

NOTE

These status flags are displayed by several forms of the SHOW TASK command. They give information on what is happening within the task and between the task and the system. They also identify certain kinds of tasks. Names prefixed by a minus (-) indicate the complement of the condition. Thus, -CHK means the task is noncheckpointable. You must understand how the system runs tasks to understand the meanings of all these flags. See the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual* and the *P/OS System Reference Manual* for more information.

3.77 SHOW TASKS/INSTALLED

SHOW TASKS/INSTALLED displays information about installed tasks in either brief or full format.

Syntax

SHOW **TA**SKS[:taskname]/**INS**TALLED[/qualifier[s]]

taskname

Specifies the task for which you want information displayed. If you do not specify a task name, information on all installed tasks is displayed.

qualifier[s]

Can be one or more of the following:

/BRIEF
/FULL

Qualifiers

/BRIEF

Requests information on installed tasks in a brief format. This is the default and need not be specified. The format of the display is as follows:

taskname ident parname priority size ddnnn:-lbn [memstate]

In this display, taskname is the name task, ident is the task version identification (or the version of the prototype task), parname is the partition in which the task is installed, priority is its priority, size is the size of the task in bytes, ddnnn: is the device from which it is to be loaded, lbn is the logical block number of its disk address, and memstate is the task memory state, which can be FIXED, CHECKPOINTED, or blank.

If the task version identification is missing (with the rest of the line moved left) or if it is garbage, the task was installed from a disk that is no longer present. If the task version number is a date, such as 07JUL, the task was compiled on that day.

/FULL

Requests the full format of the SHOW TASKS/INSTALLED command. This format displays a detailed list of the states of one or all of the installed tasks in the system, depending on the presence of the taskname parameter. The format of the display is as follows:

```
taskname tcbaddr parname pcbaddr taskaddrlimits pri defpri
STATUS:  statusflags
TI - ddn:  IOC - iocount EFLG - eventflags
```

This display is in the same format as that of SHOW TASKS/ACTIVE/FULL. (See Section 3.76 for a description of the display.)

taskname

Specifies a task for which full information is to be displayed. If you do not specify a task name, information about all installed tasks is displayed.

3.78 SHOW TASK:taskname/DYNAMIC

SHOW TASK:taskname/DYNAMIC displays a moving picture of a task's activity on the terminal.

Syntax

SHOW **TA**SK:taskname/**D**YNAMIC[/qualifier]

taskname

Specifies the task you want to inspect.

qualifier[s]

Can be one or more of the following:

/RATE:n
/PRIORITY:n

Qualifiers

/DYNAMIC

Specifies that you want the moving display. This function is performed by the RMD task. RMD has four display pages:

- Task that displays a task header
- Active that displays all or part of the Active Task List
- Memory that displays the contents of memory
- Help that provides help on RMD

Once you have invoked any one of these pages, through either a SHOW TASK/DYNAMIC command or a SHOW MEMORY command, you can move from one page to the other without leaving RMD. The first character of each page name is a command to RMD to go to that page. In addition, a <CTRL/[> entered from a page permits you to enter setup commands for that page. See Chapter 8 for more information on the features of RMD.

The setup commands for the Task page permit you to change the task header being displayed. The task you name remains the default display for the Task page. This means you can observe a single task header, then jump to the Memory page or the Active page, and then back to the Task page and the named task will still be on display.

/RATE:n

The /RATE qualifier allows you to set the rate at which the RMD display screen is to be replotted on the first display. The value n is the number of seconds between replots. The default n is 1. You can change the rate by pressing ESC and entering a new rate. The rate setting returns to 1 as soon as you change pages. If you want a slower rate, you must reset the rate each time you change the page.

Use this qualifier to slow down the display.

Task headers may change more often than once per second, but once per second is the most rapid rate available.

/PRIORITY:n

This qualifier specifies the highest priority to be displayed. The default is 250, the highest priority on the system. Use this qualifier to shorten the display to one screen length.

Notes

Tasks built including ODT can be observed using this command, but if you are single-stepping through the task or otherwise using breakpoints, the registers displayed will be those of the ODT task, not those of the named task.

This display is particularly useful for observing complex assemblies, as you can tell how far the assembly has gone by which files are open. You may also be interested in observing the Task Builder at work.

3.79 SHOW TASKS/ACTIVE/DYNAMIC

SHOW TASKS/ACTIVE/DYNAMIC displays either a moving picture of the Active Task List on a video terminal, or a snapshot display on a hard-copy terminal.

Syntax

SHOW **TA**SKS/**AC**TIVE/**D**YNAMIC[/qualifier]

qualifier

/RATE:n

Qualifiers

/RATE:n

The /RATE qualifier allows you to set the rate at which the RMD display screen is to be replotted on the first display. The value n is the number of seconds between replots. The default n is 1. You can change the rate once the display begins by pressing the ESC key and entering a new rate. The rate setting returns to 1 as soon as you change pages. If you want a slower rate, you must reset the rate each time you change the page.

Use this qualifier to slow down the display.

The Active Task List may change more often than once per second, but once per second is the most rapid rate available.

3.80 SHOW TASK/LOGICAL_UNITS

SHOW TASK:taskname/LOGICAL_UNITS displays the static logical unit number (LUN) assignments for an installed task.

Syntax

SHOW **TA**SK:taskname/**LO**GICAL_UNITS

taskname

Specifies the task for which you want LUN assignments displayed. This must be the name of a task installed through INSTALL, and not through the install-run-remove form of RUN.

The display consists of a list of physical devices and corresponding static LUN assignments. The display does not show any dynamic LUN assignments, even when the specified task is running.

Prompts

Taskname? taskname

3.81 SHOW TERMINAL

SHOW TERMINAL displays information about your terminal and another optional terminal connected to the system. Each SHOW TERMINAL attribute is directly related to a SET TERMINAL attribute (refer to Set Terminal for a discussion of these attributes).

Syntax

SHOW **TE**RMINAL[:ttnnn:] [/attributes[s]]

ttnnn

Is the number of the terminal for display

attribute

Can be one or more of the following:

Group 1: Common Use

/[NO]LOWERCASE /[NO]UPPERCASE
/WIDTH:n

Group 2: Terminal Setup

/VT100	/VT101
/VT102	/VT105
/VT125	/VT131
/VT132	/VT2XX
/LA50	/LA100
/LA210	/LQP02
/PC300	/[NO]FORM_FEED
/[NO]TAB	/[NO]HARDCOPY
/[NO]SCOPE	/PAGE_LENGTH:n

Group 3: Task Setup

/[NO]ECHO	/[NO]ESCAPE
/[NO]TYPEAHEAD[:n]	/[NO]WRAP
/[NO]EIGHT_BIT	

Prompts

None

3.82 SPAWN

The SPAWN command allows tasks to be executed in the background. These tasks allow you to continue to use the terminal. For example, SPAWN can initiate a task building session, allowing you to perform other tasks.

SPAWN should not be used with tasks that attach the terminal. SPAWN will return an error message, if it was asked to initiate an uninstalled task.

Syntax

SPAWN cmd

cmd

Is any PRO/Tool Kit Command Language command.

Prompts

None

3.83 START

START resumes the execution of a task that was halted by a STOP$S directive. Starting a task that has been stopped is different from continuing a suspended task.

Syntax

START taskname

taskname

Is the name of the task you want to start.

Prompts

Taskname? taskname

3.84 START/UNBLOCK

This command continues the execution of a task that was blocked by the STOP/BLOCK command.

Syntax

START/**UN**BLOCK taskname

taskname

Is the name of the task to be unblocked.

Prompts

Taskname? taskname

3.85 STOP/BLOCK

This command blocks an installed, active task. After execution of this command, the task no longer executes or competes for memory (refer to START/UNBLOCK).

Syntax

STOP/**BL**OCK taskname

taskname

Is the name of the task to be blocked.

Prompts

Taskname? taskname

3.86 SUBMIT/REMOTE

With PRO/DECnet only, SUBMIT/REMOTE directs an existing command file on a remote node to be executed. The file is queued to the remote node's command file or batch file processor. If the command file is not on the node where it will be executed, you may want to use a COPY command with a /SUBMIT qualifier.

The success of the command does not guarantee that the batch or command file was executed successfully, only that the execution request was given successfully to the remote processor.

Syntax

SUBMIT/**RE**MOTE[/qualifier[,s]] commandfile[,s]

qualifier

Can be one or more of the following:

/LOG
/NOWARNINGS

commandfile

The output file descriptor for each remote node command file that you want to execute. Command files are not deleted after execution.

Prompts

File(s)? commandfile[,s]

Qualifiers

/LOG

Specifies that the names of files submitted are to be displayed as the operation is performed.

/NOWARNINGS

Specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

3.87 TYPE

This command displays the contents of a file or group of files on the terminal.

Syntax

TYPE[/qualifier[s]] filespec[,s]

qualifier

Can be one or more of the following:

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/NOWARNINGS
/SHARED

With PRO/DECnet DCL extensions, the following qualifiers are also available:

/LOG
/MACY11

filespec

Is the name of the file to be displayed.

Prompts

File(s)? filespec[,s]

Qualifiers

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the TYPE command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the TYPE command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the TYPE command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the TYPE command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the TYPE command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the TYPE command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/NOWARNINGS

Specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

/SHARED

Specifies that other users are able to access the file while you are typing it.

Additional Qualifiers Available with PRO/DECnet

Command Qualifiers

/LOG

Specifies that the names of files typed and their sizes are to be displayed as the operation is performed.

File Qualifiers

/MACY11

Specifies that the input or output file on the remote TOPS-10 or TOPS-20 system is in MACY11 format. This qualifier is used to inform the remote DECsystem to handle the file appropriately when copying to or from the P/OS system. /MACY11 must be entered after the file specification in the command line.

3.88 UNLOAD

The UNLOAD command removes a loadable device driver from memory. Note that the device driver data base will not be removed from the system. Reboot the system to remove the driver data structures.

Note the following restrictions:

- If a device is mounted, attached, or has outstanding I/O, its driver cannot be unloaded.
- The UNLOAD command cannot remove a data base from memory, even if the data base was loaded by means of the LOAD command.

Syntax

UNLOAD dd:

dd:
Is a two-character ASCII device name.

Prompt

Device? dd:

3.89 UNLOCK

UNLOCK unlocks locked files. Locked files are files that have been improperly closed because a task aborted or stopped execution while the file was open. Locked files are identified by an L in the directory listing.

Syntax

UNLOCK[/qualifier[s]] filespec[,s]

qualifier

Can be one or more of the following:

/DATE:dd-mmm-yy
/SINCE:dd-mmm-yy
/THROUGH:dd-mmm-yy
/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy
/TODAY
/EXCLUDE:filespec
/[NO]WARNINGS

filespec

Is the name of the file to be unlocked.

Prompts

File(s)? filespec[,s]

Qualifiers

/DATE:dd-mmm-yy

The /DATE qualifier specifies that you want the UNLOCK command to affect only files created by the value specified for /DATE.

/SINCE:dd-mmm-yy

The /SINCE qualifier specifies that you want the UNLOCK command to affect only files created on or since the value specified by /SINCE.

/THROUGH:dd-mmm-yy

The /THROUGH qualifier specifies that you want the UNLOCK command to affect only files created on or before the value specified by /THROUGH.

/SINCE:dd-mmm-yy/THROUGH:dd-mmm-yy

The /SINCE and /THROUGH qualifiers can be combined to specify that you want the UNLOCK command to affect only files created within that range.

/TODAY

The /TODAY qualifier specifies that you want the UNLOCK command to affect only files created on the same day as the command is issued.

/EXCLUDE:filespec

The /EXCLUDE qualifier specifies that you want the UNLOCK command not to affect certain files. The filespec argument to /EXCLUDE can contain wildcards, but the filespec must contain a version number, either explicitly or as the "*" wildcard.

/[NO]WARNINGS

The /[NO]WARNINGS qualifier specifies that the "No such file" error messages should not be displayed when the input files specified do not exist.

CHAPTER 4

THE INDIRECT COMMAND PROCESSOR

This chapter describes the Indirect Command Processor ("Indirect") and indirect command files. Section 4.33 through 4.54 describe the processor directives that control execution of Indirect. The final two sections in the chapter provide information on compatibility with RSX systems, as well as errors that Indirect can return.

4.1 INDIRECT COMMAND FILES

An indirect DCL command file is a text file containing a list of DCL-specific command lines and special directives that allow you to control command file processing. The indirect command processor reads the indirect DCL command file, interprets the directives, and passes the DCL commands to DCL.

To initiate an indirect DCL command file, enter the file specification, preceded by an at sign (@), whenever DCL can accept input. For example:

```
$ @DCLIPT.CMD
```

The default file type for indirect DCL command files is .CMD. Thus, the command line in the previous example could also be input as follows:

```
$ @DCLIPT
```

Indirect DCL command files can also be nested. The maximum level of nesting is four.

Note that the Indirect directives described in this chapter can only be used in indirect DCL command files.

4.2 INDIRECT COMMAND PROCESSOR

When processing an indirect command file, the Indirect task first reads the command file and interprets each command line either as a command to be passed directly to DCL or as a request for action by Indirect. The directives to Indirect are distinguished by a period (.) as the first character in the line.

The Indirect directives form a procedural language that allows you to:

- Define and assign values to logical, numeric, and string symbols
- Substitute a symbol's value into any line of the command file
- Perform arithmetic functions
- Manipulate strings
- Display text on the user's terminal screen
- Ask questions of a user
- Control the flow of a command file
- Call subroutines
- Detect error conditions
- Test symbols and conditions
- Create and access data files
- Parse commands and data
- Enable or disable any of several operating modes
- Display forms and control user entry of data

Two directives (.BEGIN and .END) permit you to use block structure in the command file. Modular, block-structured command files are easier to debug and maintain. More importantly, *Begin-End blocks isolate local symbol definitions* as well as labels, and thus conserve symbol table space.

When you define a symbol, Indirect creates an entry for the definition in an internal symbol table. These symbol table entries retain their definitions throughout execution of the command file if defined locally, or throughout the execution of

all levels of nested command files if defined globally. Local symbols defined within a block, however, retain their definitions only throughout the execution of the commands within that block; they are erased from the symbol table when Indirect encounters an .END directive.

An Indirect directive (.ENABLE GLOBAL) allows the definition of some symbols as global to all file levels. Otherwise, each time Indirect enters a deeper level, it masks all symbols defined by the previous level out of the symbol table so that only symbols defined in the current level are available. These symbols are local only to the level of the file in which they are defined. When control returns to a previous level, the symbols defined in that level become available once again and the ones from the deeper level are lost.

When Indirect reaches the end of the highest level indirect command file, it displays the following message (unless inhibited by the .DISABLE DISPLAY directive):

```
$ @<EOF>
```

and then exits. DCL then prompts for a command from the keyboard. Indirect displays on the requesting terminal's screen every DCL command line as it is executed.

An Indirect file can also include comments, which the processor prints at the terminal screen. Comments that begin a line interpreted by DCL have a leading semicolon (;). Comments that appear after the start of a DCL command line have a leading exclamation point (!). All lines in an indirect command file that begin with a period in column 1 followed by a semicolon (.;) are treated as comments but are not displayed when the file is processed.

Indirect attaches the terminal while processing contiguous comment lines that begin with a semicolon. This permits you to type CTRL/O and suppress a lengthy comment. Output is resumed by typing another CTRL/O or is resumed at the next DCL command line or Indirect directive statement in the command file.

If the specified command file cannot be found in the current directory, Indirect will attempt to translate the logical name IND$COMMAND_LIBRARY and look for the command file there. This permits a user to place commonly used command files in one place. The following DCL command establishes this behavior.

```
$ ASSIGN DW1:[USERLIBRY] IND$COMMAND_LIBRARY
```

Subsequently, the user command:

```
$ @COMPILE
```

results in Indirect first looking for COMPILE.CMD in the current default directory and, if not found, looking for:

```
DW1:[USERLIBRY]COMPILE.CMD
```

Note that the PRO/Tool Kit uses LB:[1,2] as the initial value of IND$COMMAND_LIBRARY. You can specify your own initial value in the file START.CMD.

4.3 SUMMARY OF INDIRECT DIRECTIVES

The Indirect directives described in this chapter are listed here by category.

Label Definition

.label:	Assigns a name to a line in the command file so that the line may be referenced.

Symbol Definition

.ASK	Defines or redefines a logical symbol and assigns the symbol a true or false value.
.ASKN	Defines or redefines a numeric symbol and assigns the symbol a numeric value.
.ASKS	Defines or redefines a string symbol and assigns the symbol a character string value.

SUMMARY OF INDIRECT DIRECTIVES

.ERASE	Deletes all local or global symbol definitions or a single global symbol definition.
.SETT/.SET	Defines or redefines a logical symbol and assigns the symbol a true or false value.
.SETN	Defines or redefines a numeric symbol and assigns the symbol a numeric value.
.SETD/.SETO	Redefines the radix of a numeric symbol.
.SETL	Defines or redefines a logical symbol and assigns the symbol a true or false value.
.SETS	Defines or redefines a string symbol and assigns the symbol a character string value.
.TRANSLATE	Attempts to translate a logical name and return its equivalence value.

File Access

.DATA	Specifies a single line of data to output to a file.
.CHAIN	Closes the current indirect file and uses commands from another file.
.CLOSE	Closes a file.
.OPEN	Creates and opens an output data file (if the file exists, creates a new version and opens it).
.OPENA	Opens an existing file and appends subsequent data (does not create a new version). Defaults to .OPEN if the file does not exist.
.OPENR	Opens a file for reading with the .READ directive.
.PARSE	Parses strings into substrings.

SUMMARY OF INDIRECT DIRECTIVES

.READ	Reads a record from a file into a specified string variable.
Logical Control	
.BEGIN	Marks the beginning of a Begin-End block.
.END	Marks the end of a Begin-End block.
.EXIT	Terminates processing of either Indirect or the current command file, returns control to the invoking terminal or to the previous level, and optionally sets special symbol <EXSTAT> value.
.GOSUB	Calls a subroutine within the command file.
.GOTO	Branches to a label within the command file.
/	Defines logical end-of-file. Terminates file processing and exits. This directive is equivalent to the .STOP directive.
.ONERR	Branches to a label upon detecting a specific Indirect error condition.
.RETURN	Effects an exit from a subroutine and returns to the line immediately following the subroutine call.
.STOP	Terminates indirect command file processing and optionally sets Indirect exit status. This directive is equivalent to the logical end-of-file (/) directive.
Logical Tests	
.IF	Determines whether or not a symbol satisfies a condition.
.IFACT/.IFNACT	Determines whether or not a task is active.
.IFDF/.IFNDF	Determines whether or not a symbol is defined.

SUMMARY OF INDIRECT DIRECTIVES

.IFENABLED/ .IFDISABLED	Tests the .ENABLE/.DISABLE options.
.IFINS/.IFNINS	Determines whether or not a task is installed in the system.
.IFLOA/.IFNLOA	Determines whether or not a device driver is loaded.
.IFT/.IFF	Determines whether a logical symbol is true or false.
.TEST	Tests the length of a string symbol or locates a substring.
.TESTDEVICE	Returns information about a device in the system.
.TESTFILE	Determines if a specified file exists and determines the physical device associated with a logical device name (performs device translation).
.TESTPARTITION	Returns information about a partition in the system.
.TESTSYSTEM	Returns information about the presence of certain operating system features.

Enable or Disable an Operating Mode

.ENABLE/.DISABLE	Enables or disables control of the following modes:

substitution
timeout parameter
lowercase
terminal attachment
data
global symbol
symbol radix
command line echo
command display
field display
command passing to DCL
file deletion
truncate error suppression
escape
CTRL/Z exit
overflow

SUMMARY OF INDIRECT DIRECTIVES

Increment or Decrement Numeric Symbols

.INC	Adds one to a numeric symbol's value.
.DEC	Subtracts one from a numeric symbol's value.

Execution Control

.DELAY	Delays the execution of an indirect command file for a specified period of time.
.PAUSE	Is a no-op under the PRO/Tool Kit. It does not suspend execution of Indirect.
.WAIT	Is a no-op under the PRO/Tool Kit. It has no effect.
.XQT	Is a no-op under the PRO/Tool Kit. It passes the DCL command line to DCL. Indirect does not proceed until that command is complete.

Special Purpose

.FORM	Provides access to the PRO/FMS form driver, allowing FMS commands to be passed to FMS.

4.4 SYMBOLS

The Indirect Command Processor enables you to define symbols. These symbols can then be tested or compared to control flow through the indirect command file. They may also be substituted in DCL commands, data records for data files, or comments to be displayed on the screen.

Symbol names are ASCII strings from one through six characters in length. They must start with a letter (A through Z) or a dollar sign ($). The remaining characters must be alphanumeric or a dollar sign.

There are three symbol types:

- Logical

- Numeric
- String

A logical symbol has either a true or false value.

A numeric symbol can have a numeric value in the range of 0 through 65535 (decimal). The symbol can be defined to have either a decimal or octal radix. The radix is relevant only when the symbol is substituted.

A string symbol has as its value a string of ASCII characters, with a length of 0 through 132 (decimal) characters.

A symbol's type (logical, numeric, or string) is defined by the first assignment directive that assigns a value to the symbol. Assignment directives can assign:

- A true or false value to define a logical symbol (defined by .ASK, .SETL, .SETT, or .SETF).
- An octal or decimal number to define a numeric symbol (defined by .ASKN, .SETN, .SETO, or .SETD).
- A character string to define a string symbol (defined by .ASKS, .READ, or .SETS).

4.4.1 Special Symbols

Indirect automatically defines special symbols dependent upon specific system characteristics and the replies to queries presented during command file execution. As with symbols, special symbols can be compared, tested, or substituted and consist of three types: logical, numeric, and string. All special symbols have a common format--angle brackets (<>) enclose the special symbol name.

4.4.1.1 Special Logical Symbols - The special logical symbols are assigned a true or false value under the following circumstances:

Symbol	Value
<ALPHAN>	Set to true if last string entered in response to an .ASKS directive or tested with a .TEST directive contains only alphanumeric characters. An empty string also sets <ALPHAN> to true.
<DEFAUL>	Set to true if answer to last query was defaulted (the <RETURN> key was pressed once) or a timeout occurred.
<EOF>	Set to true if the last .READ or .ASKx directive resulted in reading past the end of the file. Otherwise, <EOF> is set to false.
<ERSEEN>	Set to true if any of the following conditions are true. • <FILERR> is less than 0, if a negative error code was returned • If an exit status <EXSTAT> value worse than <WARNING> was returned • <EOF> is set to true • If you used the command line .SETT <ERSEEN> The command line .SETF <ERSEEN> sets the following conditions: • <FILERR> is set to 0 • <EXSTAT> is set to 0 • <EOF> is set to false
<ESCAPE>	Set to true if last question was answered with an ALTmode or ESCape. Otherwise, <ESCAPE> is set to false.
<FALSE>	Logical constant used for comparisons with the .IF directive or as a default for the .ASK directive.
<NUMBER>	Set to true if the last string entered in response to an .ASKS directive or tested with a .TEST directive contains only numeric characters. An empty string also sets <NUMBER> to true.

<OCTAL> Set to true if the answer to the last .ASKN directive or the radix of the numeric symbol tested in the last .TEST directive is octal or if the last string tested with a .TEST directive contained all numeric characters in the range 0 through 7.

<PRIVIL> Set to true if the current user is privileged. Its value is derived from the flag contained in the terminal data base. The flag is set when IND executes.

<RAD50> Set to true if last string entered in response to an .ASKS directive or tested with a .TEST directive contains only Radix-50 characters. Radix-50 characters are the uppercase alphanumeric characters plus period (.) and dollar sign ($). A blank is not a Radix-50 character in this context. An empty string also sets <RAD50> true.

<TIMOUT> Set to true if timeout mode is enabled and the last .ASKx directive timed out waiting for a user response.

<TRUE> Logical constant used for comparisons with the .IF directive or as a default for the .ASK directive.

4.4.1.2 Special Numeric Symbols - The special numeric symbols are assigned the following values:

Symbol	**Value**
<ERRCTL>	Controls the way in which Indirect processes errors. The symbol is treated as an 8-bit mask. For each class of error that a user's .ONERR target routine processes, the appropriate bit is set in the mask. If the bit is clear, Indirect exits after printing the error information. See Section 4.34 for details on .ONERR.

If the eighth bit, the sign bit or 200 (octal), is set, Indirect does not print any information about the error.

The initial default value for <ERRCTL> is 1, which implies that only class 1 errors can be handled with an .ONERR address, and that error messages will be printed. Note that if you attempt to trap errors other than default class 1, processing cannot continue in most cases. The error service routine is limited to a fatal error message and .EXIT. The internal state of Indirect is indeterminate in all but class 1 error cases.

See Section 4.56.2 for a list of error messages and their asigned values.

<ERRNUM> Assigned the class number of an error that Indirect has finished processing. This value can be used for processing specific error types with an .ONERR routine.

<ERRSEV> Assigned the error severity mask associated with the error that Indirect has finished processing. This bit mask corresponds to the bit mask <ERRCTL> used to control the processing.

<EXSTAT> Assigned a numeric value depending upon the status of the last DCL command executed. User programs can generate any 16-bit value as an exit status. Most DIGITAL-supplied programs, however, exit with one of the following status values:

Value

0 Warning
1 Success
2 Error
4 Severe Error

<FILERR> Assigned the FCS-11 or I/O driver status code resulting from a .TESTFILE, .OPENx, or .READ directive operation. See Appendix B for a description of the codes.

<FILER2> Assigned the second error code word that is returned by the FMS-related .FORM command.

<FORATT> Assigned the octal value of the file attributes that were used to open the data files.

<MEMSIZ> Assigned the value of the current system memory size in K words (K is 1024.).

<SPACE> Assigned the number, in octal, of free bytes in the internal symbol table for Indirect. The number does not reflect the amount of space that could be gained by the automatic extension of the Indirect task.

<STRLEN> Assigned the length of the string entered in response to the last .ASKS directive or the string tested by the last .TEST directive.

<SYMTYP> Assigned the numeric code for the type of symbol tested with a .TEST directive. The symbol types have the following code numbers:

```
String  - 4
Numeric - 2
Logical - 0
```

<SYSTEM> Assigned an octal number to represent the operating system on which Indirect is running. The value 11 is for the P/OS system.

<SYUNIT> Assigned the unit number of the user's default device (SY:).

<TICLPP> Assigned the current page length setting for the screen.

<TICWID> Assigned the current page width setting for the screen.

<TISPED> Assigned the baud rate for transmitting characters from the host system to the terminal. Indirect attemtps to determine the baud rate when you first invoke it. This information is useful for determining the quality and quantity of information to be transmitted. The following list gives the octal value that corresponds to each baud rate.

Octal Value	Baud	Octal Value	Baud
1 or 0	0	13	1200
2	50	14	1800
3	75	15	2000
4	100	16	2400
5	110	17	3600
6	134	20	4800
7	150	21	7200
10	200	22	9600
11	300	23	EXTA
12	600	24	EXTB

<TITYPE> Assigned the terminal type of the terminal from which Indirect is running. The Professional terminal type is 35.

4.4.1.3 Special String Symbols - The special string symbols are assigned the following string values.

Symbol **Value**

<ACCOUN> (RSX-11M-PLUS and P/OS systems only.) Assigned certain accounting information from a user's accounting block (UAB). The information is in the following format (note the trailing comma):

unam,sessid,accnum,cpu,dir,qio,tas,atsks

unam The first 14 characters of the user name as it appears in the system account file.

sessid Session ID, null on P/OS.

accnum Account number, null on P/OS.

cpu CPU ticks used since login, null on P/OS.

dir Number of system directives issued since login, null on P/OS.

qio Number of QIO directives issued since login, null on P/OS.

tas Number of tasks run since login, null on P/OS.

atsks	Number of active tasks, null on P/OS
<CLI>	Assigned the acronym (3 through 6 letters) of the current command line interpreter. This symbol is fixed at "DCL" under the PRO/Tool Kit.
<DATE>	Assigned the current date. The date format is dd-mmm-yy.
<DIRECT>	Assigned the name of the current default directory; the format is [ddddddddd].
<EXSTRI>	Contains the string results from a more deeply nested indirect command file. The results are sent to the calling command file. Cleared after each DCL command.
<FILATR>	Assigned the seven words of file attribute information contained in the FCS-11 File Descriptor Block (FDB).
<FILSPC>	Assigned the fully-qualified specification for the file referred to with the last .OPEN, .OPENA, .OPENR, or .TESTFILE directive operation, or in the last specification for a nested command file.
<LOGDEV>	Assigned the device name and unit number of the user's login account on multiuser protection systems. On other systems, <LOGDEV> is assigned SY0: at the time IND is invoked.
<LOGUIC>	Assigned the login UIC of the current user on multiuser protection systems; the format is [ggg,mmm]. On other systems, <LOGUIC> is assigned the current default UIC.
<NETNOD>	Assigned the DECnet node name of the system. If the asystem is not on the DECnet system, <NETNOD> is assigned the string PRO.

<SYSDEV>	Assigned the physical name for the boot device for the system. The device name is in the form ddn.
<SYDISK>	Assigned the device mnemonic (two letters) of the user's default device (SY:); format is dd.
<SYSID>	Assigned the operating system's baselevel number.
<TIME>	Assigned the current time; format is hh:mm:ss.
<UIC>	Assigned to default UIC.

4.4.2 Numeric Symbols and Expressions

A numeric symbol is a string of digits representing a value in the range of 0 through 177777 (octal) (0 through 65535 decimal if immediately followed by a period or if decimal mode has been enabled). If an arithmetic operation yields a result outside of this range, a fatal error occurs and the following message is displayed:

```
IND -- NUMERIC UNDER- OR OVERFLOW
```

A numeric symbol or constant may be combined with another numeric symbol or constant by a logical or arithmetic operator to form a numeric expression. Arithmetic operators are used to add (+), subtract (-), multiply (*), and divide (/). Logical operators are the inclusive OR (!), logical AND (&), and NOT (#). No embedded blanks or tabs are permitted between operators.

Numeric expressions are evaluated from left to right unless parentheses are used to form subexpressions that are evaluated first. For example, the directive lines:

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+N2*4
```

assign numeric symbol N3 the value 24 (octal), whereas the following directive lines assign numeric symbol N3 the value 16 (octal):

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+(N2*4)
```

Numeric expressions are permitted as second operands in numeric .IF and .SETN directives. They are also permitted as range and default arguments in .ASKN and .ASKS directives. The directives .EXIT and .STOP permit numeric expressions to represent exit status.

With each numeric symbol, Indirect associates a radix -- either octal or decimal. The radix of a numeric symbol changes each time the symbol is assigned a new value. If you use a numeric expression to assign a new value to a symbol and all operands in the expression are octal, then the symbol is set to octal. If any operand in the expression is decimal, the symbol is set to decimal. For example:

```
.SETN N2 3.                ! N2 is decimal
.SETN N3 N1+3              ! N3 is octal
.SETN N3 N1+3.             ! N3 is decimal
.SETN N3 N1+N2             ! N3 is decimal
```

You can also assign a new value to a symbol with the .ASKN directive.

The .SETO and .SETD directives allow you to change the radix of a numeric symbol without changing the value of the symbol. For example:

```
.SETN N1 10.               ! N1 = 10 decimal
.SETO N1                   ! N1 = 12 octal
```

The radix of a numeric symbol does not affect arithmetic operations or comparisons. The radix is important only when substituting a numeric symbol into a string. If the radix of the symbol is octal, the value of the symbol is substituted into the string as an octal number. If the radix is decimal, the value is substituted as a decimal number. For example:

```
.SETN N1 10.               ! N1 = 10 decimal
; N1 = 'N1'                ! Displayed as ; N1 = 10
.SETO N1                   ! Make N1 octal
; N1 = 'N1'                ! Displayed as ; N1 = 12
```

If you substitute a numeric symbol into a string and the substituted number is decimal, a period (.) following the symbol name causes a trailing period to be included in the string (following the substituted number). For example:

```
.SETN N1 10.               ! N1 = 10 decimal
; N1 = 'N1.'               ! Displayed as ; N1 = 10.
.SETO N1                   ! Make N1 octal
; N1 = 'N1.'               ! Displayed as ; N1 = 12
```

You can also force a numeric symbol to be substituted as an octal or decimal number by using a substitution format control string. For example:

```
.SETN N1 10.                    ! N1 = 10 decimal
; N1 = 'N1%D'                   ! Displayed as ; N1 = 10
; N1 = 'N1%O'                   ! Displayed as ; N1 = 12
```

4.4.3 String Symbols, Substrings, and Expressions

A string constant is a string of any printable characters enclosed by quotation marks. Empty strings are also permitted. The number of characters cannot exceed 132 (decimal). For example:

```
"ABCDEF"
   ""
```

String symbols may have the value of any string constant. The value is assigned by a .SETS or .ASKS directive. For example, the directive statements:

```
.SETS   S1   "ABCDEF"
.SETS   S2    S1
```

assign string symbol S2 the value of string symbol S1 (that is, ABCDEF).

A substring facilitates the extraction of a segment from the value of a string symbol. You can use substrings only in second operands of .SETS and .IF directives. For example, the directive statements:

```
.SETS   S1   "ABCDEF"
.SETS   S2    S1[1:3]
```

assign string symbol S2 the value of string symbol S1 beginning at character one and ending at character three (that is, ABC).

You can also use the syntax [n:*] to extract the characters from position n to the end of the string. For example, the directive statements:

```
.SETS  S1  "ABCDEF"
.SETS  S2  S1[3:*]
```

assign string symbol S2 the value "CDEF".

You can combine a string constant, symbol, or substring with another string constant, symbol, or substring by the string concatenation operator (+) to form a string expression.

String expressions are permitted as second operands in .SETS and .IF directives where the first operand is a string symbol. For example, the directive statements:

```
.SETS   S1   "A"
.SETS   S2   "CDEF"
.SETS   S3   S1+"B"+S2[1:3]
```

assign string symbol S3 the value of the concatenation of string symbol S1, string constant "B", and the first three characters of string symbol S2 (that is, ABCDE).

4.4.4 Reserved Symbols

Parameters for a command file can be passed to Indirect for processing. The parameters are stored in the following reserved local symbols:

```
P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, COMMAN
```

The symbol COMMAN contains everything in the issuing command line, including the specification for the command file itself. The symbols P0 through P9 contain individual elements of the command line. The elements are delimited by spaces.

With the .GOSUB directive any parameters to the right of the label and to the left of a comment are transferred to the symbol COMMAN. The value of COMMAN can then be parsed to obtain formal call parameters.

4.4.5 Symbol Value Substitution

Substitution can occur in any command line. Indirect can use the values assigned to logical, numeric, string, or special symbols by replacing a normal parameter (for example, a device unit) with the symbol name enclosed in apostrophes (for example, 'DEV'). When a previous directive has enabled substitution mode (.ENABLE SUBSTITUTION), Indirect replaces the symbol name enclosed in apostrophes with the value assigned to the symbol.

When Indirect encounters an apostrophe, it treats the subsequent text, up to a second apostrophe, as a symbol name. Indirect then searches the table of symbols for the corresponding symbol and

substitutes the value of the symbol in place of the symbol name and surrounding apostrophes in the command line.

For example, the first three lines in the following example appear in an indirect command file. When the processor executes these lines, it displays the last two lines at the entering terminal's screen.

```
.ASKS DEV MOUNT ON DEVICE?
.ENABLE SUBSTITUTION
MOUNT 'DEV'

$ * MOUNT ON DEVICE?  [S]:  DZ2:
$ MOUNT DZ2:
```

DZ2: was entered in response to the displayed question. This reply assigned the string value DZ2: to string symbol DEV. Then when Indirect read:

```
MOUNT 'DEV'
```

it substituted for 'DEV' the value assigned to DEV (that is, DZ2:). If substitution mode was not enabled, Indirect would simply have passed the line to DCL as it appeared in the command file (that is, MOUNT 'DEV').

If substitution mode is enabled (it is enabled by default, if you have not disabled it), an apostrophe signals the beginning of a string symbol. Thus, to include a single quote as text within a command line (rather than as the start of a symbol), you must replace the single quote with two contiguous apostrophes ('').

If substitution mode is enabled, Indirect displays the following command file line:

```
!  DON''T SHOOT

        as

!  DON'T SHOOT
```

4.4.5.1 Substitution Format Control - *The conversion of numeric* values to strings and the placement of string and logical values in a substitution operation can be controlled with a format control string. The control string is in the following form:

```
...'symbol%controlstring'...
```

The control string begins with the percent sign (%) and ends with the second of the two apostrophes that denote the substitution operation. The control string consists of one or more of the following characters:

C Compress leading, embedded, and trailing blanks, and remove embedded nulls.

D Force the conversion of a numeric symbol to decimal.

O Force the conversion of a numeric symbol to octal.

S Perform signed conversion for a numeric symbol.

M Perform magnitude conversion for a numeric symbol.

Z Return leading zeros for a positive numeric value.

Rn Right-justify the resulting string, truncating to 'n' decimal characters if necessary.

Ln Left-justify the resulting string, truncating to 'n' decimal characters if necessary.

X Convert the variable to Radix-50 characters.

V If the symbol being substituted is numeric, convert the low byte to its equivalent ASCII character and substitute it.

If the symbol being substituted is a string, convert the first character to its octal representation and substitute it.

Indirect does not perform a consistency check on the control string. If you specify conflicting format characters, Indirect uses the last one specified.

4.5 SWITCHES

The indirect command processor accepts five switches: /TR, /DE, /CLI, /LB, and /LO.

Switch **Function**

/[NO]TR Displays a trace of the indirect command file on the screen. This function is useful for debugging an

indirect command file. Each command line, including Indirect directive statements, is displayed. The default is /NOTR.

/[NO]DE Indicates that the indirect command file is to be deleted when its processing is complete unless a logical end-of-file (/) or .STOP directive is encountered before the end of the file. The default is /NODE.

/[NO]CLI Passes commands not processed by Indirect to DCL. The default is /CLI.

/LB Indicates that the specified file is a universal library of command procedures and that the specified module is the procedure to be executed.

When command procedures, which are indirect command files, are inserted into a universal library with the Librarian command, you can subsequently reference them with /LB:module.

Command libraries are built by creating a universal library and inserting command files into it. You can then reference the procedures in the library with the following command line:

```
@command-library/LB:module
```

The default file type for a command library is .CLB.

If you do not specify a module (@command-library/lb), Indirect attempts to locate a module called .MAIN..

See the example below.

The command file PARAM.CMD contains parameter definitions for the .SETN directive and the command file SYSPRC.CMD contains system-specific procedures. Use the following command lines to create the command library and enter the command files into it:

```
$ LIBR/CREATE/UNIVERSAL SYSTART.CLB
$ LIBR/INSERT SYSTART.CLB PARAM.CMD
$ LIBR/INSERT SYSTART.CLB PROCED.CMD
```

You can then use the following command lines to reference the command library modules:

```
@SYSTART/LB:PARAM   !Define global symbol
```

```
@SYSTART/LB:PROCED  !Run init procedure
```

DIGITAL supplies a library of command procedures. The library is LB:[1,2]INDSYS.CLB and it contains the following procedures:

INDCFG Displays the current build parameters for the running Indirect task.

```
@LB:[1,2]INDSYS/LB:INDCFG
```

INDDMP Dumps to the screen the contents of the Indirect symbol table.

```
@LB:[1,2]INDSYS/LB:INDDMP/LO
```

INDPRF A sample procedure to fully parse file-name strings.

To return a parsed string in <EXSTRI>:

```
@LB:[1,2]INDSYS/LB:INDPRF DW1:[DATA]st.dat;2
```

INDSFN Returns system configuration information.

To return in <EXSTRI> whether the Floating Point unit is present:

```
@LB:[1,2]INDSYS/LB:INDSFN FPP
```

INDVFY Displays the values of all of the special symbols.

```
@LB:[1,2]INDSYS/LB:INDVFY
```

QIOERR Returns a string expansion of the <FILERR> error codes.

To returns in <EXSTRI> a string expansion for the code.

```
@LB:[1,2]INDSYS/LB:QIOERR -33.
```

FMSDEM Demonstrates the PRO/FMS interface incorporated into Indirect.

See the description of the .FORM command.

.MAIN Demonstrates how to use the command library.

```
@LB:[1,2]INDSYS/LB
```

.INDEX Displays an index of the procedures in the library.

```
@LB:[1,2]INDSYS/LB:.INDEX
```

The following command line shows the format for invoking a command procedure in the library:

```
@LB:[1,2]INDSYS/LB:procedurename
```

/[NO]LO Indicates that when a new command file is executing, it can have access to the local symbols created by its calling command file. The default is /NOLO.

4.6 USING INDIRECT DIRECTIVES

Directives must be separated from their arguments and from DCL-specific commands by at least one space. Only one directive per command line is allowed.

You can insert any number of blanks and horizontal tabs in three places of a command line:

- At the start of the command line
- Immediately following the colon (:) of a label
- At the end of the command line

This allows you to format the command files for readability. The recommended procedure is to begin labels in column one and everything else in column nine (after one horizontal tab).

An important exception is the lines processed between .ENABLE and .DISABLE DATA directives; no blanks or tabs are removed from these lines. For example:

```
        .IFT Z .GOTO 10
        MACRO/NOLIST TEST
.10:    LINK TEST
        .OPEN DATFIL
        .DATA XXXXX
.ENABLE DATA
This is data
that goes into
the data file.
.DISABLE DATA
        .GOTO 20
```

Note that the .DISABLE DATA statement must begin in column 1 or Indirect will place it in the data file.

4.6.1 Logical Tests

A number of .IF directives make tests; if the test is true, Indirect processes the remainder of the command line. Logical tests can be combined to form compound logical tests by using the .AND and .OR directives.

An implied .AND is effected when more than one .IF appears on the same line without being separated by an .AND directive.

When using .AND and .OR, the .AND directive takes precedence over the .OR directive, as shown in the following example:

```
.IFT A .OR .IFT B .AND .IFT C .GOTO D
```

That is, Indirect reads the line as:

```
.IFT A .OR (.IFT B .AND .IFT C) .GOTO D
```

Examples follow:

```
.IFT A .AND .IFF B  .GOTO HELP
```

If the logical symbol A is true and the logical symbol B is false, control passes to the line containing the label .HELP:.

```
.IFT A .IFF B  .GOTO HELP
```

Same effect as the previous directive (.AND implied).

```
.IFT A .OR  .IFF B RUN TESTER
```

If the logical symbol A is true or if the logical symbol B is false, the RUN command is issued.

The following sections describe all of the Indirect directives.

4.7 / - DEFINE LOGICAL END-OF-FILE

The logical end-of-file directive (/) terminates file processing and exits. The message

```
$ @ <EOF>
```

is then displayed.

Format

```
/
```

is the first nonblank character on a line.

You can use this directive at any location in the command file to quickly terminate file processing, but care should be taken to avoid an inadvertent exit.

Example

```
          .ASK CONT DO YOU WISH TO CONTINUE
          .IFT CONT .GOTO 100
          /
.100:
```

4.8 .ASK - ASK A QUESTION AND WAIT FOR A REPLY

The .ASK directive prints a question on the screen, waits for a reply, and sets a specified logical symbol to the value of true or false, depending on the reply. If the symbol has not already been defined, Indirect makes an entry in the symbol table. If the symbol has been defined, Indirect resets its value (true or false) in accordance with the reply. Indirect exits with a fatal error if the symbol was previously defined as a string or numeric symbol.

Formats (brackets are required syntax)

```
.ASK ssssss txt-strng
.ASK [default:timeout] ssssss txt-strng
.ASK [:timeout] ssssss txt-strng
```

where:

ssssss	The 1- to 6-character symbol to be assigned a true or false value.
txt-strng	The question or prompt that Indirect displays.
default	The default response; used if the question is answered with an empty line (null) or if timeout occurs. The default can be <TRUE> or <FALSE> or another logical variable.
timeout	The timeout count. Indirect waits this long for a response, then applies the default answer. The format for timeout is nnu, where nn is the decimal number of time units to wait and u is T (ticks), S (seconds), M (minutes), or H (hours). The timeout count is used only if timeout mode is enabled (.ENABLE TIMEOUT).

The entire .ASK statement must fit on one command line.

Note that if you omit the default value but specify a timeout count, the colon is required for positional identification.

When executing an .ASK directive, Indirect displays txt-strng prefixed by an asterisk and suffixed with "? [Y/N]:". Indirect recognizes five answers:

1. Y<RET>

   Set symbol ssssss to true.

2. N<RET>

   Set symbol ssssss to false.

3. <RET>

   Set symbol to false or to user-specified default value. <RET> indicates the <RETURN> key.

4. <ESC>

   Set symbol ssssss to true and set the special logical symbol <ESCAPE> to true only if escape recognition has been enabled. <ESC> indicates the <ESCAPE> key.

NOTE

If escape sequence recognition is not enabled, the <F11> key (ESC) functions as the escape key

5. <CTRL/Z>

   If CONTROL-Z mode is enabled, set <EOF> to true and proceed, or else exit immediately.

Example

The directive statement:

```
.ASK INSFOR DO YOU WANT TO INSTALL FORTRAN
```

displays

```
$ * DO YOU WANT TO INSTALL FORTRAN? [Y/N]:
```

on the screen. Symbol INSFOR will be set to true or false after you type Y, N, the <RETURN> key, or the <ESCAPE> key (if escape recognition is enabled).

4.9 .ASKN - ASK FOR DEFINITION OF A NUMERIC SYMBOL

The .ASKN directive prints on the terminal screen a request for a numeric value, waits for it to be entered, optionally tests the range for the numeric response and/or applies a default value, and sets the specified symbol accordingly.

If the symbol has not previously been defined, Indirect makes an entry in the symbol table. If the symbol has already been defined, Indirect resets its value in accordance with the reply. Indirect exits with a fatal error if the symbol was previously defined as a logical or string symbol.

Formats (brackets are required syntax)

```
.ASKN ssssss txt-strng
.ASKN [low:high:default:timeout] ssssss txt-strng
```

where:

ssssss	The 1- to 6-character symbol to be assigned a numeric value.
txt-strng	The question or prompt that Indirect displays.
low:high	A numeric expression giving the range for the response.
default	A numeric expression or symbol giving the default value.
timeout	The timeout count. Indirect waits this long for a response, then applies the default answer. The format for timeout is nnu, where nn is the decimal number of time units to wait and u is T (ticks), S (seconds), M (minutes), or H (hours). The timeout count is valid only if timeout mode is enabled (.ENABLE TIMEOUT).

The entire .ASKN statement must fit on one command line.

Note that if you omit any of the parameters within the square brackets, any preceding colons are required for positional identification.

The command line cannot exceed 132 (decimal) characters in length. When executing an .ASKN directive, Indirect displays txt-strng prefixed by an asterisk and suffixed with [O]: to indicate that the response is considered as octal or [D]: to indicate that the response is considered as decimal. The reply

must be a number either within the specified range or in the range 0 through 177777 (octal) (by default) or 0 through 65535 (decimal).

If the response is outside the specified range, the message

```
IND -- VALUE NOT IN RANGE
```

is displayed and the query repeated.

If an arithmetic operation yields a result greater than 177777 (octal) when computing the actual value of any of the arguments (low, high, or default), a fatal error occurs and the message

```
IND -- NUMERIC UNDER- OR OVERFLOW
```

is displayed.

If the response is an empty line (null) and a default value (default) was not specified, Indirect applies a default of 0. Note that in this case, the range, if specified, must include 0.

The response may be either octal or decimal; a leading pound sign (#) forces octal, a trailing period (.) forces decimal. In the absence of both, Indirect applies a default radix. The default radix is decimal if either the range or default values are decimal expressions (followed by a period). Otherwise, the default radix is octal (unless decimal mode has been enabled). Indirect displays the default type as either [O] or [D].

To force a default decimal radix without specifying a range argument, use the following construction:

```
.ASKN [::0.] A ENTER VALUE
```

or

```
.ENABLE DECIMAL
.ASKN A ENTER VALUE
```

Examples

- The directive statement:

```
.ASKN  SYM  DEFINE NUMERIC SYMBOL A
```

displays

```
$ * DEFINE NUMERIC SYMBOL A [O]:
```

on the terminal screen where:

[O] is the default radix (octal).

Indirect then defines symbol SYM according to the reply entered.

- The directive statement:

```
.ASKN [2:35:16:20S] NUMSYM DEFINE NUMERIC SYMBOL A
```

displays

```
$ * DEFINE NUMERIC SYMBOL A [O R:2-35 D:16 T:20S]:
```

in the format [x R:low-high D:default T:timeout]

where:

x	O if the default radix is octal or D if it is decimal.
R:low-high	The specified range.
D:default	The specified default.
T:timeout	The specified timeout count before the default answer is applied.

Indirect then checks that the response string is in the specified range.

- The directive statement:

```
.ASKN [NUMSYM+10:45:NUMSYM+10] SYM DEFINE NUMERIC SYMBOL B
```

displays (assuming the value of 16 octal for NUMSYM):

```
$ * DEFINE NUMERIC SYMBOL B [O R:26-45 D:26]:
```

4.10 .ASKS - ASK FOR DEFINITION OF A STRING SYMBOL

The .ASKS directive prints on the terminal screen a request for a string value to define a specified symbol and optionally tests that the number of characters in the response string falls within the specified range.

If the symbol has not previously been defined, Indirect makes an entry in the symbol table. If the symbol has already been defined, Indirect resets its value in accordance with the reply. Indirect exits with a fatal error if the symbol was defined previously as a logical or numeric symbol. If the number of characters is out of the specified range, the message

```
IND -- STRING LENGTH NOT IN RANGE
```

is displayed and the question repeated.

Formats (brackets are required syntax)

```
.ASKS ssssss txt-strng
.ASKS [low:high:default:timeout] ssssss txt-strng
```

where:

ssssss	The 1- to 6-character symbol to be assigned a string value.
txt-strng	The question or prompt that Indirect displays.
low:high	A numeric expression giving the range for the number of characters permitted in the response string.
default	A string expression or symbol giving the default value.
timeout	The timeout count. Indirect waits this long for a response, then applies the default answer. The format for timeout is nnu, where nn is the decimal number of time units to wait and u is T (ticks), S (seconds), M (minutes), or H (hours). The timeout count is valid only if timeout mode is enabled (.ENABLE TIMEOUT).

The entire .ASKS statement must fit on one command line.

Note that if you omit any of the parameters within the square brackets, any preceding colons are required for positional identification.

When executing an .ASKS directive, Indirect displays txt-strng prefixed by an asterisk (*) and suffixes it with [S]:. The reply must be an ASCII character string.

Examples

- The directive statement: .ASKS NAM PLEASE ENTER YOUR NAME

  displays

  ```
  $ * PLEASE ENTER YOUR NAME [S]:
  ```

  on the terminal screen. Indirect then defines symbol NAM according to the string reply entered.

- The directive statement:

  ```
  .ASKS [1:15::10S] MIDNAM PLEASE ENTER YOUR MIDDLE NAME
  ```

  displays

  ```
  $ * PLEASE ENTER YOUR MIDDLE NAME [S R:1-15 T:10S]:
  ```

  in the format [S R:low-high T:timeout]

  where:

S	The symbol type (string).
R:low-high	The specified range for number of characters.
T:timeout	The specified timeout count.

4.11 .BEGIN - BEGIN BLOCK

The .BEGIN directive marks the beginning of a Begin-End block. The block must be terminated with an .END directive.

Labels and local symbols defined following the .BEGIN directive are local to the block instead of being used throughout the entire command file. Therefore, labels and local symbols defined inside a block lose definition outside the block. (Labels and symbols defined outside a block retain definition throughout the file.) Labels and symbols defined outside a block and then modified within the block, however, assume and retain the value assigned in the block.

Labels and local symbols defined within a block lose definition with an .ERASE LOCAL directive statement or with the .END directive.

.BEGIN must be the only directive on a command line. For example, the .BEGIN directive cannot appear on the same line as an .IF directive.

Format

```
.BEGIN
```

as the only directive on the line.

4.12 .CHAIN - CONTINUE PROCESSING USING ANOTHER FILE

The .CHAIN directive closes the current file, erases all local symbols, clears any .ONERR arguments, empties the direct access label cache, and continues processing using command lines from another file. However, the .CHAIN directive does not close data files or change the nested-file level.

Format (brackets not part of syntax)

```
.CHAIN filename[/switches]
```

where filename is the name of the file that contains the new command lines. The /switches are any of the optional switches described in Section 4.5.

Example

The directive statement:

```
.CHAIN OUTPUT
```

transfers control to the file OUTPUT.CMD.

4.13 .CLOSE - CLOSE SECONDARY FILE

The .CLOSE directive closes the secondary file opened by an .OPEN directive.

Format (brackets not part of syntax)

```
.CLOSE [#n]
```

where:

#n	An optional file number in the range 0 to 3. The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

4.14 .DATA - OUTPUT DATA TO SECONDARY FILE

The .DATA directive specifies text that is to be output to a secondary file previously opened by an .OPEN directive.

When Indirect processes the text string that follows the .DATA directive, it ignores a leading space (if present), assuming it to be a separator between the directive and the text string. Any other spaces or tabs are transferred to the data file.

Format (brackets not part of syntax)

```
.DATA [#n] txt-strng
```

where:

txt-strng	The text to be output to the secondary file.
#n	An optional file number in the range 0 to 3. The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

The command line cannot exceed 132 (decimal) characters and the specified text string cannot continue onto the next line. If a secondary file is not open, an error condition exists; Indirect issues an error message and begins error processing.

Example

```
.SETS SEND "THIS IS DATA"
.OPEN TEMP
.DATA 'SEND'
.CLOSE
```

These directives output THIS IS DATA to the secondary file TEMP.DAT (.DAT is the default file type for a data file).

4.15 .DEC - DECREMENT NUMERIC SYMBOL

The .DEC directive decrements a numeric symbol by one. Indirect exits with a fatal error if the symbol was defined previously as a logical or string symbol.

Format

```
.DEC ssssss
```

where:

ssssss The 1- to 6-character numeric symbol.

Example

```
.DEC LOOPCT
```

This directive decrements by 1 the value assigned to the numeric symbol LOOPCT.

4.16 .DELAY - DELAY EXECUTION FOR A SPECIFIED PERIOD OF TIME

The .DELAY directive delays further processing of the file for a specified period of time.

Format

```
.DELAY nnu
```

where:

nn	The decimal number of time units to delay.
u	T - ticks S - seconds M - minutes H - hours

The parameter nn is decimal by default.

If quiet mode is disabled when the .DELAY directive is executed, Indirect issues the message:

```
IND -- DELAYING
```

When the time period expires and the task resumes, Indirect issues the message:

```
IND -- CONTINUING
```

Example

The directive statement:

```
.DELAY 20M
```

delays processing for 20 (decimal) minutes.

4.17 .DISABLE - DISABLE OPTION

The .DISABLE directive disables a specified operating mode previously activated by an .ENABLE directive.

Format

```
.DISABLE option[,option...]
```

The following is a list of the operating modes that can be disabled:

ATTACH	DETACH	GLOBAL	TIMEOUT
CONTROL-Z	DISPLAY	LOWERCASE	TRACE
DATA	ESCAPE	CLI	TRUNCATE
DECIMAL	ESCAPE-SEQ	OVERFLOW	SUBSTITUTION
DELETE		QUIET	

Note that when you disable detach mode from a command file and then request a task or DCL command to display information, the command file may not be able to continue executing. The task or DCL command may need to attach to the terminal to display the information but will not be able to do so because Indirect cannot detach from the terminal.

4.18 .ENABLE - ENABLE OPTION

The .ENABLE directive is used to invoke several operating modes. Each mode is independent of the others; all of them can be active simultaneously. When Indirect starts to process a file, the initial settings are:

ATTACH	enabled	GLOBAL	enabled
CONTROL-Z	disabled	LOWERCASE	enabled
DATA	disabled	CLI	enabled
DECIMAL	disabled	OVERFLOW	disabled
DELETE	disabled	QUIET	disabled
DETACH	enabled	SUBSTITUTION	enabled
DISPLAY	enabled	TIMEOUT	enabled
ESCAPE	disabled	TRACE	disabled
ESCAPE-SEQ	disabled	TRUNCATE	disabled

In attach mode, Indirect attaches to the screen when displaying comment lines. In detach mode, it detaches from the screen when processing command lines. Enabling both of these modes allows you to type CTRL/O to suppress a lengthy comment.

Enabling Control-Z mode allows a command file to detect a CTRL/Z response to a question and continue processing. If Control-Z mode is disabled and you type CTRL/Z in response to an .ASKx question, Indirect exits. If Control-Z mode is enabled, the special symbol <EOF> is set to true and Indirect continues processing the command file.

In data mode, Indirect outputs lines that follow an .ENABLE DATA directive statement to a secondary file. (The .DATA directive sends a single line of text to a secondary file.) To disable data mode, the .DISABLE DATA statement must begin in the first column. Otherwise, Indirect copies the statement itself into the data file. The .ENABLE DATA directive also has an optional argument (#n) that specifies which file the data is to go into. See the description of the .DATA directive for more information.

In global symbol mode, symbol names that begin with a dollar sign ($) are defined as global to all levels of indirect files; once such a symbol has been defined, all levels recognize it. Symbols that do not begin with a dollar sign are recognized only within the level that defines them.

In decimal mode, all numeric symbols are created or redefined by default as decimal instead of as octal.

In delete mode, the current command file is deleted when Indirect processes the last command line in the file.

In display mode, Indirect displays the current fields for the .ASKx directives and @ <EOF>. If display mode is disabled, Indirect displays only the text string for the .ASKx directive and suppresses @ <EOF>.

In CLI mode, commands not processed by Indirect are passed to DCL. CLI mode is equivalent to the functions of the /CLI switch.

In lowercase mode, characters read from the terminal in response to .ASKS directives are stored in the string symbol without lower- to uppercase conversion. The representation of characters is significant when comparing strings since the .IF directive distinguishes between lowercase and uppercase characters.

In substitution mode, Indirect substitutes a string for a symbol. The symbol must begin and end in apostrophes ('symbol'). For example, if the symbol A has been assigned the string value THIS IS A TEST, then every 'A' will be replaced by THIS IS A TEST. When substitution mode is enabled, Indirect performs substitutions in each line before scanning the line for directives and DCL commands.

Escape recognition (.ENABLE ESCAPE) permits the response to an .ASK, .ASKN, or .ASKS directive to be an escape character. A question answered with a single escape character sets the special logical symbol <ESCAPE> to true. The escape character must be used only as an immediate terminator to the question; if one or more characters precede the escape character, an error condition exists. In this case, the message

```
IND -- INVALID ANSWER OR TERMINATOR
```

is printed and the question repeated. Note that if you press the <ESCAPE> key in response to an .ASK directive, the specified logical symbol (ssssss of .ASK ssssss txt-strng) is also set to true.

Escape-sequence recognition (.ENABLE ESCAPE-SEQ) forces Indirect to attach to the terminal for escape-sequence recognition, using the IO.ATT!TF.ESQ I/O function.

In this mode, the result of an .ASKx or .READ statement from the terminal will contain the terminating escape character and escape sequence, as documented in the *P/OS System Reference Manual*.

Overflow mode allows signed arithmetic in numeric expressions. Enabling the mode provides for numeric expressions and operations that otherwise would result in the "Numeric under- or overflow" error message.

In quiet mode, Indirect does not echo DCL command lines or

comments. The command lines are executed normally and, if they return a message or display, the message or display is printed on the screen.

In timeout mode, Indirect uses the timeout parameters specified with the .ASKx directives. Indirect waits for the timeout count to elapse and then applies the default answer to the directives. Timeout mode must be enabled (the default) to use the timeout counts for the .ASKx directives.

In trace mode, command lines that Indirect has processed are displayed on the terminal screen. As each line is processed, it is displayed with its nesting level and an exclamation mark (!). Trace mode is equivalent to the function of the /TR switch.

In truncate mode, Indirect ignores any truncate errors on a .READ directive. A truncate error occurs when a line in a file is too long. If the full record cannot fit within the 132 (decimal)-character limit of the symbol, the record is truncated.

Formats (brackets not part of syntax):

```
.ENABLE option[,option...]

.ENABLE DATA [#n]
```

where:

#n	An optional file number in the range 0 to 3. The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

Examples

- Substitution mode:

```
.ENABLE SUBSTITUTION
.ASKS FIL SPECIFY SOURCE FILE
MACRO 'FIL'
```

When the file is executing, the corresponding lines displayed at the terminal screen are:

```
$ * SPECIFY SOURCE FILE [S]: SOURCE
$ MACRO SOURCE
```

- Control-Z mode:

```
.ENABLE CONTROL-Z
.ASK RESP DO YOU WISH TO CONTINUE
.IFT <EOF> .GOTO CLENUP
.IFF RESP> .GOTO CLENUP
```

If you type CTRL/Z in response to the question, <EOF> is set to true and Indirect transfers to CLENUP.

4.19 .END - END BLOCK

The .END directive marks the end of the Begin-End block. If Indirect encounters more .END directives than .BEGIN directives, command processing terminates and the following message is displayed:

```
IND -- ILLEGAL NESTING
```

Format

```
.END
```

as the only directive on the line.

4.20 .ERASE - DELETE SYMBOLS

The .ERASE directive deletes all local or global symbol definitions, or a specific global symbol definition. When you define a symbol, either locally (by defining a symbol value) or globally (by enabling global symbol mode and preceding the symbol name with a dollar sign ($)), Indirect creates an entry in the symbol table. The .ERASE directive erases either all local or all global entries, or a specific global entry, in the table.

Following an .ERASE directive, you can redefine symbol values as well as symbol type.

Formats

```
.ERASE LOCAL
.ERASE GLOBAL
.ERASE SYMBOL global-symbol
```

An .ERASE LOCAL directive outside of a Begin-End block erases all local symbols defined within the current file.

An .ERASE LOCAL directive within a Begin-End block erases only those local symbols defined within the block.

However, note that the following actions also occur:

1. Local symbols defined within a nested file are erased when that file exits.

2. Local symbols defined within a Begin-End block are erased with .END.

3. Local symbols defined outside of Begin-End blocks are visible, modifiable, and not erasable within a Begin-End block.

An .ERASE GLOBAL, either outside of or within a Begin-End block, erases all global symbols.

An .ERASE SYMBOL global-symbol erases the specified global symbol. (Individual local symbols are not erasable.)

Example

```
.ERASE LOCAL
```

This directive erases all local symbol definitions used in the indirect command file.

```
.ERASE SYMBOL $SWITC
```

This directive erases the single global symbol "$SWITC."

4.21 .EXIT - EXIT CURRENT COMMAND FILE

The .EXIT directive terminates processing of the current command file or Begin-End block and returns control to the previous-level command file or, if the directive is executed within a block, to the line following the .END directive. If the directive is encountered at the uppermost indirect nesting level, Indirect exits and passes control to DCL (see the .STOP directive).

The .EXIT directive also allows you to optionally specify a value to copy into the special symbol <EXSTAT>.

Format (brackets not part of syntax)

```
.EXIT [value]
```

where value is an optional numeric expression copied to <EXSTAT>.

Examples

The following line is in an indirect command file called TEST1:

```
@TEST2
```

The file TEST2.CMD contains the following line:

```
.EXIT
```

When Indirect encounters the .EXIT directive in TEST2, control returns to TEST1.CMD.

If the .EXIT directive in TEST2.CMD includes a numeric expression, for example:

```
.EXIT N+2
```

Indirect evaluates the expression and copies the value into <EXSTAT>.

4.22 .FORM - ACCESS FORM DRIVER

The .FORM command provides access to the PRO/FMS-11 form driver from a command file. You can specify commands in the command file that perform operations such as:

- Directing the display of a form
- Directing the motion of the cursor from field to field
- Acquiring and parsing the user typein

The syntax of the .FORM command parallels the format of the MACRO-11 call interface to FMS-11. For details, see the *FMS-11/RSX Software Reference Manual*, particularly the chapters on form driver operation and the MACRO-11 interface.

Format

```
.FORM FNC,p1,p2,...pn
```

where:

FNC	is a three-letter code indicating which FMS operation is to be performed. These codes are a subset of the codes used in the MACRO-11/FMS interface and are summarized below.
p1...pn	are string or numeric symbols or constants conforming to Indirect syntax rules.

Formal parameters:

LINENUM	The screen line number where form display is to begin.
RETTRM	The name of a numeric variable to contain the code for the terminator typed by the user.
FILENAME	The name of a string variable or string constant naming the file in which the form definitions are stored.
RETNAM	The name of the field completed by the user.
RETINX	The index of the field completed by the user.
TERMINATOR	The code for the terminator to be processed.

VALUE	A string variable or constant to be placed in the indicated field.
FORMNAME	The name of the form to display.
FLDNAME	The name of a field defined in the currently displayed form.
INDEX	In an indexed field, the index referencing the specific field being addressed.
RETVAL	The name of a string variable into which the returned value will be placed.

Examples

String values supplied as input to the .FORM command can be expressed as a constant enclosed in quotation marks or as the name of a previously defined string variable. For example, the following have equivalent results:

```
.FORM OPN,"FMSDEM.FLB"   !define form library filename
```

and

```
.SETS LIBR "FMSDEM.FLB"
.FORM OPN,LIBR           !define form library filename
```

String and numeric values returned as output from the .FORM command are passed as though a .SETS or .SETN command were being executed. This means that the name of the variable to receive the value must be supplied, and that it must either have not been defined or is previously defined as the appropriate string or numeric type. For example:

```
.FORM GET,"CHOICE",,,,FLDVAL ! return value of field
"CHOICE"
```

The Indirect local symbol FLDVAL is defined or redefined as required and contains the string typed by the user to fill the field named CHOICE on the currently displayed form.

Remember that Indirect can handle strings only as long as 132. characters, so that values returned to strings from the form driver must be shorter than that length. This is particularly important with the ALL and RAL commands, which attempt to place the string values of all fields displayed on the form into a single Indirect string variable. A better programming practice would be to use a series of GET commands addressing each

individual field.

A demonstration procedure is included in LB:[1,2]INDSYS.CLB. This procedure library is packaged on the PRO/Tool Kit distribution kit. To execute the demonstration procedure, type the following command line:

```
$ @LB:[1,2]INDSYS.CLB/LB:FMSDEM
```

After the terminal type setting is verified, a temporary copy of the forms library is placed in your directory. The procedure is identical to that provided in MACRO-11 form on the FMS-11/RSX kit. Refer to the *FMS-11/RSX Software Reference Manual* for a complete description of this demonstration.

Using the following DCL command, extract the FMS demonstration procedure and use it as an extended example for building your own command procedures.

```
$ LIBRARIAN/EXTRACT/OUT=FDEM.CMD LB:[1,2]INDSYS.CLB FMSDEM
```

Commands:

CSH -- Clear screen and show form

```
.FORM CSH, FORMNAME [,LINENUM]
```

The form driver clears the entire screen and displays the specified form from the currently open form library. If LINENUM is supplied, it is interpreted as an integer line number which overrides the form starting line number supplied by the forms editor.

SHO -- Show form

```
.FORM SHO, FORMNAME [,LINENUM]
```

The form driver clears only the portion of the screen required for the specified form, then displays the form named in FORMNAME from the currently open forms library. If LINENUM is present, it overrides the starting line number supplied by the forms editor.

GET -- Get value for specified field

```
.FORM GET, FLDNAME [,INDEX [,RETNAM [,RETINX [,RETVAL
                [,RETTRM]]]]]
```

The form driver places the cursor at the initial position of the specified field and accepts input from the keyboard for that field.

ANY -- Return any field value

```
.FORM ANY ,RETNAM ,[RETINX] ,RETVAL [,RETTRM]
```

The form driver waits for the operator to fill any field. The cursor may be positioned in a field that is not display-only. The field name and its resulting value are returned.

ALL -- Return all fields

```
.FORM ALL [,RETVAL [,RETTRM]]
```

After the operator has filled any or all fields of a form and presses the <ENTER> key, the form driver returns all field values as a concatenated string into the string variable RETVAL. Remember that strings may have a maximum length of 132. characters so that the concatenated value may not exceed this length.

DAT -- Get named data from form

```
.FORM DAT, FLDNAME, [INDEX], RETVAL
```

The form driver returns the value from the named data portion of the form. If INDEX is supplied, it is interpreted as the index for the named data value to be returned.

GSC -- Get current line of scrolled area

```
.FORM GSC, FLDNAME, RETVAL [,RETTRM]
```

The form driver returns the contents of the entire current scrolled data line as a concatenated string.

CLS -- Close forms library

```
.FORM CLS
```

The currently open forms library is closed.

OPN -- Open forms library

```
.FORM OPN, FILENAME
```

The forms driver attempts to open the specified file as a library of form definitions. The library is built by the forms utility program.

PSC -- Put to current line of scrolled area

```
.FORM PSC, FLDNAME, VALUE
```

The form driver outputs the data specified to the current line of the scrolled area. The scrolled area is identified by naming in FLDNAME any field in that area.

TRM -- Process field terminator

```
.FORM TRM, [FLDNAME], [VALUE], TERMINATOR [,RETNAM
                [,RETINX]]
```

The user supplies the numeric code for the terminator, processed in TERMINATOR. The form driver then performs cursor and field positioning accordingly. The *FMS-11/RSX Software Reference Manual* describes terminator codes and actions.

PUT -- Put to specified field

```
.FORM PUT, FLDNAME, [INDEX], VALUE
```

This displays the value specified in the named field.

PAL -- Put all fields

```
.FORM PAL [,VALUE]
```

The contents of VALUE are used to fill all fields of the current form. Remember that the VALUE string may have a maximum of 132. characters.

LST -- Output to last line of screen

```
.FORM LST [,VALUE]
```

The form driver clears the last line of the screen and displays the specified string. (The last line is not normally accessible via a form.) This is the only way to display messages in this screen location.

RAL -- Return all fields

```
.FORM RAL, VALUE
```

RTN -- Return value for specified field or for all fields

```
.FORM RTN, [FLDNAME] , [INDEX], RETVAL
```

If the FLDNAME parameter is not present the form driver

returns the concatenated string of the current values for all fields in the form. See the note under the ALL command.

If FLDNAME is present, the form driver returns the value only for the named field.

SPN/SPF -- Supervisor mode control

```
.FORM SPN
.FORM SPF
```

Turn supervisor mode on (SPN) or off (SPF). The form editor permits defining some fields as display only if supervisor mode is off.

4.23 .GOSUB - CALL A SUBROUTINE

The .GOSUB directive saves the current position in an indirect command file and then branches to a label. The label identifies an entry point to a subroutine that is terminated by a .RETURN directive.

When you issue a .GOSUB directive from within a Begin-End block, Indirect saves the current block context and then scans down the file searching for the first occurrence of the subroutine label. Note that during the scan, Indirect ignores any intervening .BEGIN or .END directives. The .RETURN directive restores previous block context. Thus, the subroutine can be contained within a Begin-End block.

The maximum nesting depth for subroutine calls is 8.

Format

```
.GOSUB label parameters
```

where label is the label that designates the first line of a subroutine, but without the leading period and trailing colon. Any parameters to the right of the label and to the left of a comment are transferred to the reserved local symbol COMMAN. The value of COMMAN can then be parsed with the .PARSE directive to obtain formal call parameters.

Example

The directive statement:

```
.GOSUB EVAL
```

transfers control to the subroutine labeled .EVAL:.

4.24 .GOTO - BRANCH TO A LABEL

The .GOTO directive causes a branch from one line in an indirect command file to another. All commands between the .GOTO directive and the specified label are ignored. Branches can go forward or backward in the file.

The target of a .GOTO branch from within a Begin-End block must be contained in that block. The .GOTO directive cannot branch into another block. When Indirect encounters a .GOTO directive within a Begin-End block, it searches for the specified label in that block.

Since Indirect only searches the one Begin-End block, you can use the same label more than once in a command file.

Format

```
.GOTO label
```

where label is the name of the label, but without the leading period and trailing colon.

Example

The directive statement:

```
.GOTO 100
```

transfers control to the line containing the label .100:.

4.25 .IF - TEST IF SYMBOL SATISFIES SPECIFIED CONDITION

The .IF directive compares a numeric or string symbol with another expression of the same type to determine if one of several possible conditions is true. If the condition is satisfied, Indirect executes the remainder of the command line.

When comparing a string symbol with a string expression, Indirect compares the ASCII values of each operand's characters (from left to right) one by one. An operand is considered greater if the first nonequal character has a greater value than the corresponding character in the other operand. Numeric symbols are compared strictly on the basis of magnitude.

Format

```
.IF symbol relop expr directive-statement
```

where:

symbol	The 1- to 6-character numeric or string symbol.
relop	One of the following relational operators: EQ or = - Equal to NE or <> - Not equal to GE or >= - Greater than or equal to LE or <= - Less than or equal to GT or > - Greater than LT or < - Less than
expr	An expression of the same type as symbol.
directive-statement	The Indirect command line to be processed if the condition is satisfied.

Examples

```
.SETS X "A"
.SETS Y "a"
.IF X LT Y .GOTO 200
```

The ASCII value of string symbol X is less than the ASCII value of string symbol Y, which satisfies the less-than condition. Thus, control passes to the line containing the label .200:.

```
.SETN N1 2
.SETN N2 7
.IF N1 <= N2 DIRECTORY/FULL
```

.IF - TEST IF SYMBOL SATISFIES SPECIFIED CONDITION

With the condition satisfied (numeric symbol N1 less than or equal to numeric symbol N2), the DIRECTORY command is processed.

```
.SETS S1 "AAb"
.SETS S2 "AA"
.SETS S3 "BBBB"
.IF S1 >= S2+S3[1:1] .INC N
```

The condition is satisfied where string symbol S1 is greater than or equal to the concatenation of string symbol S2 and the first character of string symbol S3 (AAb >= AAB). Therefore, Indirect increments numeric symbol N.

4.26 .IFACT/.IFNACT - TEST IF TASK IS ACTIVE OR DORMANT

The .IFACT or .IFNACT directive tests whether a task is active (.IFACT) or dormant (.IFNACT). If the test is true, the rest of the command is processed. If the specified task is not installed, Indirect assumes the dormant condition.

Formats

```
.IFACT  taskname directive-statement

.IFNACT  taskname directive-statement
```

where:

taskname	A 1- to 6- character legal task name.
directive-statement	The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFACT   REPORT  .GOTO 350

.IFNACT  REPORT  RUN REPORT
```

4.27 .IFDF/.IFNDF - TEST IF SYMBOL DEFINED OR NOT DEFINED

The .IFDF or .IFNDF directive tests whether a logical, numeric, or string symbol has been defined (.IFDF) or not defined (.IFNDF). If the test is true, the rest of the command line is processed. This directive does not test the value of the symbol.

Formats

```
.IFDF  ssssss  directive-statement

.IFNDF  ssssss  directive-statement
```

where:

ssssss	The 1- to 6-character symbol being tested. The symbol can be local, global, or an Indirect special symbol.
directive-statement	The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFDF  A  .GOTO 100

.IFNDF  A  .ASK A DO YOU WANT TO SET TIME
```

4.28 .IFINS/.IFNINS - TEST IF TASK INSTALLED OR NOT INSTALLED

The .IFINS or .IFNINS directive tests whether a task is installed (.IFINS) or not installed (.IFNINS) in the system. If the test is true, the rest of the command line is processed.

Formats

```
.IFINS  taskname  directive-statement

.IFNINS  taskname  directive-statement
```

where:

taskname	A 1- to 6-character task name.
directive-statement	The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFINS  ...PIP  .GOTO 250

.IFNINS  ...PIP  INS DW1:[ZZTKT]PIPRES.TSK
```

4.29 .IFENABLED/.IFDISABLED - TEST IF MODE ENABLED OR DISABLED

The .IFENABLED or .IFDISABLED directive tests whether an operating mode has been enabled with the .ENABLE directive or disabled with the .DISABLE directive. (See the description of the .ENABLE directive for the list of operating modes.)

Formats

```
.IFENABLED option directive-statement

.IFDISABLED option directive-statement
```

where:

option	The same operating mode option (with the exception of DATA) used with the .ENABLE or .DISABLE directive.
directive-statement	The Indirect command line to be processed if the condition is satisfied.

4.30 .IFLOA/.IFNLO - TEST IF DRIVER LOADED OR NOT LOADED

The .IFLOA or .IFNLOA directive tests whether a driver is loaded (.IFLOA) or not loaded (.IFNLOA) in the system. If the test is true, the rest of the command line is processed. For purposes of this directive, resident drivers are considered loaded.

Formats

```
.IFLOA dd:  directive-statement

.IFNLOA dd:  directive-statement
```

where:

dd:	A device driver.
directive-statement	The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFLOA XK:  .GOTO 250

.IFNLOA XK:  ;Sorry, XK Driver not available
```

4.31 .IFT/.IFF - TEST IF SYMBOL TRUE OR FALSE

The .IFT or .IFF directive tests whether a logical symbol is true or false. If the test is true, Indirect processes the rest of the command line.

Indirect exits with a fatal error if the tested symbol was previously defined as a numeric or string symbol.

Formats

```
.IFT  ssssss  directive-statement

.IFF  ssssss  directive-statement
```

where:

ssssss	The 1- to 6-character logical symbol being tested.
directive-statement	The Indirect command line to be processed if the condition is satisfied.

Examples

```
.IFT  A  .GOTO  100

.IFF  B  .GOTO  200
```

4.32 .INC - INCREMENT NUMERIC SYMBOL

The .INC directive increments a numeric symbol by one. Indirect exits with a fatal error if the symbol was previously defined as a logical or string symbol.

Format

```
.INC ssssss
```

where:

ssssss	The 1- to 6-character numeric symbol being incremented.

Example

```
.INC B
```

Increment by 1 the value assigned to the numeric symbol B.

4.33 .label: - DEFINE A LABEL

Labels always appear at the beginning of the line; they may be on a line with additional directives and/or a DCL command, on a line with a comment, or on a line by themselves. When control passes to a line with a label, the line is processed from the first character after the colon.

Commands do not have to be separated from the label by a space. Only one label is permitted per line. Labels are one through six characters in length and must be preceded by a period and terminated by a colon. A label may contain only alphanumeric characters and/or dollar signs ($).

It is also possible to define a label as a direct access label; once the label is found, its position in the command file is saved. This allows subsequent jumps to frequently-called labels or subroutines to be effected quickly. The first statement processed after a jump to a direct access label is the one on the next line. The maximum number of direct access labels you can define within an indirect command file is 16. If you define more than the maximum number allowed, the subsequent direct access labels replace the earliest, and so on.

To declare a label for direct access, leave the line following the colon blank.

Example

```
.100:    .ASK A DO YOU WANT TO CONTINUE?

         .IFT A .GOSUB 200
          .
          .
.200:
         .;THIS IS THE START OF A SUBROUTINE
          .
          .
          .
         .RETURN
```

4.34 .ONERR - BRANCH TO LABEL ON DETECTING AN ERROR

If Indirect detects one of the errors list below, control passes to the line containing the specified label. This feature provides you with a means of gaining control to terminate command file processing in an orderly manner.

- Undefined symbol
- Bad syntax
- Unrecognized command
- String substitution error
- Symbol type error (.IF, .IFT, .IFF, .INC, .DEC)
- Redefinition of a symbol to a different type (.ASK, .ASKN, .ASKS, .SETT, .SETF, .SETL, .SETN, .SETD, .SETO, .SETS)
- Data file error (.OPEN, .OPENA, .OPENR, .DATA, .CLOSE, or .READ between .ENABLE DATA and .DISABLE DATA)

Note that the .ONERR directive applies only to the error conditions listed; errors returned from a task external to Indirect (for example, a DCL syntax error) are not processed by the .ONERR directive.

Format

```
.ONERR  label
```

Upon detecting an error, the processor passes control to the line starting with .label:. The .ONERR directive must be issued before Indirect encounters the error condition. If the directive is executed (one of the listed errors is encountered), error processing passes to the specified label. If the label specified by the .ONERR directive does not exist and an error condition has occurred, command processing terminates.

Once an .ONERR condition has occurred, another .ONERR directive must be issued to trap a future error.

Example

```
.ONERR  100
```

Upon detecting one of the error conditions, Indirect passes control to the line labeled .100:.

4.35 .OPEN - OPEN SECONDARY FILE

The .OPEN directive opens a specified secondary file as an output file. The .DATA directive is used to place data in this secondary file.

Format (brackets not part of syntax)

```
.OPEN [#n] filename
```

where:

filename	A file to be opened as an output file. The default file type is .DAT.
#n	An optional file number in the range 0 to 3. The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

Note that you cannot include a comment that begins with a semicolon (;comment) in an .OPEN statement. Doing so results in a syntax error. (Comments that begin with an exclamation mark (!comment) are accepted.)

Example

```
.OPEN SECOUT
```

This directive opens the file SECOUT.DAT as an output file.

4.36 .OPENA - OPEN SECONDARY FILE FOR APPEND

The .OPENA directive opens a secondary file and appends all subsequent data to the file.

Format (brackets not part of syntax)

```
.OPENA [#n] filename
```

where:

filename	A secondary file to be opened with subsequent data appended to it. The default file type is .DAT.
#n	An optional file number in the range 0 to 3. The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

Note that you cannot include a comment that begins with a semicolon (;comment) in an .OPENA statement. Doing so results in a syntax error. (Comments that begin with an exclamation mark (!comment) are accepted.)

If the specified file does not already exist, .OPENA becomes the .OPEN directive by default.

Example

```
.OPENA SECOUT
```

This directive opens the file SECOUT.DAT as an output file and appends subsequent data to it.

4.37 .OPENR - OPEN FILE FOR READING

The .OPENR directive opens a file for reading with the .READ directive.

Format (brackets not part of syntax)

```
.OPENR [#n] filename
```

where:

filename — A file to be opened for reading. The default file type is .DAT.

#n — An optional file number in the range 0 to 3. The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.

You cannot include a comment that begins with a semicolon (;comment) in an .OPENR statement; it results in a syntax error. (Comments beginning with an exclamation mark (!comment) are accepted.)

Examples

```
.OPENR INDADD
```

This directive opens the file INDADD.DAT for reading with the .READ directive.

```
.OPENR DATLIB.ULB/LB:DATINP
```

This directive opens for reading the library module DATINP contained in the universal library DATLIB.

4.38 .PARSE - PARSE STRINGS INTO SUBSTRINGS

The .PARSE directive parses strings in a command line into substrings.

Format

```
.PARSE <string> <control-string> <var1> <var2> ...  <var9>
```

The string is broken up into substrings as specified by the control string. The substrings are stored in the specified variables. The first character of the control string delimits the first substring, the second character of the control string delimits the second substring, and so on. The last character of the control string is repeated if the number of variables exceeds the length of the control string. If you specify more variables than substrings, the additional variables are set to null strings. If you specify fewer variables than the number of substrings that can be parsed, the last variable contains the unparsed fragment of <string>.

The symbol <STRLEN> contains the actual number of substrings that Indirect processed (including explicit null substrings).

Example

```
.PARSE COMMAN " ," FILE A1 A2 A3 A4 A5
```

Given that COMMAN contains "TESTFILE IND,DCL,,LOA", this directive has the following results:

```
FILE = TESTFILE
A1   = IND
A2   = DCL
A3   = null
A4   = LOA
A5   = null

<STRLEN> contains a 5.
```

4.39 .PAUSE - PAUSE FOR OPERATOR ACTION

.PAUSE is provided for compatibility with RSX Indirect. It is a no-op under the PRO/Tool Kit.

4.40 .READ - READ NEXT RECORD

The .READ directive reads the next record into a specified string variable. The entire record is written into the variable. If the record is longer than 132 (decimal) characters, an error occurs.

After every .READ operation, the special symbol <FILERR> contains the FCS-11 file code for the read and the special symbol <EOF> reflects whether an end-of-file was found. (Note that .OPENR does not clear <EOF>.) If an error or end-of-file occurs, the string variable remains unchanged from its previous state.

Format (brackets not part of syntax)

```
.READ [#n] ssssss
```

where:

#n	An optional file number that specifies the file from which the record is to be read. The file number must be one of the numbers used in a previous .OPENR statement. The default is #0. You can substitute a numeric symbol for the value n by enclosing the symbol in apostrophes.
ssssss	The string variable into which the record will be read.

Example

```
                .ENABLE SUBSTITUTION

                .OPENR FILE
                IF <FILERR> NE 1 .GOTO ERROR
.LOOP:
                .READ RECORD
                .IFT <EOF> .GOTO DONE
                .IF <FILERR> NE 1 .GOTO ERROR
                ; 'RECORD'
                .GOTO LOOP
                  .
.ERROR:           .
                  .

.DONE:          .CLOSE
                  .
                  .
                  .
```

These directives open the file FILE.DAT for reading, read each record into the string variable RECORD, display each record on the terminal screen, and close the file.

4.41 .RETURN - RETURN FROM A SUBROUTINE

The .RETURN directive signifies the end of a subroutine and returns control to the line immediately following the .GOSUB directive that initiated the subroutine.

Format

```
.RETURN
```

4.42 .SETT/.SETF/.SETL - SET SYMBOL TO TRUE OR FALSE

The .SETT, .SETF, and .SETL directives define or change the value of a specified logical symbol. If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the logical symbol to the value specified. If the symbol has already been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the logical symbol was defined previously as a numeric or string symbol.

Formats

```
.SETT  ssssss

.SETF  ssssss

.SETL  ssssss  llllll
```

where:

ssssss	The 1- to 6-character logical symbol to be assigned a true or false value.
llllll	A logical or numeric expression. ssssss is assigned the value of llllll when the logical expression is evaluated.

Examples

```
.SETT  X
```

This directive sets the logical symbol X to true.

```
.SETF  ABCDE
```

This directive sets the logical symbol ABCDE to false.

```
.SETL  TEST SWITCHA!SWITCHB
```

This directive sets the logical symbol TEST to true if SWITCHA or SWITCHB is true.

4.43 .SETN - SET SYMBOL TO NUMERIC VALUE

The .SETN directive defines or changes the value of a specified numeric symbol. If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the symbol to the numeric value specified. If the symbol has already been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the numeric symbol was previously defined as a logical or string symbol.

Format

```
.SETN ssssss numexp
```

where:

ssssss	The 1- to 6-character numeric symbol.
numexp	A numeric expression.

When specifying a numeric value to assign to a symbol, you may combine a numeric symbol or constant with another numeric symbol or constant to form a numeric expression. If numeric expressions are used, no embedded blanks or tabs are permitted. Evaluation is done from left to right unless parentheses are used to form subexpressions that are evaluated first. The radix of an expression is octal if all the operands are octal and decimal mode has not been enabled; otherwise the radix is decimal.

Examples

```
.SETN NUMBER 27
```

This directive assigns to the numeric symbol NUMBER the value 27 (octal).

```
.SETN A1 3*(A2-5)
```

This directive assigns the numeric symbol A1 the value of symbol A2 minus 5 multiplied by 3.

4.44 .SETO/.SETD - SET SYMBOL TO OCTAL OR DECIMAL

The .SETO and .SETD directives redefine the radix of a specified numeric symbol (without affecting the symbol's actual value).

If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the symbol to the specified radix with a value of 0. If the symbol has already been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the symbol was previously defined as a logical or string symbol.

Formats

```
.SETO ssssss

.SETD ssssss
```

where:

ssssss	The 1- to 6-character numeric symbol to be assigned an octal or decimal radix.

Example

```
.SETN A 10      ; Sets symbol A to 10 (octal)
.SETD A         ; Defines A as a decimal radix symbol with a
                  value of 8 (decimal).
.SETO A         ; Defines A back to original radix with a
                  value of 10 (octal).
```

4.45 .SETS - SET SYMBOL TO STRING VALUE

The .SETS directive defines or changes the string value of a specified string symbol. If the symbol has not been defined, Indirect makes an entry in the symbol table and sets the symbol to the specified string value. If the symbol has been defined, Indirect resets the symbol accordingly. Indirect exits with a fatal error if the symbol was defined previously as a logical or numeric symbol.

Format

```
.SETS ssssss strexp
```

where:

ssssss The 1- to 6-character string symbol.

strexp Any string expression.

Indirect assigns to the specified symbol the string value represented by the string expression strexp. If a string constant is used in strexp, the constant must be enclosed by quotation marks ("constant").

You can combine a string symbol, constant, or substring with another string symbol or substring by the string concatenation operator (+) to form a string expression.

Examples

```
.SETS A "ABCDEF"
```

This directive assigns to the string symbol A the string value ABCDEF.

```
.SETS STR2 "ZZZ"
```

This directive assigns string symbol STR2 the value ZZZ.

```
.SETS X STR2+"ABC"
```

This directive assigns string symbol X the value of symbol STR2 plus ABC (that is ZZZABC).

```
.SETS X STR2+A[1:3]
```

This directive is equivalent to the previous directive; it assigns the string symbol X the string value of STR2 plus the first three characters of string A (that is ZZZABC).

The substring select expression is of the form:

"[start-index:ending-index]".

```
.SETS MYFILE <DIRECT>+"MYFILE.TXT"
```

This directive assigns the string symbol MYFILE the string value of the current directory and the string contained within the quotation marks. For example, if the current directory is [USERFILES], then MYFILE is assigned the string value [USERFILES]MYFILE.TXT.

4.46 .STOP - TERMINATE COMMAND FILE PROCESSING

The .STOP directive immediately terminates command file processing and exits. The message

```
$ @ <EOF>
```

is then displayed (unless .DISABLE DISPLAY is in effect).

The .STOP directive allows you to optionally set the exit status for Indirect execution.

Format (brackets not part of syntax)

```
.STOP [value]
```

where:

value	An optional numeric expression to serve as the exit status for Indirect. If you do not specify an exit status value, the .STOP directive is identical to the logical end-of-file directive (/).

Example

```
.STOP 0
```

This directive terminates command file processing and sets the exit status for Indirect to 0.

4.47 .TEST - TEST SYMBOL

The .TEST directive has two different functions. It tests a variable and sets various special symbols accordingly, and it does substring searches and sets the special symbol <STRLEN> accordingly.

Format 1

```
.TEST ssssss
```

where:

sssss The 1- to 6-character symbol to be tested.

The results of the test are as follows:

- If variable is a string, <SYMTYP> is set to 4 and <STRLEN> contains the length of the string. Also, the special symbols <ALPHAN>, <NUMBER>, <RAD50>, and <OCTAL> are set based on a scan of the characters of variable.
- If variable is numeric, <SYMTYP> is set to 2.
- If variable is octal, <SYMTYP> is set to 2 and <OCTAL> is set to TRUE.
- If variable is logical, <SYMTYP> is set to 0.

Format 2

```
.TEST string substring
```

where:

string A string symbol or constant.

substring A string expression.

In this case, the substring is searched for in the specified string. If the substring is present, <STRLEN> is set to the position of the starting character of the substring within the string. If substring is not present, <STRLEN> is set to 0.

Examples

1. If SUM is a string symbol, the directive statement:

```
.TEST SUM
```

   sets <SYMTYP> to 4 and places the number of characters represented by the symbol SUM into <STRLEN>.

2. The directive statements:

```
.SETS MAIN "ABCDEF"
.TEST MAIN "C"
```

   set <STRLEN> to 3, C's position in string ABCDEF.

4.48 .TESTDEVICE - TEST DEVICE

The .TESTDEVICE directive allows a command file to acquire information about any device in the system. The information, including error indications, is contained in the string symbol <EXSTRI>. Each device attribute in the string is separated by a comma (which allows processing by the .PARSE and .TEST directives). The first field of the string is the full physical name of the device. The next four fields are octal representations of the device-characteristics words (U.CW1 through U.CW4 of the Unit Control Block). Additional fields contain more information about the device.

Format

```
.TESTDEVICE dd[nn]:
```

where:

dd[nn]: The device about which the command file is requesting information.

The information stored in <EXSTRI> is in the following form:

```
ddnn:,xx,xx,xx,xx,atr,atr...,atr,
```

where:

ddnn: The physical device name for the device specified in the command line.

xx,xx, xx,xx The four device-characteristics words in octal notation.

atr One or more of the following device attributes:

- NSD "No such device" is configured into this system.
- LOD The device driver is loaded.
- UNL The device driver is not loaded.

Example

```
.TESTDEVICE SY:
```

This directive acquires information about user logical device SY: and stores it in <EXSTRI>.

4.49 .TESTFILE - TEST A FILE

The .TESTFILE directive determines if a specified file exists.

If you specify a file in the command line, the results of a .TESTFILE operation are contained in the symbols <FILSPC> and <FILERR>. <FILSPC> contains the fully qualified file specification and <FILERR> contains the FCS status code resulting from the search for the file.

Formats

```
.TESTFILE filespec
```

where:

filespec Is the file to be tested.

Examples

```
.TESTFILE IND.MAP
```

This directive assigns the following values if the file exists:

```
<FILERR> = 1
<FILSPC> = DW1:[USERFILES]IND.MAP;4
```

If the file does not exist, the directive assigns the following values:

```
<FILERR> = 230.
<FILSPC> = DW1:[USERFILES]IND.MAP;0
```

The following directive translates the logical name TI: into its physical device name.

```
.TESTFILE TI:
```

The directive assigns the symbol values as follows:

```
<FILERR> = 1
<FILSPC> = TT1:.DAT;0
```

4.50 .TESTPARTITION - TEST A PARTITION

The .TESTPARTITION directive allows a command file to obtain information about a partition in the system. The partition can be the one in which Indirect is running or any other partition. You can use the directive to verify that a partition is large enough before installing a task in it or that the partition is present before loading a special system. Indirect returns the information (in the special symbol <EXSTRI>) in the following format:

```
partition-name,base,size,type,
```

where base and size are in 64-byte blocks and type is SYS for system-controlled partitions, USR for user-controlled partitions, or NSP for an unknown partition name. If the partition is not found, Indirect returns a "No Such Partition" error in the form:

```
partition-name,,,NSP,
```

Format

```
.TESTPARTITION partition-name
```

where:

partition-name	A 1- to 6-character legal partition name.

Example

```
.TESTPARTITION GEN
;GEN,1500,2303,SYS,
```

This directive obtains information about the partition named GEN. The partition has a starting address of 150000 (octal), is 230300 (octal) bytes long, and is a system-controlled partition.

4.51 .TESTSYSTEM - TEST SYSTEM

The .TESTSYSTEM directive allows a command file to acquire information about the presence of certain operating system features.

Format

```
.TESTSYSTEM  keyword  number
```

where:

keyword — identifies the system component to be returned. Valid keywords are:

- OPTION which interprets the immediately following symbol (number) as a number indicating the system feature to be interrogated.
- SERIAL which returns the processor serial number in the <EXSTRI> return string value.

number — is a number representing the desired feature. (Symbolic equivalents for these numbers, called system feature symbols, are listed in the *P/OS System Reference Manual* under the description of the FEAT$ system directive.) For convenience, the command library INDSYS.CLB contains a procedure (INDSFN) that performs feature testing. The return value of string <EXSTRI> is the string "<TRUE>" or "<FALSE>", depending upon whether or not the current system contains or was built with the indicated feature.

Example

```
@DW1:[1,2]INDSYS/LB:INDSFN  HF$FPP ! is FPP present?
.SETL  FPUPRS  '<EXSTRI>'
```

Sets the logical symbol FPUPRS to true or false, depending on whether or not the floating point chip is installed on this system. HF$FPP is a system feature symbol.

```
.TESTSYSTEM OPTION -16.
.SETL  FPUPRS  '<EXSTRI>'
```

Does the same as the previous example, except that it uses the actual number representing the FPP (-16.) rather than the system feature symbol (HF$FPP).

4.52 .TRANSLATE - TRANSLATE A LOGICAL NAME

This directive attempts to translate a logical name as defined by the DCL command ASSIGN.

Format

```
.TRANSLATE  logical-name
```

If available, the translated value of 'logical-name' is returned in <EXSTRI>. The logical-name must have been previously defined (using the ASSIGN command, for example).

The .TRANSLATE command will translate a logical name by first searching in the USER logical name table. If the logical name is not found, the SESSION logical name table is searched. If the logical is still not found the SYSTEM table is searched.

NOTE

You cannot limit the search to a particular table as you can in DCL.

Example

The directive:

```
.TRANSLATE  LDW001:
;'<EXSTRI>'
```

displays the following:

```
;BIGDISK
```

4.53 .WAIT - WAIT FOR A TASK TO FINISH EXECUTION

.WAIT is provided for compatibility with command files moved from RSX-11 systems. It is a no-op under the PRO/Tool Kit.

4.54 .XQT - INITIATE PARALLEL TASK EXECUTION

.XQT is provided for compatibility with command files transferred from RSX-11 systems. The remainder of the command line is passed to DCL as though the ".XQT" had not been present.

4.55 COMPATIBILITY WITH COMMAND FILES FROM RSX SYSTEMS

Many features of the indirect command processors present on RSX-11M and RSX-11M-PLUS has been preserved under the PRO/Tool Kit. Because of differences in the goals of the P/OS system, some commands and symbols have little or no meaning. For example:

- The .WAIT and .XQT commands are present and parsed but behave as no-operation commands.

- The .XQT command is implemented as a synchronous operation, since it is not possible to initiate multiple commands or programs for parallel execution.

- The .PAUSE command does not pause. Command file processing continues without delay.

The following special symbols are available in the symbol table of the PRO/Tool Kit INDIRECT. When transporting procedures from other RSX-11 systems that make use of these symbols, examine such usage and make any appropriate changes.

<ALTMOD>	use <ESCAPE>
<BASLIN>	always <FALSE>
<CONFIG>	see module INDCFG in the procedure
<ERRCTL>	behave as on RSX-11 systems
<ERRNUM>	behave as on RSX-11 systems
<ERRSEV>	behave as on RSX-11 systems
<FILATR>	as on RSX-11 systems
<FMASK>	see module INDSFN in library LB000:[1,2]INDSYS.CLB.
<IAS>	always <FALSE>
<LIBUIC>	not meaningful
<LOCAL>	always <TRUE>
<MAPPED>	always <TRUE>
<NETUIC>	not meaningful
<NXTSYM>	use module INDDMP in LB000:[1,2]INDSYS.CLB
<RSX11D>	Always false in RSX-11M/M-PLUS and P/OS systems
<SYSUIC>	not meaningful
<TISPED>	always 0 on TT1:

4.56 INDIRECT MESSAGES

When Indirect encounters an error, it prints the appropriate error message and the command line in which the error occurred. If the line contained a substitution, the line as it appeared before the substitution took place is also displayed.

4.56.1 Information-Only Messages

@ <EOF>

Indirect has reached the end-of-file for the outermost command file and is terminating execution.

IND -- CONTINUING

Indirect is resuming execution after a .PAUSE or .DELAY directive.

IND -- DELAYING

A .DELAY directive was just executed, halting the processing of an indirect command file for a specified period of time.

IND -- INVALID ANSWER OR TERMINATOR

In response to a question from .ASK, you entered something other than Y, N, or null, followed by a RETURN; or you did not enter a numeric value in response to an .ASKN question; or you pressed the <ESCAPE> key either without escape recognition enabled or as a character other than the first one following the question. The question will be repeated.

IND -- VALUE NOT IN RANGE

The response to an .ASKN or .ASKS question was not within the specified range. Indirect repeats the question.

4.56.2 Error Messages

IND -- BAD RANGE OR DEFAULT SPECIFICATION

An illegal character was specified as a range or default argument. Only numeric expressions are permitted.

IND -- COMMAND FILE OPEN ERROR

The file being invoked in an @file or @file/LB:module command line cannot be found or opened.

IND -- DATA FILE ERROR, CODE x.

Indirect encountered an error while processing an .OPEN, .OPENA, .CLOSE, or .DATA directive or a data mode access to the secondary file.

IND -- FILE ALREADY OPEN

An .OPEN or .OPENA directive specified a file that was already open.

IND -- FILE NOT FOUND

An @filename or .CHAIN directive specified an incorrect file name or nonexistent file.

IND -- FILE NOT OPEN

Indirect encountered a .DATA or .CLOSE directive that did not reference an open file.

IND -- FILE READ ERROR

An error was detected in reading the indirect command file. This error is usually caused by records that are more than 132 (decimal) bytes long.

IND -- ILLEGAL FILE NUMBER

The file number in an .OPEN, .OPENA, .OPENR, .DATA, .ENABLE DATA, .READ, or .CLOSE directive is not in the range of 0 through 3.

IND -- ILLEGAL NESTING

Too many Begin-End blocks have been nested in the indirect command file. The maximum nesting depth is limited to the size of the symbol table.

IND -- INITIALIZATION ERROR, CODE x.

Indirect failed to complete initialization when you invoked it. The following list gives the meaning of the displayed code number:

1. Unable to acquire system information such as the UIC or device name.
2. Impure area setup failed.
3. Unable to acquire task-specific information.
4. Unable to acquire terminal-type information.
5. Unable to acquire the disk name and other information about the system device (SY:).
6. Unable to allocate enough space for command and data I/O buffers. The EXTEND TASK directive failed to return sufficient space for Indirect to allocate the buffers.
7. Initialization of allocated buffers failed.
8. Initialization of the DATA file structures failed.
9. Allocation of FCS-11 buffers for data and command lines failed.

10. Symbol table initialization failed.

11. Initialization cleanup failed.

12. Unable to obtain initial command line.

13. Unable to initialize the FMS-11 forms driver impure area.

>13. Error codes greater than 13 are returned by special purpose initialization modules.

Error number 6 is the only initialization error that you should encounter. If any other error from 1 through 12 persists, call your DIGITAL Customer Support Center.

IND -- INVALID KEYWORD

An unrecognized keyword (preceded by a period) was specified.

IND -- LABEL NOT AT BEGINNING OF LINE

The specified label does not start in the first column of the line. All labels must do so.

IND -- MAXIMUM INDIRECT FILE DEPTH EXCEEDED

An attempt was made to reference an indirect command file at a nested depth greater than the maximum specified in the build file for the Indirect task.

IND -- NO POOL SPACE

The executive dynamic memory allocation has been exhausted.

IND -- NULL CONTROL STRING

The control string specified with the .PARSE directive was null (there were no characters between the quotation marks).

IND -- NUMERIC UNDER- OR OVERFLOW

The evaluation of a numeric expression yielded a value outside the range 0 through 177777 (octal).

IND -- REDEFINING A READ-ONLY SYMBOL

An attempt was made to assign a new value to a read-only symbol. Read-only symbols cannot be overwritten.

IND -- REDEFINING SYMBOL TO DIFFERENT TYPE ssssss

An .ASK, .ASKN, .ASKS, .READ, .SETT, .SETF, .SETL, .SETN, or .SETS directive was used in an attempt to set the specified, already defined symbol to a different type. The first definition of a symbol determines its type (logical, numeric, or string); subsequent value assignments must conform to the original type.

IND -- .RETURN WITHOUT .GOSUB

A .RETURN directive was specified without a previous call to a subroutine (.GOSUB).

IND -- STRING EXPRESSION LARGER THAN 132. BYTES

An attempt was made to generate a string expression longer than 132 (decimal) characters.

IND -- STRING SUBSTITUTION ERROR

Indirect encountered an error during a substitution operation. A probable cause for the error is either the omission of a second apostrophe or the specification of a symbol that is not defined.

IND -- SUBROUTINE NESTING TOO DEEP

The maximum subroutine nesting level was exceeded. The maximum level is specified in the build file for the Indirect task.

IND -- SYMBOL TABLE OVERFLOW ssssss

The symbol table was full and there was no space for symbol ssssss.

IND -- SYMBOL TYPE ERROR ssssss

The symbol ssssss was used out of context for its type; for example, a numeric expression referenced a logical symbol. Only symbols of the same type can be compared.

IND -- SYNTAX ERROR

The format of the specified command line is incorrect.

IND -- UNDEFINED LABEL .label:

The label .label: specified in a .GOTO, .GOSUB, or .ONERR directive could not be found.

IND -- UNDEFINED SYMBOL ssssss

The symbol ssssss was referenced, but it had not been defined.

CHAPTER 5

FILE COMPARE UTILITY (CMP)

The File Compare Utility (CMP) compares the contents of two ASCII files on a line-by-line basis, determining whether parallel records are identical. The utility produces a listing of the differences between the two files.

Using CMP, you can perform the following file-compare functions:

- Generate a listing showing the differences between the two files. Each difference is listed as a pair: first, the lines from the first file, then the lines from the second file.

- Generate a listing in the form of one list, with differences marked by change bars.

- Generate output suitable for input to the Source Language Input Program utility (SLP). This output contains the SLP commands and input required to make the first input file identical to the second input file. (For more information on SLP, see Chapter 10.)

CMP provides switches that allow you to control compare processing. Using these switches, you can control the comparison of blanks, tabs, form feeds, and comments. You can also control line numbering and specify the number of lines required for CMP to consider that a match has been made between lines in the two files.

5.1 INVOKING CMP

You can invoke CMP in two ways:

1. Invoke the DIFFERENCES command, which in turn invokes CMP. See Chapter 3 for a description of the DIFFERENCES command.

2. Invoke CMP directly from the DCL command level.

To invoke CMP directly from the DCL command level, enter the following command:

```
$ RUN $CMP
```

You receive a new prompt to indicate that you are in the CMP environment:

```
CMP>
```

Once you are in the CMP environment, CMP waits for your command. The following section describes the command format.

5.2 CMP COMMAND FORMAT

The format for a CMP command is:

```
[outfile[/sw...]=] infile1,infile2
```

outfile

The file specification for the output file. This file can be in one of three formats, depending on the switch you specify in the command line. The defaults are:

SY0:	User's default system device
[curdir]	Current directory CMP is running under
FILCOM	Default file name
.LST	Default file type

However, if you do not specify an output file, the output defaults to your terminal screen. For example:

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

CMP lists the differences between FILE1.MAC and FILE2.MAC on your terminal screen. If you type the equal sign but give no output file specification, only the total number of differences is output to your terminal screen. For example:

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

/sw...

Switches that you apply to the output file specification. Some of the switches can be negated and some are mutually exclusive. See Section 5.3 for more information.

infile1

The file specification for the input file to be compared to infile2. The file name of this file must be specified. The default file type is .MAC.

infile2

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

```
CMP>DZ1:[FOO]EXEC,;2
```

CMP interprets the second input file as DZ1:[FOO]EXEC.MAC;2.

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

5.3 CMP SWITCHES

This section lists the CMP switches, describes the function of each one, and gives the default setting for each one. You specify switches after the output file in the command line.

/BL
/-BL — Specifies that blank lines in both files be included in compare processing. If this switch is specified in the form /-BL, blank lines are not included in compare processing. /-BL is the default switch.

/CB
/-CB

Specifies that CMP list infile2 with change bars, in the form of exclamation marks (!), to denote which lines do not have a corresponding line in infile1. When a section of lines in infile1 has been deleted in infile2 (the output listing file), the first line not deleted is marked. /-CB is the default switch.

You can change the change bar character from the exclamation mark to any character you wish by means of the /VB switch.

/CO
/-CO

Specifies that CMP include comments (that is, text preceded by a semicolon) in compare processing. /CO is the default switch.

/DI
/-DI

Specifies that CMP list the differences between the two files (rather than marking the lines in infile2). /DI is the default switch.

/CB and /DI are mutually exclusive switches. If you specify both, /CB overrides /DI.

/FF
/-FF

Specifies that CMP include records consisting of a single form-feed character in compare processing. /-FF is the default switch.

/LI:n

Specifies that a number (n) of lines must be identical before CMP recognizes a match. /LI:3 is the default switch.

When it encounters a match, CMP lists all the preceding nonmatching lines, along with the first line of the matched sequence of lines to help you find the location in the file where the match occurred.

/LN
/-LN

Specifies that lines in the output file be preceded by their line number. Line numbers are incremented by one for each record read, including blank lines. /LN is the default switch. If you specify /SL, /LN is unnecessary.

/MB
/-MB

Specifies that CMP include all blank and tab characters in a line in compare processing. If you specify /-MB, CMP interprets any sequence of blank and/or tab characters as a single blank character. However, all spaces and tabs are printed in the output listing. /MB is the default switch.

/SL[:au]

Directs CMP to generate an output file suitable for use as SLP command input. When you specify /SL, CMP generates the SLP command input necessary to make

infile1 identical to infile2. If a 1- to 8-character alphanumeric symbol is included after the /SL switch (:au), an audit trail is specified for SLP input. Section 5.4.3 gives an example of how CMP generates SLP command input. (For information on SLP, see Chapter 10.) /-SL is the default switch.

/TB
/-TB — Specifies that CMP include all trailing blanks on a line in compare processing. If you specify /-TB, CMP ignores all blanks following the last nonblank character on a line. When you specify /-CO and /-TB together, blanks that precede a semicolon (;) are considered trailing blanks and are ignored. /TB is the default switch.

/VB:nnn — Specifies an octal character code for the character you want to use as a change bar. You use this switch with the /CB switch. The value nnn specifies the octal character code. For example, you can specify /VB:174 for a vertical bar. /VB:041 (for an exclamation mark) is the default switch.

CMP default switch settings are listed in Table 5-1.

Table 5-1: Summary of CMP Default Switch Settings

Default	Description
/-BL	Do not compare blanks.
/-CB	Do not generate change bars.
/CO	Compare comments.
/DI	List only the differences between the two files.
/-FF	Do not compare form-feed characters.
/LI:3	Find three identical lines before a match can occur.
/LN	Generate numbered lines.
/MB	Compare all blank and tab characters.
/-SL	Do not generate an output file suitable for use as SLP command input.

Default	Description
/TB	Compare all trailing blanks.
/VB:041	Set the exclamation mark (ASCII 041) as the change bar character. Used with /CB.

5.4 FORMATS OF CMP OUTPUT FILES

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

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

This section gives an example of each of these formats. In the examples in the subsequent sections, the following files are used as infile1 (TEST1.DAT;1) and infile2 (TEST2.DAT;1):

DW1:[USERFILES]TEST1.DAT;1

```
LINE1
LINE2
LINE3
LINE4
LINE5
12345
23456
34567
LINE9
LINE10
LINE11
EXTRA
```

DW1:[USERFILES]TEST2.DAT;1

```
LINE1
LINE2
LINE3
LINE4
LINE5
45678
56789
67891
LINE9
LINE10
LINE11
EXTRA
EXTRA
EXTRA
EXTRA
```

5.4.1 Differences Format

If you enter a command line and do not specify any switches, CMP lists the differences between the two files on your terminal screen or in an output file. The differences are listed in pairs; first, the lines from infile1 that do not have counterparts in infile2 are listed, then the lines from infile2 that do not have counterparts in infile1 are listed. Each set of lines is terminated by the first line (or set of lines) for which a match is successful.

The following example shows the format of output generated without any switches. The output file is generated with the CMP command:

```
CMP>TESTDIF.DAT=TEST1.DAT,TEST2.DAT

*****************************************
   1)   DW1:[USERFILES]TEST1.DAT;1
     6    12345
     7    23456
     8    34567
     9    LINE9
*****************
   2)   DW1:[USERFILES]TEST2.DAT;1
     6    45678
     7    56789
     8    67891
     9    LINE9
*****************************************
   1)   DW1:[USERFILES]TEST1.DAT;1
*****************
   2)   DW1:[USERFILES]TEST2.DAT;1
    13   EXTRA
    14   EXTRA
    15   EXTRA

      2 differences found
```

The input files are TEST1.DAT and TEST2.DAT, which are shown in Section 5.4. There are two sets of differences separated by a long line of asterisks. (When there are several sets of differences, CMP separates each set from the next set by a long line of asterisks.) The short line of asterisks separates the pair of differences that comprise the set.

Note that because /LI:n was not specified, the number of lines required for a match defaults to 3. Thus, CMP found two differences.

5.4.2 Change Bar Format

You use the /CB switch to generate a listing containing change bars that show the differences between two files. In the CMP command line, infile2 is the listing you want generated.

The following example shows the format of output with change bars applied to lines from two files that do not match line for line. The output file is generated with the CMP command:

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

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

```
 1           LINE1
 2           LINE2
 3           LINE3
 4           LINE4
 5           LINE5
 6  !        45678
 7  !        56789
 8  !        67891
 9  !        LINE9
10           LINE10
11           LINE11
12           EXTRA
13  !        EXTRA
14  !        EXTRA
15  !        EXTRA

 2 differences found
```

5.4.3 SLP Command Input Format

You use the /SL[:au] switch to generate a file containing records to be used as SLP command input. /SL directs CMP to generate the SLP edit command lines and input lines required to make infile1 identical to infile2.

After executing CMP, you execute SLP (CMP does not generate an SLP command line). For a complete description of the SLP utility, see Chapter 10 in this manual.

The following example shows the format of output generated using the /SL switch. The output file is generated with the CMP command:

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

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

5.5 CMP MESSAGES

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

CMP -- n differences found

Explanation: CMP found n differences between the two files.

User Action: This is an informational message.

CMP -- Command syntax error

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

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

CMP -- Error reading input file

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

User Action: Reenter the command line.

CMP -- Error writing output file

Explanation: An I/O error occurred while CMP was writing the output file.

User Action: The output device may be full or bad. Check this, then reenter the command line.

CMP -- Illegal /LI value

Explanation: You specified a negative value for the number of lines required for a match.

User Action: Reenter the command line with a legal value specified.

CMP -- Illegal switch or switch value

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

User Action: Reenter the command line using a legal switch or switch value.

CMP -- Open failure on input file #1

Explanation: CMP could not open the first input file.

User Action: Check the file specification for the first input file and reenter the command line using the correct file specification.

CMP -- Open failure on input file #2

Explanation: CMP could not open the second input file.

User Action: Check the file specification for the second input file and reenter the command line using the correct file specification.

CMP -- Open failure on output file

Explanation: CMP could not open the specified output file.

User Action: Check the file specification for the output file and reenter the command line using the correct file specification.

CMP -- Too many differences for available core

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

User Action: You cannot compare the two files.

CHAPTER 6

FILE DUMP UTILITY (DMP)

The File Dump Utility (DMP) enables you to examine the contents of a specific file or volume of files. You can format the output in ASCII, octal, decimal, hexadecimal, or Radix-50 form and dump it to any suitable output device, such as a printer, terminal screen, or disk.

You can dump the header and/or virtual blocks of a file or only the virtual records of a file. If you are dumping a volume, you can specify a range of logical blocks. DMP handles blocks of up to 256 (decimal) words in length. The maximum block size must not exceed this length.

DMP operates in two basic modes: file mode and device mode. Use file mode to dump virtual records or virtual blocks; use device mode to dump logical blocks.

File Mode

In file mode, one input file is specified, and all or a specified range of virtual blocks are dumped. You can also dump all the virtual records of a specified file in this mode. The input device must be a FILES-11 formatted disk.

In file mode, you can specify that data be dumped one record or one block at a time. A virtual block or record refers to one block or record of data in a file. Virtual blocks and records are numbered sequentially from 1 through n, where n is the total number of blocks or records in the file. Virtual block 0 contains the header of the file. Use the /BL:n:m switch to dump virtual blocks and the /RC switch to dump virtual records. The /BL and /RC switches are mutually exclusive. (DMP switches are listed in Section 6.3.)

Device Mode

In device mode, you specify only the input device, and a specified range of logical blocks is dumped. The /BL:n:m switch is a required parameter in this mode.

A logical block refers to a physical 512-byte block on disk. Logical blocks are numbered from 0 to n-1, where n is the total number of logical blocks on the device.

NOTE

The volume must be mounted foreign.

6.1 INVOKING DMP

You can invoke DMP in two ways:

1. Invoke the DUMP command, which in turn invokes DMP. See Chapter 3 for a description of the DUMP command.

2. Invoke DMP directly from the DCL command level.

To invoke DMP directly from the DCL command level, enter the following command:

```
$ RUN $DMP
```

You receive a new prompt to indicate that you are in the DMP environment:

```
DMP>
```

Once you are in the DMP environment, DMP waits for your command. The following section describes the command format.

6.2 DMP COMMAND FORMAT

The format for a DMP command is:

```
[outfile][/sw][/sw...]=inspec[/sw][/sw...]
```

outfile

Specifies the output file. If the output file name and file type are unspecified, DMP creates the file DMPFIL.DMP. TI: (terminal) is also an acceptable outfile specification.

/sw

Specifies one of the switches listed in Section 6.3. Unless otherwise indicated in a switch description, all switches can be applied either to the input file or to the output file with equal effect. DMP will allow multiple dumps in a single command line. Therefore, any or all of the current format switches may be specified. Certain switches are mutually exclusive. For example, the /HX, /LW, and /WD switches are mutually exclusive hexadecimal dump switches. The first one in the following order will be the only one executed: /LW, /WD, /HX.

inspec

Specifies the input device and file or input device only. In file mode, the equal sign and the input file name and file type are required because DMP does not provide a default for either of them. However, the input file version number defaults to the latest version and the device defaults to SY: and the current directory.

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

For a complete description of file specifications, see the *Tool Kit User's Guide*.

6.3 DMP SWITCHES

DMP switch specifications consist of a slash (/) followed by a switch name, optionally followed by a value. The value is separated from the switch by a colon (:). DMP functions are implemented by the following switches:

Default	The default is a word mode octal dump.

/AS

Specifies that the data should be dumped one byte at a time in ASCII mode. The control characters (0-37 octal) are printed as eight-bit characters consisting of a circumflex (^), followed by the alphabetic character corresponding to the character code plus 100 octal. For example, bell (code 7 octal) is printed as ^G (code 107 octal). Lowercase characters (140-177 octal) are printed as a percent sign (%), followed by the corresponding uppercase character (character code minus 40 octal), unless the /LC switch is specified. DMP will also display the DEC Multinational Characters.

NOTE

The /AS and /OCT switches are mutually exclusive when dumping bytes.

/BA:n:m

Specifies a 2-word base block address (the initial base address is 0,0), where n is the high-order base block address (octal), and m is the low-order base block address (octal). The address may also be specified in decimal by using a period after the number. All future block numbers specified by the /BL switch will be added to this value to obtain an effective block number. This switch is useful for specifying block numbers that exceed 16 bits. For example:

```
DMP>/BA:1:0
```

specifies that all future block numbers will be relative to 65536 decimal (200000 octal).

```
DMP>/BA:0:0
```

clears the base address. Once the /BA switch is specified, it remains in effect until it is used again to set a new base address.

When the /BA switch appears in a command line, no blocks are dumped. The only result of the command line is to set the base address.

/BL:n:m

Specifies the range of blocks to be dumped, where n is the first block and m is the last block. The values of n and m must not exceed 16 bits. In file mode only, the /BL switch is not required. If the /BL switch is not specified, DMP will dump all blocks of the specified file, relative to the current base address.

If /BL:n:m is specified in file mode, it specifies the range of virtual blocks to be dumped. If /BL:n:m is specified as /BL:0 in file mode, no virtual blocks are dumped. This is useful for dumping only the header portion of the file (see /HD). The /BL switch and the /RC switch are mutually exclusive.

The /BL:n:m switch is a required parameter in device mode. When used in device mode, it specifies the range of logical blocks to be dumped.

The value n represents the block number of the first block dumped. Successive blocks are labeled with a block number one higher than the preceding block number. The dump will continue until the block labeled m is dumped.

/BY

Specifies that the data be dumped in octal byte format.

/DC

Specifies that the data be dumped in decimal word format.

/FI:file-number:
sequence-number

In file mode, the file number can be used instead of a file name as a file specification for input.

/HD[:F or :U]

This switch is an optional parameter used in file mode. If specified, the /HD switch causes the file header as well as the specified or implied portion of the file to be dumped. Example:

```
DMP>TI:=JMF.DAT/HD/BL:5:6
```

This example dumps the header of JMF.DAT in header format and virtual blocks 5 and 6 in octal format.

In addition, this switch has two options. "F", the default, causes a FILES-11 formatted dump of the header. "U" specifies an unformatted octal dump. An octal dump also occurs when DMP is used on non-FILES-11 headers.

If you want only the header portion of the file to be dumped, specify:

```
/HD/BL:0
```

/HF

Specifies the format for data blocks that have the FILES-11 header structure. Other blocks are output as a data dump in the format selected by /AS, /BY, and so on, in default octal words.

Example:

```
DMP>HEAD.LST=[0,0]INDEXF.SYS/HF
```

This example generates a dump of the index file INDEXF.SYS and formats all the headers in the file.

/HX

Specifies that the data be dumped in hexadecimal byte format. Note that a hexadecimal dump reads from right to left. (See also the /LW and /WD switches.)

/ID

Causes DMP's version to be identified. This switch may be specified on a command line by itself at any time.

Example:

```
DMP>/ID
DMP--DMP VERSION M07.1C
```

/LB

Requests logical block information for a file. The starting block number and a contiguous or noncontiguous indication for the file are displayed.

Example:

```
DMP>TI:=RICKSFILE.DAT;3/LB
STARTING BLOCK NUMBER = 0,135163 C
```

The file RICKSFILE.DAT, version 3, is a contiguous file starting at block number 0,135163. (See /BA:n:m for block number description.)

/LC — Specifies that the data should be dumped in lowercase characters.

/LIM:n:m — Specifies the range of bytes, n through m, of each record or block to be dumped. The default remains /OCT if you do not specify format switches.

/LW — Specifies that the data be dumped in hexadecimal double-word format.

/MD[:n] — Specifies a memory dump and allows control of line numbers. Line numbers are normally reset to zero whenever a block boundary is crossed. The /MD switch allows lines to be numbered sequentially for the full extent of the file, that is, the line numbers are not reset when block boundaries are crossed. The optional value (:n) specifies the value of the first line number. The default is 0. The /MD switch is used with the output file specification.

/OCT — Specifies that the data should be dumped in octal format in addition to other formats specified. If no DMP format switches are specified, the default is octal. The /AS switch and the /OCT switch are mutually exclusive when dumping bytes.

/R5 — Specifies that data be dumped in Radix-50-format words.

/RC — Specifies that data be dumped a record at a time (rather than a block at a time). The data format is determined by setting any of these format switches: /AS, /DC, /HX, /LW, /R5, or /WD.

The largest record that DMP can process is limited by the amount of space available to the DMP task. DMP's task image has 512 (decimal) bytes allocated to it initially.

The /RC switch and the /BL switch are mutually exclusive.

/WD — Specifies that the data be dumped in hexadecimal word format.

6.4 DMP EXAMPLES

Three examples of dump listings are included in this section to illustrate how the various DMP switches can be used. DMP edits blocks or records 16 (decimal) bytes at a time. The dump includes the indicated number of valid bytes in the block or record. The remaining number of bytes are listed as null bytes (0).

6.4.1 A Multiple Format Dump

The following command line shown dumps virtual blocks 5 and 6 of DSC.MAC in hexadecimal, Radix-50, and decimal format. Each line of the output file will appear in three different formats.

```
DMP>DOC.DMP=[USERFILES]DSC.MAC/HX/R5/DC/BL:5:6
```

The contents of DOC.DMP are:

```
DUMP OF DW1:[USERFILES]DSC.MAC;1 - FILE ID 17725,11,0
                VIRTUAL BLOCK 0,000005 - SIZE 512. BYTES


4E 41 4D 4D 4F 43 20 41 20 3B 00 1E 53 45 52 49          0000    ;HX
000000    MFY ML7   0 EFK EFQ L$K LN/ LT3                       ;R5
0.       21065. 21317. 00030. 08251. 08257. 20291. 19789. 20033.;DC

53 45 53 53 45 43 4F 52 50 20 44 4E 41 20 2C 44          0010
000020    GCL JP2 J7F L22 L$Z KCK MMK ML7
16.      11332. 16672. 17486. 20512. 20306. 17731. 21331. 21317.

53 52 49 46 20 3B 00 39 00 3B 00 01 2E 54 49 20          0020
```

```
DUMP OF DW1:[USERFILES]DSC.MAC;1 - FILE ID 17725,11,0
               VIRTUAL BLOCK 0,000006 - SIZE 512. BYTES


20 44 4E 46 55 42 24 20 51 45 20 30 52 20 46 49         0000
000000    KI3 MEX EF  M E E1H MYZ LT8 EFT
0.       17993. 21024. 08240. 20805. 09248. 21826. 20038. 08260.

44 4E 45 09 00 09 50 4F 4F 4C 20 45 56 41 45 4C         0010
000020    KCT M2A EFU L$T L39   I KA3 J7F
16.      17740. 22081. 08261. 20300. 20559. 00009. 17673. 17486.

54 53 24 20 54 45 4C 09 00 2B 00 50 4F 4F 4C 20         0020
     .
     .
```

6.4.2 A Record Dump

The following command line dumps all of the virtual records of YACHT.SEQ in ASCII and decimal word format:

```
DMP>REC.DMP=[USERFILES]YACHT.SEQ/RC/AS/DC
```

The contents of REC.DMP are:

```
DUMP OF DW1:[USERFILES]YACHT.SEQ;1 - FILE ID 15451,35,0
                 RECORD NUMBER 01. - SIZE 41. BYTES


000000    A  L  B  E  R  G              3  7     M  K
0.       19521. 17730. 18258. 08224. 08224. 14131. 19744. 08267.

000020    I  I        K  E  T  C  H     3  7     2  0  0
16.      18761. 08224. 17739. 17236. 08264. 14131. 12832. 12336.

000040    0  0  1  2  3  6  9  5  1 ^@ ^@ ^@ ^@ ^@ ^@ ^@
32.      12336. 12849. 13875. 13625. 00049. 00000. 00000. 00000.


                 RECORD NUMBER 02. - SIZE 41. BYTES


000000    A  L  B  I  N                  7  9
0.       19521. 18754. 08270. 08224. 08224. 14647. 08224. 08224.
```

```
000020                  S  L  O  O  P     2  6     0  4  2
16.       08224. 08224. 19539. 20303. 08272. 13874. 12320. 12852.

000040    0  0  1  0  1  7  9  0  0 ^@ ^@ ^@ ^@ ^@ ^@ ^@
32.       12336. 12337. 14129. 12345. 00048. 00000. 00000. 00000.
     .
     .
     .
```

6.4.3 A Header Dump

The following command line dumps only the header of DSC.MAC.

```
     DMP>DHR.DMP=[USERFILES]DSC.MAC/HD/BL:0
```

```
The contents of DHR.DMP are:
DUMP OF DW1:[USERFILES]DSC.MAC;1 - FILE ID 17725,11,0
                                        FILE HEADER
HEADER AREA
        H.IDOF          027
        H.MPOF          056
        H.FNUM,
        H.FSEQ          (17725,11)
        H.FLEV          401
        H.FOWN          [200,200]
        H.FPRO          [RWED,RWED,RWED,RWED]
        H.UCHA          000=
        H.SCHA          000 =
        H.UFAT
                F.RTYP  002 = R.VAR
                F.RATT  002 =   FD.CR
                F.RSIZ  116 = 78.
                F.HIBK  H:0 L:000040 = 32.
                F.EFBK  H:0 L:000040 = 32.
                F.FFBY  532 = 346.
                (REST)
                000000 000000 000000 000000 000000 000000 000000 000000
                000000
IDENTIFICATION AREA
        I.FNAM,
        I.FTYP,
        I.FVER          DSC      .MAC;1
        I.RVNO          1
        I.RVDT          13-OCT-80
        I.RVTI          09:52:46
        I.CRDT          13-OCT-80
        I.CRTI          09:52:45
        I.EXDT          --
```

```
MAP AREA
        M.ESQN          000
        M.ERVN          000
        M.EFNU,
        M.EFSQ          (0,0)
        M.CTSZ          001
        M.LBSZ          003
        M.USE           014 = 12.
        M.MAX           314 = 204.
        M.RTRV
        SIZE    LBN
        12.     H:000 L:036215 = 15501.
        3.      H:000 L:036235 = 15517.
        1.      H:000 L:036250 = 15528.
        2.      H:000 L:036272 = 15546.
        3.      H:000 L:036313 = 15563.
        11.     H:000 L:036411 = 15625.
CHECKSUM
        H.CKSM          122620
```

6.5 DMP ERROR MESSAGES

DMP -- BAD DEVICE NAME

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

User Action: Reenter the command line specifying the correct device.

DMP -- BLOCK SWITCH REQUIRED IN LOGICAL BLOCK MODE

Explanation: /BL must be specified.

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

DMP -- CANNOT FIND INPUT FILE

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

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

DMP -- COMMAND SYNTAX ERROR

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

User Action: Reenter the command line specifying the correct syntax.

DMP -- FAILED TO ASSIGN LUN

Explanation: An illegal device was entered in a file specification.

User Action: Reenter the command line specifying the correct device.

DMP -- FAILED TO READ ATTRIBUTES

Explanation: A file was specified for which you did not have read access privileges.

User Action: Rerun DMP after you have changed the protection of the file to READ access.

DMP -- ILLEGAL SWITCH

Explanation: A switch was specified that is not a valid DMP switch, or a legal switch was used in an invalid manner.

User Action: Reenter the command line specifying the correct switch.

DMP -- ILLEGAL USE OF /RC SWITCH

Explanation: The /RC switch can be used only in file mode (see the beginning of this chapter).

User Action: Reenter the command line specifying a file name.

DMP -- ILLEGAL VALUE ON HD SWITCH

Explanation: An option was entered other than F or U for the /HD switch.

User Action: Reenter the command line specifying the correct option.

DMP -- I/O ERROR ON INPUT FILE

or

DMP -- I/O ERROR ON OUTPUT FILE

Explanation: One of the following conditions exists:

- A problem exists on the physical device.
- The file is corrupted or the format is incorrect.
- The output volume is full.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

DMP -- NO INPUT FILE SPECIFIED

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

User Action: Reenter the command line specifying an input file.

DMP -- NO LISTS OR WILD CARDS ALLOWED

Explanation: Either a command line with more than one input or output file name was entered, or a wildcard was entered as a file specification.

User Action: Reenter the command line, specifying only one input file specification and one output file specification. No wildcard specifications are allowed.

DMP -- OPEN FAILURE ON INDIRECT FILE

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

- The file is protected against access.
- A problem exists on the physical device.
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

DMP -- OPEN FAILURE ON INPUT FILE

or

DMP -- OPEN FAILURE ON OUTPUT FILE

Explanation: One of the following conditions exists:

- The file is protected against access.
- A problem exists on the physical device.
- The named file does not exist in the specified directory.
- The volume is not mounted.
- The specified file directory does not exist.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

CHAPTER 7

LIBRARIAN UTILITY PROGRAM (LBR)

The Librarian Utility Program (LBR) allows you to create, update, modify, list, and maintain library files. A library file is a direct access file that contains a collection of related files. LBR organizes files, usually having the same file type, into library modules so that you have rapid and convenient access to your files.

Library files contain two directory tables: the Entry Point Table (EPT) and the Module Name Table (MNT). The EPT contains entry point names that consist of global symbols defined as entry points in MACRO source programs. The MNT contains names of the modules in the library. Both tables are alphabetically ordered.

There are three kinds of library files: object, macro, and universal, described below.

Object library files (.OLB) contain object files (.OBJ). The module names are derived from .TITLE directives, while the entry point names are derived from global symbols defined in the module. LBR references the module code in the library by the module name. The source program references object library modules by the entry point name. Entry points apply only to object libraries.

You use object module libraries as input to the Professional Application Builder (PAB). PAB searches for definitions of all global symbols referenced in a program in the following manner. First, PAB searches the other modules specified, then it searches the specified user-written object module library, and finally, it searches the system library.

Macro library files (.MLB)

contain source macro files (.MAC). The module names are derived from .MACRO directives. From each macro definition, LBR extracts the name and creates an entry in the module name table. The entry in the module name table is the means by which the assembler finds the associated macro definition in the library.

You use macro library modules as input to the Professional MACRO-11 Assembler. The assembler searches the specified library for macros listed in .MCALL statements and called in the source program before searching the system macro library.

Universal library files (.ULB)

contain modules inserted from any kind of file, whether it be a program or text. The module names are either user-specified in the Insert (/IN) switch, or derived from the file name at the time of insertion.

Primarily, you use universal libraries to package related files together. You can reference a universal library module in a program by using the Universal Library Access ($ULA) system library routine. $ULA, specified in the macro source program, establishes the necessary conditions for access (read-only) to a universal library module.

Section 7.3 describes how you invoke LBR.

7.1 FORMAT OF LIBRARY FILES

A library file consists of a library header, an entry point table, a module name table, the library modules and their headers, and any available space.

The entry point table has zero length for macro and universal libraries. Figure 7-1 illustrates object and macro library file format. Figure 7-2 illustrates universal library file format.

7.1.1 Library Header

The header section is a full block in which the first 24 (decimal) words are used to describe the current status of the library. The header's contents are updated as the library is

modified. This allows LBR to access the necessary information to perform its functions (for example, Insert, Compress, and Delete). The twenty-fourth word in the library header is the default insert file type for universal libraries and is undefined for macro and object libraries. See Figure 7-3.

7.1.2 Entry Point Table

The entry point table consists of 4-word elements containing an entry point name (words 0-1) and a pointer to the module header of the module where the entry point is defined (words 2-3). See Figure 7-4. This table is searched when a library module is referenced by one of its entry points. The table is sequenced in order of ascending entry point names. The entry point table applies only to object library files.

7.1.2.1 Module Name Table - The module name table is searched when the library module is referenced by its module name rather than by one of its entry points. It is made up of 4-word elements: a module name (words 0-1) and a pointer to the module header (words 2-3). See Figure 7-5. The module name table is sequenced in order of ascending module names.

7.1.2.2 Module Header - Each module starts with a header of eight words for object and macro modules and 32 (decimal) words for universal modules. The module header contains information about the module such as the type and status of the module, its length (number of words), and its attributes. See Figures 7-6 and 7-7.

In object and universal modules, the low-order bit of the attributes byte is set if the module has the selective search attribute. In universal modules, bit 1 of the attributes byte is set if the input file was contiguous. Also, in object modules, the two words of type-dependent information contain the module identification defined by the .IDENT directive at assembly time. In macro modules, these two words are undefined.

For universal modules, type-dependent identification is derived from the file type and version number of the input file.

Universal libraries allow you to change the module header, which contains optional descriptive information, by means of the Modify Header switch (/MH).

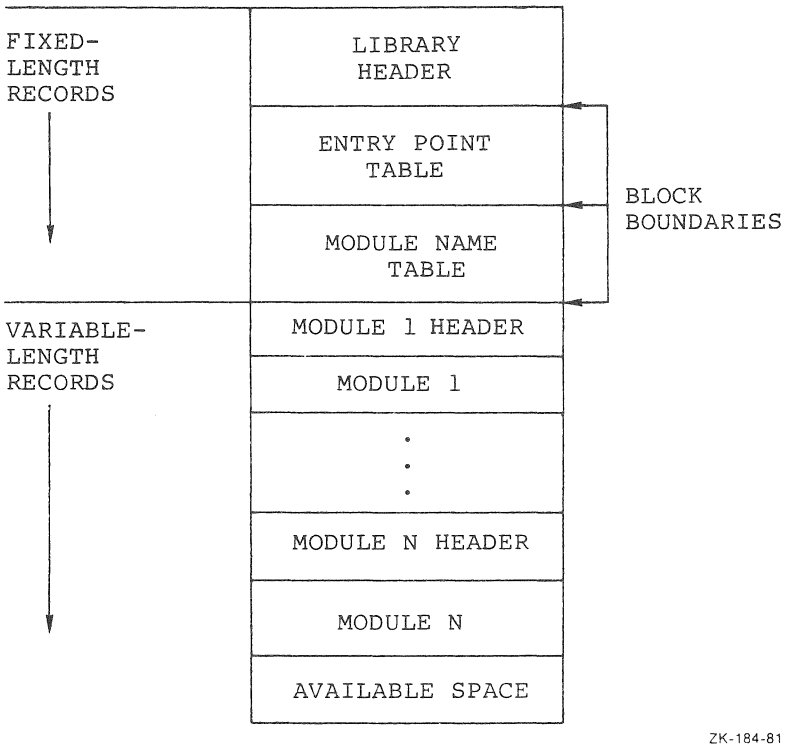
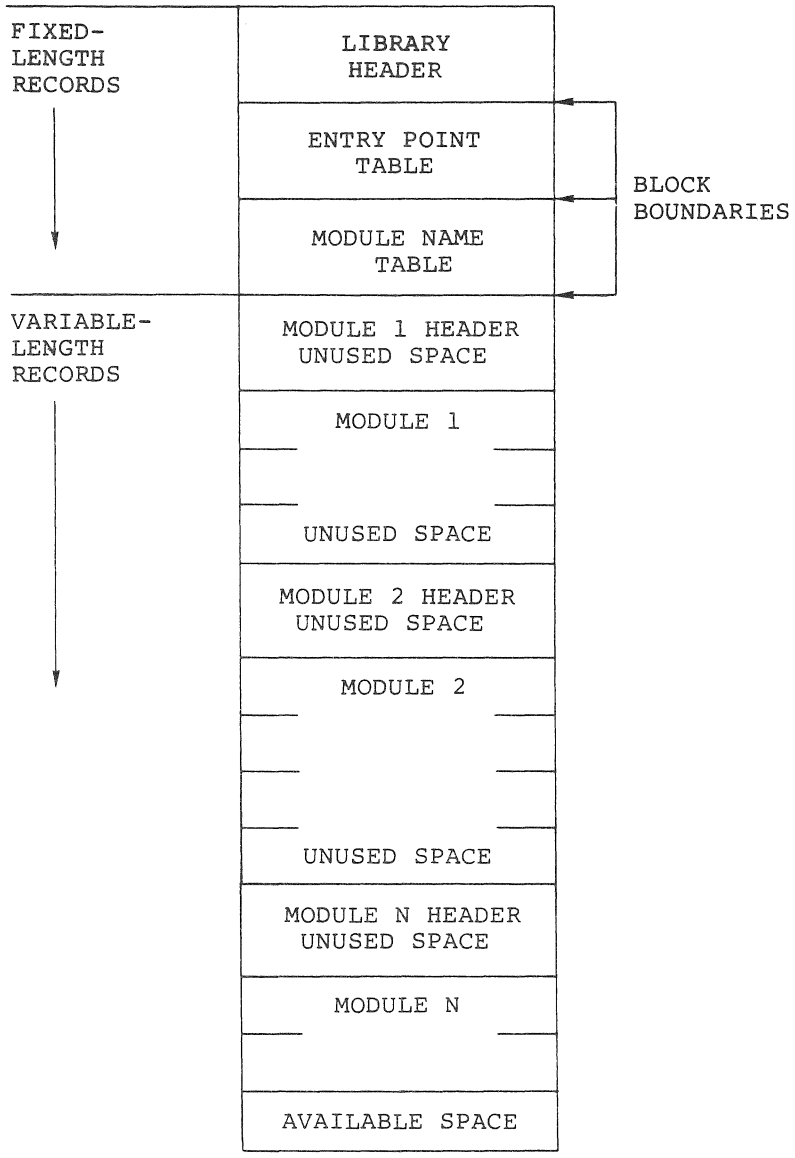


Figure 7-1: General Format for Object and Macro Library Files



NOTE

All universal module headers and the first record of each universal module will start on a block boundary.

Figure 7-2: Universal Library File Format

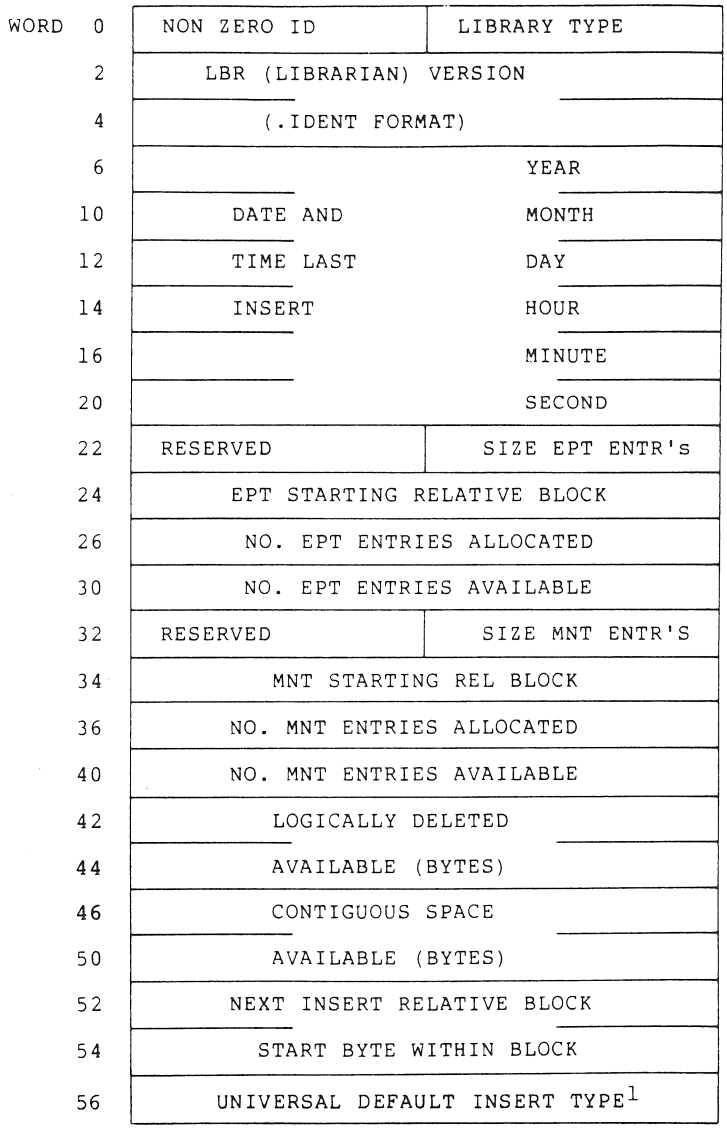


Figure 7-3: Contents of Library Header

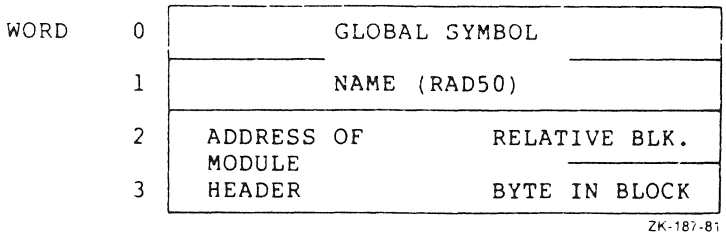


Figure 7-4: Format of Entry Point Table Element

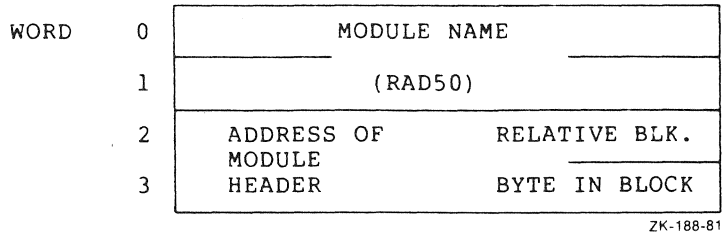


Figure 7-5: Format of Module Name Table Element

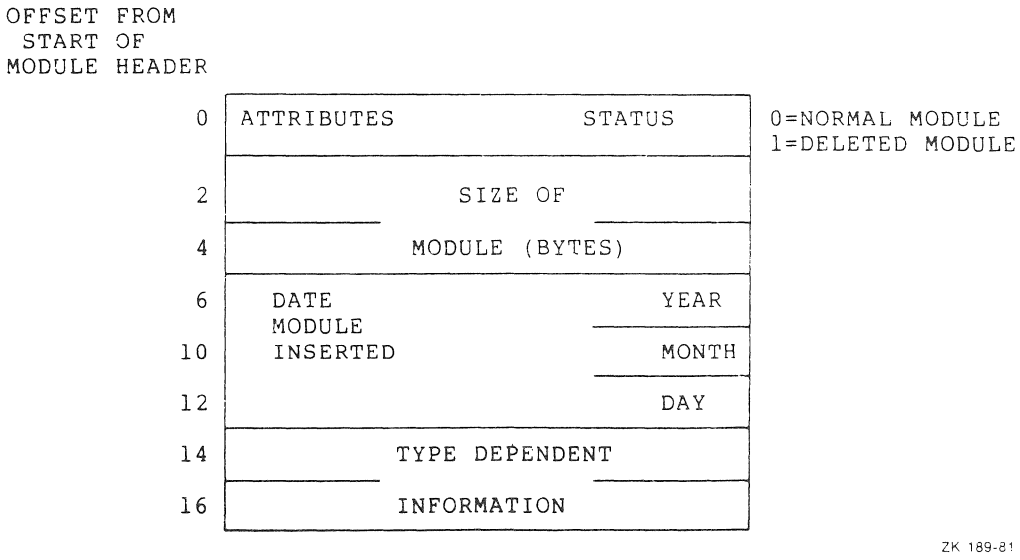


Figure 7-6: Module Header Format for Object and Macro Libraries

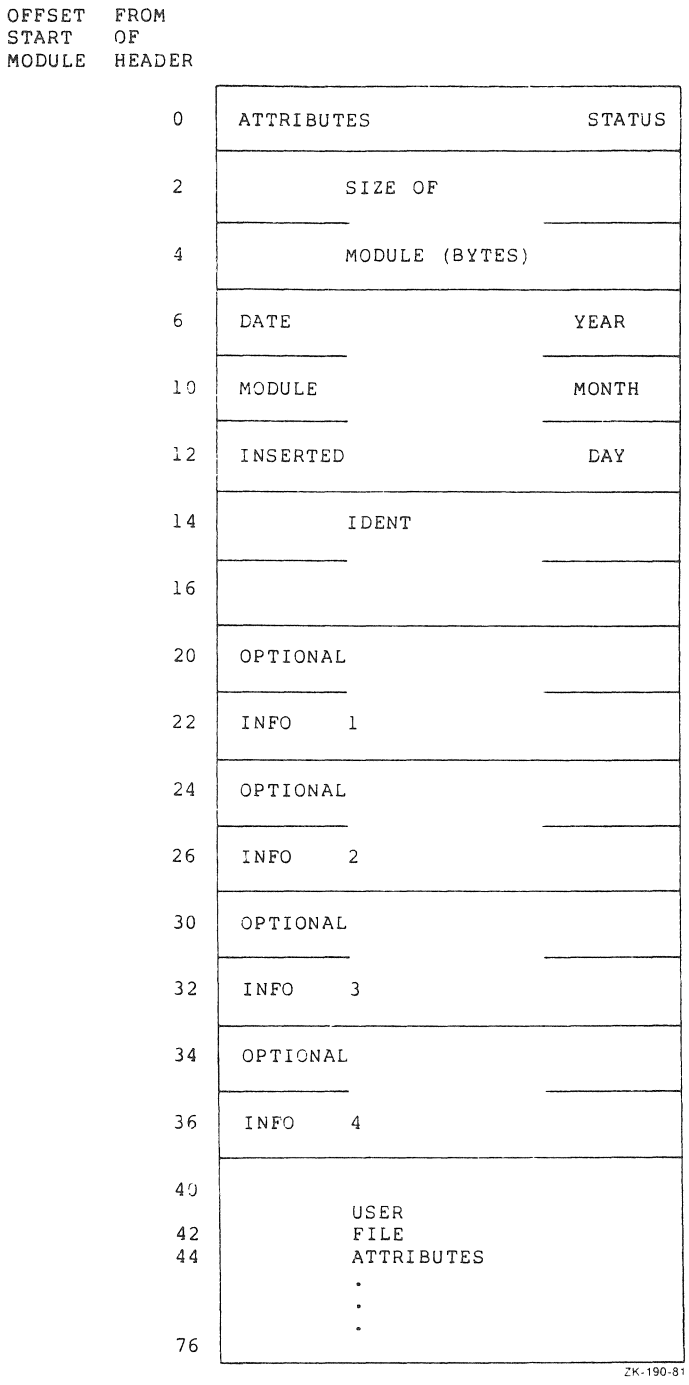


Figure 7-7: Module Header Format for Universal Libraries

Note to Figure 7-7

Locations 40 (octal) through 76 (octal) contain user file attributes. User file attributes comprise the first of five sections in the FDB of the original file from which the module was created. The user file attributes are:

- Record type
- Record attribute
- Record size
- Highest virtual block
- End of file block number
- Optional information

When you create a file and insert it into a universal library, LBR copies the input file attributes to the module header. You can modify some of these attributes by using the modify header switch (/MH).

7.2 LBR RESTRICTIONS

The following restrictions apply when using LBR:

- Limit of 65,536 (64K) words per module.
- Limit of 65,536 (64K) blocks per library.
- Tables should be allocated their anticipated maximum size. Expanding table allocations requires using the Compress switch (/CO) to copy the entire file.
- A fatal error results if an attempt is made to insert a module into a library that contains a module with a different name from, but with the same entry point as, the inserted module. For further information, refer to the discussion of the /IN switch in Section 7.5.8.
- The use of wildcards in file specifiers is not allowed (that is, forms such as *.OBJ, where the * indicates all modules with type .OBJ).

The library's tables must contain enough space for both the modules being replaced and their replacements, because the new modules are entered and the old modules are only logically (not physically) deleted.

7.3 INVOKING LBR

You can invoke LBR in two ways:

1. Invoke one of the LIBRARY commands, which in turn invokes LBR. See Chapter 3 for a description of the LIBRARY commands.

2. Invoke LBR directly from the DCL command level.

To invoke LBR directly from the DCL command level, enter the following command:

```
$ RUN $LBR
```

You receive a new prompt to indicate that you are in the LBR environment:

```
LBR>
```

Once you are in the LBR environment, LBR waits for your command. LBR accepts command lines in the following general format:

```
outfile[,listfile]=infile1[,infile2,...infilen]
```

LBR allows only one level of indirect command file nesting.

7.4 DEFAULTS FOR LBR FILE SPECIFIERS

Table 7-1 describes the defaults for LBR file specifiers.

Table 7-1: LBR File Specifiers Defaults

Specifier	Default
dev:	Output File SY0: Listing File The device that was specified for the output file; otherwise, the default for the output file. Input File For the first input file specifier, SY0:. For subsequent input file specifiers, the device specified in the previous input file specifier; otherwise, the default for the previous input file specifier.
[dir]	Output File The default directory under which LBR is currently running. Listing File The directory that was specified for the output file; otherwise, the default for the output file specifier. Input File For the first input file specifier, the directory under which LBR is currently running. For subsequent input file specifiers, the directory specified in the previous input file specifier; otherwise, the default for the previous input file specifier.
filename	No default. Must be specified.

DEFAULTS FOR LBR FILE SPECIFIERS

Specifier	Default
.type	Output File Depends on the default in effect (see Section 7.5.4), except when the Compress (/CO) or Create (/CR) switch is specified (see Sections 7.5.1 and 7.5.2, respectively). Listing File .LST Input File Refer to the descriptions of Compress (Section 7.5.1), Insert (Section 7.5.8), and Replace (Section 7.5.12) switches.
;ver	Latest version of the file, or latest version plus 1 for the output file when the Compress (/CO), Create (/CR), or Extract (/EX) switches are specified.
/switch	Output File /IN (Insert) List File /LI (List module names) Input File None

7.5 LBR SWITCHES

LBR uses switches appended to file specifications to invoke functions. These switches are summarized in Table 7-2.

Table 7-2: LBR Switches

Option	Switch	Function
Compress	/CO	Compress a library file.
Create	/CR	Create a library file.
Delete	/DE	Delete a library module and all its entry points.
Default	/DF	Specify default library file type.
Delete Global	/DG	Delete a library module entry point.
Entry Point	/EP	Include entry point elements in the library entry point table.
	/-EP	Exclude entry point elements in the library entry point table.
Extract	/EX	Extract (read) one or more modules from a library file and write them into a specified output file.
Insert	/IN	Insert a module.
List	/LI	List module names.
	/LE	List module names and entry points.
	/FU	List module names and full module description.
Modify Header	/MH	Modify a universal module header.
Replace	/RP	Replace a module.
	/-RP	Do not replace a module.
Selective Search	/SS	Set the selective search attribute in the module header.
Squeeze	/SZ	Reduce the size of the macro source.
	/-SZ	Do not reduce the size of a specific macro source.

7.5.1 Compress Switch (/CO)

Use the Compress switch (/CO) to physically delete all logically deleted records, to put all free space at the end of the file, and to make the free space available for new library module inserts. Additionally, the library table specification may be altered for the resulting library. LBR accomplishes this by creating a new file that is a compressed copy of the old library file. The old library file is not deleted after the new file is created.

The /CO switch can be appended only to the output file specification. The format for specifying the /CO switch is:

```
outfile/CO:size:ept:mnt=infile
```

outfile

Specifies the file that is to become the compressed version of the input file. The default file type is .OLB if the input file is an object library, .MLB if the input file is a macro library, or .ULB if the input file is a universal library.

/CO

Specifies the Compress switch.

:size

Specifies the size of the new library file in 256 (decimal)-word blocks. The size of the old library file is the default size.

:ept

Specifies the number of Entry Point Table (EPT) entries to allocate. If the value specified is not a multiple of 64 (decimal), the next highest multiple of 64 (decimal) is used. The number of EPTs in the old library file is the default value. This parameter is always forced to zero for macro libraries and universal libraries. The maximum number of entries is 4096 (decimal).

:mnt

Specifies the number of Module Name Table (MNT) entries to allocate. If the value specified is not a multiple of 64 (decimal), the next highest multiple of 64 (decimal) is used. The number of MNTs in the old library file is the default value. The maximum number of entries is 4096 (decimal).

infile

Specifies the library file to be compressed. The default file type is .OLB for object libraries, .MLB for macro libraries, and .ULB for universal libraries. The actual default file type is determined by the current default library file type (see Section 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 (decimal) blocks
EPT  = 128 (decimal) entry points
MNT  = 64 (decimal) module names
```

The new file, RICKLIB.OLB, receives a version number that is one version greater than the latest version for the file.

Both files, RICKLIB.OLB and SHEILA.OLB, reside in the default directory file on SY0:.

7.5.2 Create Switch (/CR)

Use the Create switch (/CR) to allocate a contiguous library file. The switch initializes the library file header, the entry point table, and the module name table.

The /CR switch can be appended only to the output file specification. The format for specifying the /CR switch is:

```
outfile/CR:size:ept:mnt:libtype:infiletype
```

outfile

Specifies the library file being created. The default file type is .OLB if an object library is being created, .MLB if a macro library is being created, or .ULB if a universal library is being created.

/CR

Specifies the Create switch.

:size

Specifies the new library file size in disk (256 decimal word) blocks. The default size is 100 (decimal) blocks.

:ept

Specifies the number of Entry Point Table (EPT) entries to allocate. The default value is 512 (decimal) for object libraries. This parameter is always forced to zero for macro libraries and universal libraries. The maximum number of entries is 4096 (decimal).

:mnt

Specifies the number of Module Name Table (MNT) entries to allocate. The default value is 256 (decimal). The maximum number of entries is 4096 (decimal).

:libtype

Specifies the type of library to be created. Acceptable values are OBJ for object libraries, MAC for macro libraries, and UNI for universal libraries. The default is the last value specified or implied with the /DF switch (see Section 7.5.4), or OBJ if /DF has not been specified.

:infiletype

Specifies the default input file type for the created universal library. If this value is not specified, the default input file type for universal libraries is .UNI. This value is not defined for object or macro libraries.

If the values specified for EPT and MNT are not multiples of 64 (decimal), EPT and MNT are automatically filled out to the next disk block boundary.

Example

```
LBR>RICKLIB/CR::128.:64.:OBJ=SHEILA,LAURA,JENNY
```

In this example, a combination of functions is performed. First, the library file RICKLIB.OLB is created in the default directory on SY0:. RICKLIB has the following attributes:

```
size = 100 (decimal) blocks (default size)
EPT  = 128 (decimal) entry points
MNT  = 64 (decimal) module names
type = .OBJ
```

Secondly, object modules from the input files SHEILA.OBJ, LAURA.OBJ, and JENNY.OBJ, which reside in the default directory on SY0:, are inserted into the newly created library file. Insert (/IN) is the default switch for input files (see Section 7.5.8).

7.5.3 Delete Switch (/DE)

Use the Delete switch (/DE) to logically delete library modules and their associated entry points (global symbols) from a library file. Up to 15 (decimal) 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 terminal screen:

```
MODULES DELETED:
```

As modules are logically deleted from the library file, the module name is printed on the terminal screen. (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 terminal screen and the processing of the current command is terminated. This message is as follows:

```
LBR -- *FATAL*-NO MODULE NAMED "name"
```

The /DE switch can be appended only to the library file specification.

When LBR deletes a module from a library file, the module is not physically removed from the file, but is marked for deletion. This means that, although the module is no longer accessible, the file space it once occupied is not available (unless the deleted module was the last one inserted). To physically remove the module from the file and make the freed space available, you must compress the library (see Section 7.5.1).

The format for specifying the /DE switch is:

```
outfile/DE:module1[:module2...:modulen]
```

outfile

Specifies the library file.

/DE

Specifies the Delete switch.

:module

Specifies the name of the module to be deleted.

Example

```
LBR>RICKLIB/DE:SHEILA:LAURA:JENNY

MODULES DELETED:

SHEILA

LAURA

JENNY
```

In this example, the modules SHEILA, LAURA, and JENNY and their associated entry points are deleted from the latest version of library file SY0:RICKLIB.OLB.

7.5.4 Default Switch (/DF)

Use the Default switch (/DF) to specify the default library file type. Acceptable default values are OBJ for object libraries, MAC for macro libraries, and UNI for universal libraries.

When a default library file type is not specified by the /DF switch, OBJ is the default library file type.

Specifying a default value:

1. Sets the default file type for the Create switch (/CR).

2. Provides a file type default value of .MLB for macro libraries, .ULB for universal libraries, and .OLB for object libraries when opening an output (library) file. Exceptions to this occur when you use /CO or /CR. When you specify /CO, the default applies to the library input file. When you specify /CR, the default file type is .OLB if an object library is being created, .ULB if a universal library is being created, or .MLB if a macro library is being created.

   The /DF switch only affects the file type of the file to be opened. After that, the library header record information is used to determine the type of library file being processed.

The /DF switch can be issued alone or appended to a library file specification. The format for specifying the /DF switch is:

outfile/DF:filetype...

or

/DF:filetype

outfile

Specifies the library file.

/DF

Specifies the Default switch.

filetype

Specifies the default library file type: OBJ for object library files, MAC for macro library files, and UNI for universal library files.

If a value other than OBJ, ULB, or MAC is specified, the current default library type will be set to object libraries and the following message will be displayed:

```
LBR -- *FATAL*-INVALID LIBRARY TYPE SPECIFIED
```

Examples

```
LBR>/DF:MAC
LBR>RICKLIB=infile
```

The file RICKLIB.MLB is opened for insertion.

```
LBR>/DF:MAC
LBR>RICKLIB/DF:OBJ=infile
```

OBJ replaces MAC as the default filetype. The file RICKLIB.OLB is opened for insertion.

```
LBR>/DF:MAC
LBR>RICKLIB/CR
```

The macro library RICKLIB.MLB is created.

```
LBR>/DF:MAC
LBR>RICKLIB/CR::::OBJ
```

Because OBJ is specified, it overrides the default (MAC). The object library RICKLIB.OLB is created.

```
LBR>/DF:OBJ
LBR>TEMP/CO=RICKLIB.MLB
```

Because RICKLIB.MLB is a macro library, MAC overrides the default (OBJ). The macro library file TEMP.MLB is created to receive the compressed output.

```
LBR>/DF:UNI
LBR>RICHLIB=TEST
```

The file RICHLIB.ULB is opened for insertion.

7.5.5 Delete Global Switch (/DG)

Use the Delete Global switch (/DG) to delete a specified entry point (global symbol) from the EPT. Up to 15 (decimal) entry points may be deleted with one command. This command does not affect the object module that contains the actual symbol definition. You may wish to delete an entry point if a module to be inserted has the same entry point.

When LBR begins processing the /DG switch, it prints the following message on the terminal screen:

ENTRY POINTS DELETED:

As entry points are deleted from the library file, the entry point is printed on the terminal screen. (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 terminal screen and the processing of the current command is terminated. This message is as follows:

LBR -- *FATAL* - NO ENTRY POINT NAMED "name"

The /DG switch can only be appended to the library file specification.

The format for specifying the /DG switch is:

outfile/DG:global1[:global2...:globaln]

outfile

Specifies the library file.

/DG

Specifies the Delete Global switch.

global

Specifies the name of the entry point to be deleted.

Example

LBR>RICKLIB/DG:SHEILA:LAURA:JENNY

ENTRY POINTS DELETED:

SHEILA

LAURA

JENNY

In this example, the entry points SHEILA, LAURA, and JENNY are deleted from the latest version of the library file named SY0:RICKLIB.OLB.

7.5.6 Entry Point Switch (/EP)

Use the Entry Point switch (/EP) to control (include or exclude) the placement of global symbols in a library entry point table. The switch can be specified in either a positive or negative format:

/EP	Include entry points in the entry point table.
/-EP	Do not include entry points in the entry point table.
/NOEP	Do not include entry points in the entry point table.

The positive format (/EP) causes all entry points in a module or modules to be entered in the library entry point table.

Either negative format (/-EP or /NOEP) provides for a module to be included in a library, but excludes the entry points in that module from being entered in the library entry point table.

/EP is the LBR default. If the switch is not specified, all entry points are entered into the library entry point table.

The /EP switch has no effect on macro or universal libraries.

The format for specifying the /EP switch is:

```
outfile[ /EP ]=infile,...infilen
       [/-EP ]
       [/NOEP]

              or

outfile=infile[ /EP ][,...infilen[/EP ]
              [-/EP ]            [-/EP]
              [/NOEP]            [/NOEP]
```

outfile

Specifies the output file. When the entry point switch is applied to this file specification, LBR assumes each of the input files contains modules for which entry points are to be either included or excluded.

infile

Specifies an input file. When the /EP switch is applied to an input file specification, LBR assumes only the input files to which the switch is applied contain modules for which entry points are to be either included or excluded.

NOTE

Although not reflected in the command formats, the positive and negative forms of the switch may be applied to both the output and input file specifications. For example, the effect of /EP applied to the output file can be overridden by applying /-EP to a specific input file.

The /-EP switch is useful for including modules that contain duplicate entry point names in the same library. The /-EP switch provides the means for entering a module in the library without having its entry points included in the library entry point table.

The /-EP switch is also useful in the case where the Task Builder uses only module names to search for modules in an object module library. In this case, entries in the library entry point table are not required. The /-EP switch can be used to exclude entry points from being entered in the library entry point table.

Depending on whether the /EP switch is applied to the output specification or to an input specification, it has either a global or local effect.

When applied to the output file specification, the /EP switch has a global effect. That is, LBR either includes all entry points in the entry point table or excludes all entry points from being entered in the entry point table.

When applied to an input file specification, the Entry Point switch has a local effect. That is, LBR either includes entry points in the entry point table, or excludes entries from being entered in the entry point table for only those modules to which the switch is applied.

Entry points in an object module are not affected by the /EP switch. The switch only affects entries in the library entry point table.

7.5.7 Extract Switch (/EX)

Use the Extract switch (/EX) to extract (read) one or more modules from an object or macro library file and write them into a specified output file. If more than one module is extracted, the modules are concatenated in the output file. The extract operation has no effect on the library file from which the modules are extracted; that file remains intact. Up to eight modules may be specified in one extract operation for object and macro libraries. However, only one module may be specified in one extract operation for a universal library.

For object and macro libraries, if no modules are specified in the command line, all modules in the library are extracted and concatenated in the output file in alphabetical order.

For universal libraries, RMS fields cannot be extracted to a record-oriented device, such as a terminal.

The /EX switch may be applied only to input file specifications. The format for specifying the /EX switch is:

```
outfile=infile/EX[:modulename1...:modulenamen]
```

outfile

Specifies the file into which extracted modules are to be stored. The default file type for this file is .OBJ if the input modules are object modules. The default file type is .MAC if the input modules are macro modules. If the library is a universal library, the outfile retains the infile type of the module extracted. (However, you are allowed to extract only one universal library module at a time.)

infile

Specifies the library file from which the modules are to be extracted. The default file type is .ULB, .OLB, or .MLB, depending on the current default library type.

/EX

Specifies the Extract switch.

modulename

Specifies the name of the module to be extracted from the library.

Examples

```
LBR>DRIVERS=RSX11M/EX:DXDRV:DKDRV:TTDRV
```

The object modules DXDRV, DKDRV, and TTDRV are concatenated in alphabetical order and written into the file DRIVERS.OBJ.

```
LBR>TI:=[1,5]RSXMAC.SML/EX:QIO$S
```

The macro QIO$S is written to the terminal screen.

```
LBR>TEST.OBS=TEST/EX
```

All of the modules in the library TEST.OLB are written into the file TEST.OBS in alphabetical order.

7.5.8 Insert Switch (/IN) for Object and Macro Libraries

Use the Insert switch (/IN) to insert modules into a library file. Any number of input files can be specified. For object and macro libraries, each input file can contain any number of concatenated input modules. For macro libraries, only first-level macro definitions are extracted from the input files. All text outside of the first-level macro definitions is ignored. (The Insert switch for Universal Libraries, is explained in Section 7.5.9.) The /IN switch is the default library file option and can be appended only to the library file specification.

If you attempt to insert an input module that already exists in the library file, the following message is printed on the terminal screen:

```
LBR -- *FATAL* DUPLICATE MODULE NAME "name" IN filename
```

Likewise, if you attempt to insert a module and a module contains an entry point that duplicates one that is already in the EPT, the following message is printed on the terminal screen:

```
LBR -- *FATAL* DUPLICATE ENTRY POINT "name" IN filename
```

The format for specifying the /IN switch is:

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

outfile

Specifies the library file into which the input modules are to be inserted. The default file type depends on the current default (see Section 7.5.4). It is .OLB if the current default is object libraries, .MLB if the current default is macro libraries.

/IN

Specifies the Insert switch.

infile

Specifies the input file containing the modules to be inserted into the library file. The default file type is .OBJ if the outfile is an object library and .MAC if the outfile is a macro library.

Example

```
RICKLIB/IN=SHEILA,LAURA,JENNY
```

In this example, the modules contained in the latest versions 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 SHEILA, LAURA, and JENNY is .OBJ if RICKLIB is an object module library, or .MAC if RICKLIB is a macro library.

7.5.9 Insert Switch (/IN) for Universal Libraries

The Insert switch (/IN) works basically the same for universal libraries as it does for object and macro libraries. However, when inserting a file into a universal library, the /IN switch is applied to the input file rather than the output file. You can also specify module name and descriptive information as switch values in the command line. In addition, LBR copies input file attributes to the module header.

The high block indicator (F.HIBK of the file's descriptor block) and the end of file indicator (F.EFBK of the file's FDB) are included in the input file's user file attributes. LBR makes the high block indicator equal to the end of file indicator in the module header. This means that when a module is extracted to a file, that file will have as many blocks allocated to it as are used.

The format for specifying the /IN switch for universal libraries is:

```
outfile=infile/IN:name:op:op:op:op
```

outfile

Specifies the universal library into which the infile is to be inserted.

infile

Specifies the input file to be inserted into the outfile. The default for the file type is the value indicated at the universal library's creation time. See Section 7.5.2.

/IN

Specifies the Insert switch.

:name

Optionally specifies the module name (up to six Radix-50 characters). The default is the first six characters of the input file name.

:op

Specifies optional descriptive information (up to six Radix-50 characters) to be stored in the module header. The default is null. If only part of the information set is specified, all preceding colons must be supplied.

Example

```
LBR>RICKLIB.ULB=JOE.TXT/IN:MOD1:THIS:IS:JAN2:TEXT
```

In this example, LBR inserts JOE.TXT into the universal library RICKLIB.ULB as MOD1. "THIS", "IS", "JAN2", and "TEXT" are stored in the module header.

You can insert JOE.TXT without the /IN switch and its values. As a result, all the information normally specified by the switch values defaults as described in this example.

7.5.10 List Switches (/LI, /LE, /FU)

Use the list switches to produce a printed listing of the contents of a library file. Three switches allow you to select the type of listing desired. These switches are as follows:

/LI	Produces a listing of the names of all modules in the library file.
/LE	Produces a listing of the names of all modules in the library file and their corresponding entry points.
/FU	Produces a listing of the names of all modules in the library file and gives a full module description for each: that is, size, date of insertion, and module-dependent information.

These switches can be appended only to the output file specification or the list file specification.

The /LI switch is the default value. It need not be specified when a listing file has been specified or when any other list switch is included in the command line.

The format for specifying list switches is:

```
outfile[,listfile]/switch(es)
```

outfile

Specifies the library file whose contents are to be listed.

listfile

Optionally specifies the listing file. If not specified, the listing is directed to the terminal screen.

/switch(es)

Specifies the list option(s) selected.

Examples

```
LBR>RICKLIB/LI
```

In this example, a listing of the names of all the modules contained in file SY0:RICKLIB.OLB is printed on the terminal screen.

```
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 terminal screen.

```
LBR>RICKLIB/FU
```

In this example, a listing of the names of all the modules in file SY0:RICKLIB.OLB, and a full description of each one contained is printed on the terminal screen.

```
LBR>DW1:[FOOBAR]RICKLIB,LP.LST/LE/FU
```

In this example, LBR creates file LP.LST in directory [FOOBAR] on DW1, which lists the module names, their entry points, and a full description of each module for file RICKLIB.

7.5.11 Modify Header Switch (/MH)

The Modify Header switch (/MH) pertains only to universal libraries and allows you to modify the optional user-specified information in the module header.

The format for specifying the /MH switch is:

```
outfile/MH:module:op:op:op:op
```

outfile

Specifies an output file for the universal library. The file type defaults to .ULB.

/MH

Specifies the Modify Header switch.

:module

Specifies the name of the module whose descriptive information is to be modified.

:op

Specifies the optional user information (up to six Radix-50 characters) to be stored in the module header. The default is null and indicates that the corresponding information field is not to be changed. Also, entering a pound sign (#) clears the corresponding information field.

Example

The optional descriptive information for module A of RICKLIB.ULB is:

"MODA" "FCHCD" "OF" "FCH"

The LBR command is:

```
LBR>RICKLIB/MH:A:FCHTS:#::
```

The optional descriptive information for module A in file RICKLIB is changed to:

"FCHTS" " " "OF" "FCH"

7.5.12 Replace Switch (/RP) For Macro and Object Libraries

Use the Replace switch (/RP) to replace modules in a library file with input modules of the same name. Any number of input files are allowed and each file can contain any number of concatenated input modules.

For macro libraries, only first-level macro definitions are extracted from the replacement files.

When a match occurs on a module name, the existing module is logically deleted and its entries are removed from the EPT.

As each module in the library file is replaced, a message is printed on the terminal screen. This message, which contains the name of the module being replaced, is as follows:

```
MODULE "name" REPLACED
```

If the module to be replaced does not exist in the library file, LBR assumes that the input module is to be inserted and automatically inserts it without printing a message.

The /RP switch can be specified in either of the following formats:

- Global format - The /RP switch is appended to the library file specification and all of the input files are assumed to contain replacement modules.
- Local format - The /RP switch is appended to an input file specification and only the file to which the /RP switch is appended is considered to contain replacement modules.

Global Format

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

outfile

Specifies the library file. The default file type depends on the current default (see Section 7.5.4). It is .OLB if the current default is object libraries or .MLB if the current default is macro libraries.

/RP

Specifies the Replace switch.

infile

Specifies the input file that contains replacement modules for the library file. The default type is .OBJ if outfile is an object library or .MAC if it is a macro library.

The Global format allows you to specify a list of input files without having to append the /RP switch to each of them.

To override the global function for a particular input file (that is, to instruct LBR to process a particular file in a list as a file containing modules to be inserted but not replaced), append /-RP or /NORP to the desired input file specification.

Local Format

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

outfile

Specifies the library file. The local format default is the same as the global format default.

infile

Specifies the input file that contains replacement modules for the output library file. The local format default is the same as the global format default.

/RP

Specifies the Replace switch. Appending the /RP switch to an input file specifier constitutes the local format of the switch. This overrides the LBR default (/IN) and instructs LBR to treat the module(s) contained in the specified file as replacement modules.

Examples

The files used in the following four examples, and the modules contained within each file, are depicted in Figure 7-8. These files are assumed to reside in the default directory on the default device and the initial state of the library file is assumed to be as shown in Figure 7-8.

1.
```
LBR>RICKLIB/RP=SHEILA,LAURA,JENNY

MODULE "SHEILA" REPLACED
MODULE "LAURA1" REPLACED
MODULE "LAURA2" REPLACED
MODULE "JENNY1" REPLACED

MODULE "JENNY2" REPLACED
```

This example uses the global format for the /RP switch. Object modules from the input files SHEILA, LAURA, and JENNY replace modules by the same names in the library

file RICKLIB, and modules JENNY3 and LAURA3 are inserted. Figure 7-9 shows the resulting library file.

2. LBR>RICKLIB=CHRIS,SHEILA/RP

```
MODULE "SHEILA" REPLACED
```

In this example, the local format of the /RP switch is used. The object module SHEILA from file SHEILA is replaced in the library file RICKLIB. The object modules in the file CHRIS are inserted in the library file. (See the description of the /IN switch in Section 7.5.8.) The resulting library file is shown in Figure 7-10.

3. LBR>RICKLIB/RP=SHEILA,LAURA,JENNY,CHRIS/-RP

```
MODULE "SHEILA" REPLACED
MODULE "LAURA1" REPLACED
MODULE "LAURA2" REPLACED
MODULE "JENNY1" REPLACED
MODULE "JENNY2" REPLACED
```

In this example, the /-RP switch is used to override the global format of the command. Object modules in files SHEILA, LAURA, and JENNY are processed as modules to be replaced, and file CHRIS is processed as a file that contains modules to be inserted. The resulting library file is shown in Figure 7-11.

4. LBR>RICKLIB/RP=SHEILA,LAURA/-RP,JENNY

```
MODULE "SHEILA" REPLACED
LBR -- *FATAL* -- DUPLICATE MODULE "LAURA1" IN
LAURA.OBJ;1
```

In this example, only module SHEILA from file SHEILA was replaced. The user specified that the modules in file LAURA not be replaced (/-RP), but inserted. One of the modules contained in file LAURA duplicated an already existing module in file RICKLIB. Therefore, LBR issued the fatal error message and terminated the processing of the current command line.

	OUTPUT LIBRARY FILES	INPUT FILES			
FILE NAME	RICKLIB.OLB;1	SHEILA.OBJ;1	LAURA.OBJ;1	JENNY.OBJ;1	CHRIS.OBJ;1
OBJECT MODULES	JENNY1 JENNY2 LAURA1 LAURA2 SHEILA	SHEILA	LAURA1 LAURA2 LAURA3	JENNY1 JENNY2 JENNY3	CHRIS1 CHRIS2

ZK-191-81

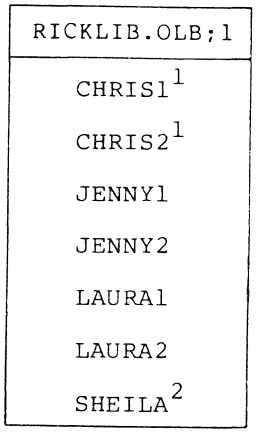
Figure 7-8: Sample Files Used in LBR Examples 1-4

RICKLIB.OLB;1
JENNY1
JENNY2
JENNY3[1]
LAURA1
LAURA2
LAURA3[1]
SHEILA

1. These modules did not exist in the library file prior to the execution of this example, but they did exist in the input files. LBR, therefore, assumed that they were to be inserted. Since LBR handled these modules as a normal insert, no message was printed on the input terminal.

ZK-192-81

Figure 7-9: Output Library File After Execution of Example 1

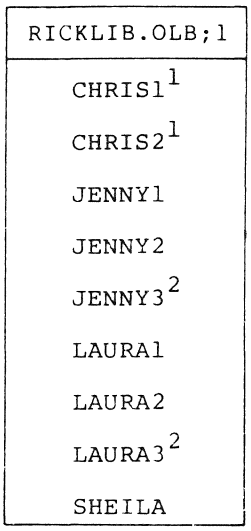


1. These modules are inserted.

2. This module is replaced.

ZK-193-81

Figure 7-10: Output Library File After Execution of Example 2



1. These modules were specified to be inserted. Had a module of the same name been present, a fatal error message would have been issued. See Example 4.

2. These modules were inserted by default.

ZK-194-81

Figure 7-11: Output Library File After Execution of Example 3

7.5.13 Replace Switch (/RP) for Universal Libraries

Use the /RP switch for universal libraries in the same way as for macro and object libraries. However, you can also specify the same values for the /RP switch as for the /IN switch for universal libraries. See Section 7.5.9.

As with macro and object libraries, you can specify the /RP switch with either the output file specification or with the input file specifications.

The global format of the /RP switch for universal libraries is:

```
outfile/RP:name:op:op:op:op=infile[,infile2,....infilen]
```

The local format of the /RP switch for universal libraries is:

```
outfile=infile/RP:name:op:op:op:op[,infile2....infilen]
```

outfile

Specifies the universal library file.

infile

Specifies the input file that contains replacement modules for the library file. The default for the file type is the value indicated at the universal library's creation time. See Section 7.5.2.

/RP

Specifies the Replace switch.

:name

Optionally specifies the module name to be replaced (up to six Radix-50 characters). The default is the first six characters of the infile name.

:op

Specifies optional descriptive information (up to six Radix-50 characters) to be stored in the module header. The default is null. If only part of the information set is specified, all preceding colons must be supplied.

Example

```
LBR>TEXT.ULB=DEBBIE.TXT/RP::THIS:IS:JAN3:UPDATE

MODULE "DEBBIE" REPLACED
```

In this example, LBR replaces the module DEBBIE in the universal library TEXT.ULB with an updated module from file DEBBIE.TXT. The date of replacement is specified by the optional user information and inserted in the module header. Note that the optional name is omitted.

The initial state of the library file is shown in Figure 7-12. The resulting library file is shown in Figure 7-13.

	OUTPUT LIBRARY FILE	INPUT FILES
FILE NAME	TEXT.ULB;1	DEBBIE.TXT
MODULES	DEBBIE BERNIE	

ZK-195-81

Figure 7-12: Sample Files for Universal Library Replace Example

TEXT.ULB;1
DEBBIE[1] BERNIE

1.The module DEBBIE was replaced. If a different infile were specified, that file would become module DEBBIE and occupy the same location in TEXT.ULB.

ZK-196-81

Figure 7-13: Output Library File After Library Replace

7.5.14 Selective Search Switch (/SS)

Use the Selective Search switch (/SS) to set the selective search attribute bit in the module header of object modules as they are inserted into an object library. The switch has no effect when applied to modules being inserted into a macro library. The switch may be specified with input files for insertion or replacement operations only, and it affects all modules in the input file to which it is applied.

Object modules with the selective search attribute are given special treatment by the Task Builder. Global symbols, defined in modules with the selective search attribute, are only included in the Task Builder's symbol table if they are previously referenced by other modules. Thus, only referenced symbols will be listed with the module in the Task Builder memory allocation file, thereby reducing task build time. The /SS switch should only be applied to object files whose modules contain only absolute (not relocatable) symbol definitions.

The format for specifying the /SS switch is:

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

outfile

Specifies the library file.

infile

Specifies the input file that contains modules to be selectively searched.

/SS

Specifies the Selective Search switch.

Example

```
LBR>ANGEL=JOHN,JILL/SS,MARK/SS,MARY
```

The object files JOHN.OBJ, JILL.OBJ, MARK.OBJ, and MARY.OBJ are inserted into object library ANGEL.OLB. The selective search attribute bit is set in both the JILL and MARK object module header.

7.5.15 Squeeze Switch (/SZ)

Use the Squeeze switch (/SZ) to reduce the size of macro definitions by eliminating all trailing blanks and tabs, blank lines, and comments from macro text. The /SZ switch is used to conserve memory in the MACRO-11 Assembler and to reduce the size of macro library files. The /SZ switch has no effect on object libraries or universal libraries.

The /SZ switch can be specified in either of two formats:

- Global format - The /SZ switch is appended to the library file specification. All of the input files are assumed to contain modules to be squeezed.
- Local format - The /SZ switch is appended to an input file specification. The /SZ switch works only on the file to which you append it.

Global Format

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

outfile

Specifies the library file.

/SZ

Specifies the Squeeze switch.

infile

Specifies the input file that contains modules to be squeezed during insertion into the library file.

Use the global format of the /SZ switch to specify a list of input files without having to append the /SZ switch to each of them. To override the global function for a particular input file (that is, to instruct LBR to process a particular file in a list as a file containing modules to be inserted but not squeezed), append /-SZ or /NOSZ to the desired input file specification.

Local Format

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

outfile

Specifies the library file.

infile

Specifies the file that contains modules to be squeezed during insertion into the library file.

/SZ

Specifies the Squeeze switch.

LBR uses the following algorithm on each line to be squeezed and then inserts the resulting line into the library file:

1. The line is examined for the rightmost semicolon (;).
2. If a semicolon is located, it is deleted, along with all trailing characters in the line.
3. All trailing blanks and tabs in the line are deleted.
4. If the resulting line is null, nothing is transferred to the library file.

If the line contains a semicolon embedded in noncomment text and you want comments squeezed, code a dummy comment for that line. The /SZ switch will use only the rightmost comment during squeeze processing.

Example

Figure 7-14 illustrates the use of the LBR /SZ switch. A file containing input text to be squeezed is illustrated, along with the text actually inserted into the library file after the squeeze operation has been completed.

```
                              BEFORE

        .MACRO  MOVSTR RX,RY,?LBL

;***    - - NOTE :                                          ;
;       BOTH ARGUMENTS MUST BE REGISTERS                    ;

LBL:    MOVB    (RX)+,(RY)+     ;MOVE A CHARACTER
        BNE     LBL             ;CONTINUE UNTIL NULL SEEN
        DEC     RY              ;BACKUP OUTPUT PTR TO NULL

;END OF MOVSTR
        .ENDM


                              AFTER

        .MACRO  MOVSTR RX,RY,?LBL
;***    - - NOTE :
;       BOTH ARGUMENTS MUST BE REGISTERS
LBL:    MOVB    (RX)+,(RY)+
        BNE     LBL
        DEC     RY
        .ENDM
```

ZK-197-81

Figure 7-14: MACRO Listing Before and After Running LBR with /SZ

7.6 COMBINING LIBRARY FUNCTIONS

Two or more library functions may be requested in the same command line. The only exceptions are that /CO cannot be requested with anything else except /LI, and /CR and /DE cannot be specified in the same command line.

Functions are performed in the following order:

1. Default switch (/DF)
2. Create switch (/CR)
3. Delete switch (/DE)
4. Delete Global switch (/DG)
5. Modify Header switch (/MH)
6. Insert (/IN), Replace (/RP), Selective Search (/SS), Squeeze (/SZ), Entry Point (/EP) switches
7. Compress switch (/CO)
8. Extract switch (/EX)
9. List switches (/LI), /LE, /FU)

Example

```
LBR>FILE/DE:XYZ:$A,FILE.LST:/LE/FU=MODX,MODY/RP
```

Functions, performed in order, are:

1. Delete modules XYZ and $A.

2. Insert all modules from MODX and replace duplicate modules of MODY.

3. Produce a listing of the resultant library file on the line printer with full module descriptions and all entry points.

7.7 LBR ERROR MESSAGES

LBR returns two types of error messages: diagnostic and fatal.

Diagnostic error messages describe a condition that requires consideration, but the nature of the condition does not warrant termination of the command. Diagnostic messages are issued to your terminal screen in the format:

```
LBR -- *DIAG* - message
```

Fatal error messages describe a condition that caused LBR to terminate the processing of a command. When this occurs, LBR returns to the highest level of command input. For example, LBR issues the fatal error message and exits if you enter the command:

```
$ RUN $LBR/COMMAND:"LBR cmd"
```

where cmd is an LBR command.

If, however, you enter the command in response to the LBR prompt, that is,

```
LBR>cmd
```

LBR issues the fatal error message and reprompts.

Fatal error messages are issued to your terminal screen in the format:

LBR -- *FATAL* - message

If a fatal error occurs during the processing of an indirect command file, the command file is closed, the fatal error message--followed by the command line in error---is issued to your terminal screen and LBR returns to the highest level of command input.

7.7.1 Effect of Fatal Errors on Library Files

The status of a library file after fatal errors is:

1. In general, output errors leave the library in an indeterminate state.

2. During the deletion process, the library is rewritten prior to the printing of the individual module-/entry-point-deleted messages.

3. During the replacement process, the library is rewritten prior to the printing of the individual module-replaced messages.

4. During the insertion process, the library is rewritten after the insertion of all modules in each individual input file, that is, between input files.

7.7.2 LBR Error Messages

LBR -- BAD LIBRARY HEADER

Explanation: The file is not a library file or it is corrupted.

User Action:

- If the file is not a library file, reenter the command line with a proper library file specified.

- If the file is a proper library file, the volume may be corrupted.

- If the volume is corrupted, it must be reconstructed before it can be used.

LBR -- CANNOT MODIFY HEADER

Explanation: An attempt was made to modify the module header of a module in an object library or macro library. No change is made to the module header.

User Action: Reenter the command line, specifying a module in a universal library.

LBR -- COMMAND I/O ERROR

Explanation: One of the following conditions may exist:

- A problem exists on the physical device.
- The file is corrupted or the format is incorrect (for example, record length exceeds 132 bytes).

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

LBR -- COMMAND SYNTAX ERROR
command line

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

User Action: Reenter the command line, using the correct syntax.

LBR -- DUPLICATE ENTRY POINT NAME "name" IN filename

Explanation: An attempt was made to insert a module into a library file when both contain an identically named entry point.

User Action: Determine if the specified input file is the correct file. If not, reenter the command line, specifying the correct input file. If the input file is the correct file, you can delete the duplicate entry point from the library and reenter the command line.

LBR -- DUPLICATE MODULE NAME "name" IN filename

Explanation: An attempt was made to insert (without replacing) a module into a library that already contains a module with the specified name.

User Action: Determine if the specified input file is the correct file. If the input file is correct, decide whether to delete the duplicate module from the library file and insert the new one, or replace the duplicate module with the /RP switch appended to the input file specification.

LBR -- EPT OR MNT EXCEEDED IN filename

Explanation: The EPT or MNT table limit was reached during the execution of an insert or replace operation.

User Action: Copy the library, increasing the table space by means of the Compress switch. Reenter the command line.

LBR -- EPT OR MNT SPACE EXCEEDED IN COMPRESS

Explanation: An EPT or MNT table size was specified for the output library file that is not large enough to contain the EPT or MNT entries used in the input library file.

User Action: Reenter the command line with a larger EPT or MNT table size specified.

LBR -- ERROR IN LIBRARY TABLES, FILE filename

Explanation: The library file is corrupted or is not a library file.

User Action: If the file is corrupted, no recovery is possible; the file must be reconstructed. If the file is not a library file, reenter the command line with the correct library file specified.

LBR -- EXACTLY ONE INPUT FILE MUST APPEAR WITH /CO

Explanation: No input library file, or more than one file, was specified when using the /CO switch.

User Action: Reenter the command line with only one input file specified.

LBR -- FATAL COMPRESS ERROR

Explanation: The input library file is corrupted or is not a library file.

User Action: No recovery is possible. The file in question must be reconstructed.

LBR -- GET TIME FAILED

Explanation: This error occurs when LBR attempts to execute a Get Time Parameters directive and fails. The error is caused by a system malfunction.

User Action: Reenter the command line. If the problem persists, call your DIGITAL Customer Support Center.

LBR -- ILLEGAL DEVICE/VOLUME
command line

Explanation: The Device specifier entered does not conform to syntax rules. A device specifier consists of two ASCII characters, followed by one or two optional octal digits.

User Action: Reenter the command line with the correct device syntax specified and followed by a colon.

LBR -- ILLEGAL DIRECTORY
command line

Explanation: The directory entered does not conform to syntax rules. Directory syntax consists of a left square bracket, followed by one to nine alphanumeric characters (or one to three octal digits, a comma, and one to three octal digits), terminated by a right square bracket (for example, [USERFILES]).

User Action: Reenter the command line with the correct directory syntax.

LBR -- ILLEGAL FILENAME
command line

Explanation: One of the following was entered:

- A file specifier that contains a wildcard.

- A file specifier that contains neither a file name nor a file type.

User Action: Reenter the command line correctly.

LBR -- ILLEGAL GET COMMAND LINE ERROR CODE

Explanation: LBR could not process the command line.

User Action: Reenter the command line. If the problem persists, call your DIGITAL Customer Support Center.

LBR -- ILLEGAL SWITCH
command line

Explanation: A non-LBR switch was specified or a legal switch was specified in an invalid context.

User Action: Reenter the command line with the correct switch specification.

LBR -- ILLEGAL SWITCH COMBINATION

Explanation: Switches were entered that cannot be executed in combination. See Section 7.6.

User Action: Reenter the command line, specifying the switches in the proper combination.

LBR -- INDIRECT COMMAND SYNTAX ERROR
command line

Explanation: An indirect command file was specified in a format that does not conform to syntax rules.

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

LBR -- INDIRECT FILE DEPTH EXCEEDED
command line

Explanation: An attempt was made to exceed one level of indirect command files.

User Action: Rerun the job with only one level of indirect command file specified.

LBR -- INDIRECT FILE OPEN FAILURE
command line

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

- The user directory area is protected against access.
- A problem exists on the physical device.
- The volume is not mounted.
- The specified file directory does not exist.
- The file does not exist as specified.
- Insufficient dynamic memory exists in the Executive.

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

LBR -- INPUT ERROR ON filename

Explanation: The file system, while attempting to process an input file, has detected an error.

A problem exists with the physical device.

User Action: Reenter the command line.

LBR -- INSUFFICIENT DYNAMIC MEMORY TO CONTINUE

Explanation: Your library is too large for the attempted operation.

User Action: Break up the library into two or three smaller libraries.

LBR -- INVALID EPT AND/OR MNT SPECIFICATION

Explanation: An EPT or MNT value greater than 4096 (decimal) was entered in a /CR or /CO switch.

User Action: Reenter the command line with the correct value specified.

LBR -- INVALID FORMAT, INPUT FILE filename

Explanation: The format of the specified input file is not the standard format for a macro source or object file, or the input file is corrupted.

User Action: Reenter the command line with the correct input file specified.

LBR -- INVALID LIBRARY TYPE SPECIFIED

Explanation: An invalid library type was specified when using the Create or Default switch. The values OBJ, MAC, and UNI are the only valid specifications. See Sections 7.5.2 and 7.5.4.

User Action: Reenter the command line with OBJ, MAC, or UNI specified.

LBR -- INVALID MODULE FORMAT in insertion module

Explanation: An attempt was made to insert a macro module into an object library.

User Action: Determine if an object file was to be inserted into an object library. If so, reenter the command line with the correct object file. If a macro library was to receive the insertion, reenter the command line with the correct macro library.

LBR -- INVALID NAME -- "name"

Explanation: A module name that contains a non-Radix-50 character was specified for deletion, insertion, or replacement of a module in a universal library or in a macro module; or a module name was specified for modification of a universal module header. Radix-50 characters consist of the letters A through Z, the numbers 0 through 9, and the special characters period (.) and dollar sign ($).

User Action: Reenter the command line with a valid name.

LBR -- INVALID OPERATION FOR OBJECT AND MACRO LIBRARIES

Explanation: Module header information was supplied for an object library or macro library in an insert or replace operation.

User Action: No action required. The command will be executed as if the information had not been supplied.

LBR -- INVALID RAD50 CHARACTER IN "character string"

Explanation: A character supplied as part of information when using the Insert, Replace, or Modify Header switches for a universal library is not a Radix-50 character.

User Action: Determine which character of the corresponding switch value is not a Radix-50 character. Reenter a Radix-50 character in place of the invalid character.

LBR -- I/O ERROR ON INPUT FILE filename

Explanation: A read error has occurred on an input file. One of the following conditions may exist:

- A problem exists on the physical device.
- The file is corrupted or the format is wrong (record length exceeds 132 bytes).

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

LBR -- LIBRARY FILE SPECIFICATION MISSING

Explanation: A command line was entered without specifying the library file.

User Action: Reenter the command line with the library file specified.

LBR -- MARK FOR DELETE FAILURE ON LBR WORK FILE

Explanation: When LBR begins processing commands, it automatically creates a work file and marks it for delete. For some reason, this marking for delete failed.

The work file constitutes a lost file because it does not appear in any file directory.

User Action: The file may be deleted by typing the following command from the DCL command level:

```
$ @VERIFY
```

LBR -- MULTIPLE MODULE EXTRACTIONS NOT PERMITTED FOR UNI MODULES

Explanation: An attempt was made to extract more than one module from a universal library. The first module specified is extracted, but others are ignored.

User Action: Reenter the command line for each additional extraction.

LBR -- NO ENTRY POINT NAMED "name"

Explanation: The entry point to be deleted is not in the specified library file.

User Action: Determine if the entry point is misspelled or if the wrong library file is specified. Reenter the command line with the entry point or the library file correctly specified.

LBR -- NO MODULE NAMED "module"

Explanation: The module to be deleted is not in the specified library file.

User Action: Determine if the module name is misspelled or if the wrong library file is specified. Reenter the command line with the module name correctly specified.

LBR -- OPEN FAILURE ON FILE filename

Explanation: The file system, while attempting to open a file, has detected an error. One of the following conditions may exist:

- The user directory area is protected against an open operation.
- A problem exists on the physical device.
- The volume is not mounted.
- The specified file directory does not exist.

- The file does not exist as specified.
- Insufficient contiguous space to allocate the library file (Compress and Create only).
- Insufficient dynamic memory exists in the Executive.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter that command line.

LBR -- OPEN FAILURE ON LBR WORK FILE

Explanation: The file system, while attempting to open the LBR work file, has detected an error. The LBR work file is created on the volume from which LBR was installed. One of the following conditions may exist:

- The volume is full.
- The device is write-protected.
- A problem exists with the physical device.
- Insufficient dynamic memory exists in the Executive.

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

LBR -- OUTPUT ERROR ON filename

Explanation: A write error has occurred on the output file. One of the following conditions may exist:

- The volume is full.
- The device is write-protected.
- The hardware has failed.

User Action: If the volume is full, delete all unnecessary files and rerun LBR. If the device is write-protected, write-enable the device and reenter the command line. If the hardware has failed, swap devices and reenter the command line or wait until the device is repaired and rerun LBR.

LBR -- POSITIONING ERROR ON filename

Explanation: A positioning error has occurred on the input file. One of the following conditions exist:

- A problem exists on the physical device.
- The file is corrupted or the format is wrong.

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

LBR -- RMS MODULES CANNOT BE EXTRACTED TO RECORD ORIENTED DEVICES

Explanation: An attempt was made to extract a module inserted from a nonsequential RMS file to a record-oriented device. This is a fatal error message.

User Action: Extract the file to a disk and then use an RMS conversion to make an RMS sequential file.

LBR -- TOO MANY OUTPUT FILES SPECIFIED

Explanation: More than two output files were specified. LBR makes the following assumptions:

- The first output file specified is the output library file.
- The second output file specified is the listing file.
- The third through n files specified to the left of the equal sign are ignored.

User Action: No action is required. LBR continues as though the extra file(s) had not been specified.

LBR -- VIRTUAL STORAGE REQUIREMENT EXCEEDS 65536 WORDS

Explanation: This error may occur if you are working with maximum size libraries. You might have specified a single command line that first logically deletes a large number of modules and entry points, then replaces them with an equally large number of modules and entry points. The replacement modules and entry points have names much different from those being replaced. This message usually indicates a

system error.

User Action: Rerun the job, but divide the complicated command line into several smaller command lines that do the same operations.

LBR -- WORK FILE I/O ERROR

Explanation: A write error has occurred on the LBR work file. One of the following conditions may exist:

- The volume is full.
- The device is write-protected.
- The hardware has failed.

User Action: If the volume is full, delete all unnecessary files and rerun LBR. If the device is write-protected, write-enable the device and reenter the command line. If the hardware has failed, swap devices and retry the command, or wait until the device is repaired and rerun LBR.

CHAPTER 8

RESOURCE MONITORING DISPLAY (RMD)

The Resource Monitoring Display (RMD) is a privileged task that displays information about the resources in your system. This information includes the active tasks, their location in memory, the amount of memory they occupy, and available pool space.

8.1 INTRODUCTION

RMD consists of *pages*. A page consists of 24 lines. The program contains two kinds of pages: display and setup.

8.1.1 Display Pages

There are four display pages available:

- Memory (M)
- Active Task List (A)
- Task Header (T)
- Help (H)

You press the terminal keys indicated in parentheses to switch display pages.

8.1.2 Setup Pages

There are three setup pages. Each setup page is associated with a display page. (The Help Display Page has no associated setup page.) You access a setup page from a display page by pressing

the CTRL-[sequence on your terminal. The setup page documents and prompts you for setup commands, which alter the content of the information displayed on the associated display page.

8.2 INVOKING RMD

You can invoke RMD in three ways:

1. Invoke the SHOW MEMORY command, which in turn invokes RMD. See Chapter 3 for a description of the SHOW MEMORY command.

2. Invoke the SHOW TASKS/DYNAMIC command, which in turn invokes RMD. See Chapter 3 for a description of the SHOW TASKS/DYNAMIC command.

3. Invoke RMD directly from the DCL command level.

To invoke RMD directly from the DCL command level, enter the following command:

```
$ RUN $RMD
```

After entering the command, you see the memory display page on your terminal screen, indicating that you are in the RMD environment. Once you are in the RMD environment, RMD waits for your command.

RMD accepts command lines in the following general format:

```
[page][,setupcommand] . . .
```

page

One of the display page abbreviations (M, A, T, or H). The default page is the Memory Display (M).

setupcommand

A valid setup command for the display page that you have selected. The setup commands are the same as those available to you from the setup page associated with the display page you specified. The default setup commands are discussed in Sections 8.4 through 8.6, which describe the content of each display page and how you use setup commands to alter display parameters.

8.2.1 Running RMD on a Second Terminal

You can connect a second terminal to the Professional through its printer port. (For details, see the section on debugging an application in the *Tool Kit User's Guide*).

Enter the following commands, either from the keyboard or from an indirect command procedure:

```
$ SET TERMINAL /TT2:  /NOSLAVE
$ ASSIGN/TASK RMD TT2:      1
$ ASSIGN/TASK RMD TT2:      2
$ SPAWN SHOW MEMORY
```

This command sequence does the following:

- It reassigns LUNs 1 and 2 to the terminal on which you want to run the task, and
- It invokes the task on the second terminal while allowing you to continue entering DCL commands at the Professional's terminal.

NOTE

You must be a privileged user to issue the SET TERMINAL /TT2: command.

8.3 THE HELP DISPLAY

The Help Display documents how you switch display pages. You switch display pages by pressing a terminal key as follows:

Key	Explanation
M	Accesses the Memory Display
A	Accesses the Active Task Display
T	Accesses the Task Header Display

The Help Display also documents how to exit from RMD and use the CTRL-[sequence to access setup pages from their associated display pages. (Because there is nothing to alter on the Help Display, no setup page is available from the Help Display.)

The Memory, Active Task, and Task Header Displays use the entire screen. Therefore, you receive no prompts or documentation on

display pages. To find out how to access a setup page from a display page or how to switch display pages, press the H key (for Help) to access the Help Display.

8.4 THE MEMORY DISPLAY

The Memory Display graphically represents system memory. It shows the approximate size and locations of partitions and active tasks, pool statistics, the name of the currently-executing task, and other information regarding operating system status. You access the Memory Display from another display page by pressing the M key (for Memory). If you invoke RMD without specifying a display page, RMD defaults to the Memory Display.

Figure 8-1 shows the Memory Display. See the numbered notes following the figure for a description of each item in the Memory Display.

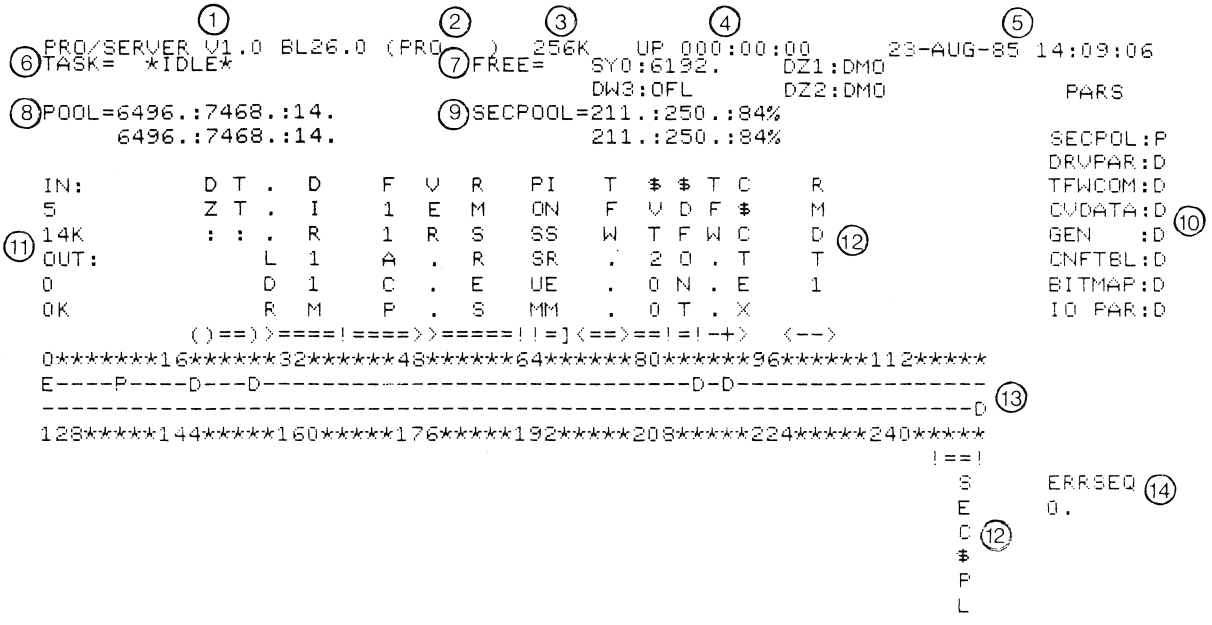


Figure 8-1: Memory Display for P/OS

Notes to Figure 8-1

(1) Operating system type, version number, and base level.

(2) A one- to six-character name that is a DECnet node name (if DECnet is running on your system).

(3) Size in K words of the system memory.

(4) Time elapsed in units of days, hours, and minutes, since the system was last booted.

(5) Current date and time.

(6) Name of the task that is currently executing or, if none is executing, *IDLE* (Executive executing the idle loop).

(7) Number of free blocks on the first four FILES-11 devices in your system. If a device is dismounted, RMD displays: "DMO". If a device is off line, RMD displays: "OFL".

(8) Pool (dynamic storage region) information in the format:

POOL=X:Y:Z

X

Number of words in the largest free block in pool

Y

Number of free words in pool

Z

Number of fragments in the pool free list

The second line records the worst case of pool since you invoked RMD. This line is most useful if RMD has been running on a second terminal since the last system boot.

(9) Secondary pool information in the format:

SECPOOL=A:B:C%

A

Number of free blocks in secondary pool

B

Total number of blocks in secondary pool

C%

Percentage of secondary pool that is free

The second line records the worst case of secondary pool since you invoked RMD. This line is most useful if RMD has been running on another terminal since the last system boot.

(10) Partitions in the system using the format:

partitionname:type

type

One of the following:

D System-controlled (dynamic) partition
P Secondary pool partition

(11) Number of tasks in memory and amount of memory they use, and number of active tasks swapped or checkpointed out of memory and the amount of memory they would require.

(12) Name of each task, common, or driver in memory and its location in memory using the following symbols to designate size, type (task, common, or driver), and other attributes:

Symbol	Attribute
< >	Active task
[ ]	Task not active, yet occupies memory
! !	Named common
+ +	Unnamed common (displayed name is first attached task)
( )	Loaded driver using device mnemonic
-----	Task not fixed in memory
=====	Task fixed in memory

The hyphens and equal signs represent the amount of memory that each task, driver, or common occupies. Where the display shows only one delimiter and no hyphens or equal signs, the open delimiter is in the same location as the closing delimiter of the preceding task.

(13) Partition size and location. The beginning of each partition is marked with the same symbols as those listed previously, plus the following additions:

E Executive
P Pool

The lines of asterisks are proportional representations of the amount of memory occupied by each partition. The numbers are in 1K word increments. Each numerical character also represents the same amount of memory as an asterisk. RMD always divides the system memory into groups of eight units.

(14) System error count sequence recorded by the Error Logger (always zero if Error Logger not present in your system).

8.4.1 Altering the Memory Display from the Setup Page

To alter the Memory Display, you press the CTRL-[sequence, which accesses the setup page for the Memory Display. The setup page documents and prompts you for commands which you use to alter the Memory Display. You can enter multiple commands after each prompt by using commas as separators. The setup commands available for altering the Memory Display are:

- FREEx=ddnn: where x is a number from 0 to 3 and ddnn: is a device name and number
- RATE=s where s is the replot rate in seconds

You can truncate these commands to their shortest unique forms. These setup parameters stay in effect until you alter them, even if you switch to another display page and back to the Memory Display.

8.4.1.1 The FREE Command - You use this command to determine the four FILES-11 devices for which you want the Memory Display to show the available free blocks. This information is (7) in Figure 8-1. The default is your system disk (SY:) and the next three FILES-11 devices in your configuration.

8.4.1.2 The RATE Command - You use this command to determine how often RMD replots the Memory Display. The default replot rate is once per second.

8.5 THE ACTIVE TASK DISPLAY

The Active Task Display shows you the active tasks in the system. You access this display from another display page by pressing the A key (for Active Task).

This display has six fields:

- Name of the task
- Length of the task in octal bytes
- Terminal that issued the task
- Running priority of the task
- Outstanding I/O count
- Status flags

The status flags use the same mnemonics as the SHOW TASK/FULL command. See Chapter 3 for definitions of the status flags.

8.5.1 Altering the Active Task Display from the Setup Page

To alter the Active Task Display, you press the CTRL-[sequence, which displays the setup page for the Active Task Display. The setup page documents and prompts you for commands that you use to alter the Active Task Display. You can enter multiple commands after each prompt by using commas as separators. The setup commands available for altering the Active Task Display are:

- OWNER=ttnn: where ttnn: is the terminal that issued the task
- PRIORITY=p where p is a task's running priority
- RATE=s where s is the replot rate in seconds
- TASK=taskname where taskname is the name of the task whose header you want to display

You can truncate these commands to their shortest unique forms. These setup parameters stay in effect until you alter them, even if you switch to another display page and back to the Active Task Display.

8.5.1.1 The OWNER Command - This command allows RMD to display only those tasks that have been issued by a particular terminal. The default is ALL, which displays tasks issued from all terminals.

8.5.1.2 The PRIORITY Command - The active task list may be too long to fit on one screen. You use the PRIORITY command to determine the highest priority tasks that you want to see. The default is 250, the highest possible priority.

8.5.1.3 The RATE Command - This command allows you to determine how often RMD replots the Active Task Display. The default replot rate is once per second.

8.5.1.4 The TASK Command - This command allows you to look at a specific task header. This command is an exception because it is the only setup command that switches display pages. There is no default for the TASK command. The Task Header Display is discussed in Section 8.6.

8.6 THE TASK HEADER DISPLAY

The Task Header Display shows you the task header of the task you specify. You access this display from another display page by pressing the T key (for Task Header). If no task is currently specified, RMD shows you the setup page first so that you can specify the task whose task header you want RMD to display.

The Task Header Display shows you the following information about the specified task:

- Name of the task
- Name of the partition in which the task runs
- Status flags, which have the same mnemonics as in the Active Task display
- Owner of the task by terminal number

- Outstanding I/O count
- Default priority
- Running priority
- Swapping priority
- Length in decimal words
- Contents of the six general purpose registers, the program counter, and the Processor Status Word
- Contents of the Directive Status Word ($DSW)
- Local event flags
- Logical unit number (LUN) assignments to a maximum of 26 LUNs

When RMD displays file names in the list of LUN assignments, the filename and directory displayed are the filename and directory of the file when it was created. If the file has been renamed, the RMD display may not reflect the current directory and/or filename.

8.6.1 Altering the Task Header Display from the Setup Page

To alter the Task Header Display, you press the CTRL-[sequence, which displays the setup page for the Task Header Display. The setup page documents and prompts you for commands that alter the Task Header Display. You can enter multiple commands after each prompt by using commas as separators. The setup commands available for altering the Task Header Display are:

- RATE=s where s is the replot rate in seconds
- TASK=taskname where taskname is the name of the task whose header you want to display

You can truncate these commands to their shortest unique forms. These setup parameters remain in effect until you alter them, even if you switch to another display page and back to the Task Header Display.

8.6.1.1 The RATE Command - This command allows you to determine how often RMD replots the Task Header Display. The default replot rate is once per second.

8.6.1.2 The TASK Command - This command changes the task header to be displayed. There is no default. The TASK command has the same function as the TASK command on the Active Task Display, except that here it does not switch display pages.

8.7 ERROR MESSAGES

RMD generates the following error messages:

RMD - Allocated screen buffer too small for this device

Explanation: RMD requires more internal memory to display the requested display on the type of terminal on which you are running RMD.

User Action: Call your DIGITAL Customer Support Center.

RMD - Illegal command - xxxxx

Explanation: You entered an illegal command xxxxx either on the command line or in response to the COMMAND> prompt on a setup page.

User Action: Enter the correct command as documented in this chapter.

RMD - Page does not exist

Explanation: You requested a display page from the command line that does not exist.

User Action: Enter the command line again specifying a correct display mnemonic.

RMD - Segment 'xxxxxx' not found

Explanation: The module xxxxxx was not found in the task image for RMD. This denotes an error in how RMD was task-built.

User Action: Call your DIGITAL Customer Support Center.

RMD - Terminal type not defined

Explanation: The operating system and RMD do not recognize your terminal type.

User Action: Check your terminal type using the command SHOW TERMINAL. If this setting is incorrect, use the SET TERMINAL command to correct your terminal type setting.

RMD - Terminal type not set

Explanation: You did not build RMD to display the requested display page on the type of terminal to which your terminal is set.

User Action: Determine your terminal type setting using the command SHOW TERMINAL. If this setting is incorrect, use the command SET TERMINAL to correct your terminal type setting.

RMD - Terminal type not yet supported

Explanation: RMD does not recognize your terminal type.

User Action: Determine your terminal type setting using the command SHOW TERMINAL. If this setting is incorrect, use the command SET TERMINAL to correct your terminal type setting.

CHAPTER 9

TASK/FILE PATCH PROGRAM (ZAP)

The Task/File Patch Program (ZAP) allows you to directly examine and modify task image and data files on a FILES-11 volume. Using ZAP, you can patch these files interactively without reassembling and rebuilding the task.

ZAP supports four types of task image files:

- Regular task image files, which include those mapped to resident and supervisor mode libraries
- Multiuser task image files
- I- and D-space (instruction and data space) tasks
- Resident libraries

NOTE

I- and D-space and supervisor mode libraries are not currently supported.

These types of task image files are discussed fully with the /List switch (refer to Section 9.1).

ZAP performs many of the functions performed by the RSX-11 on-line debugging utility, ODT. Thus, working knowledge of ODT is helpful in using ZAP. See the *IAS/RSX-11 ODT Reference Manual* for more information on ODT.

ZAP provides the following features:

- Operating modes that allow you to access specific words and bytes in a file, modify locations in a file, list the disk block and address boundaries for each overlay segment in a task image file on disk, and open a file for reading only.

- A set of internal registers that include eight Relocation Registers.

- Single-character commands that, with other command line elements, allow you to open and close locations in a file and to display and manipulate the values in those locations.

Although the ZAP program is relatively straightforward to use, patching locations in a task image file requires knowing how to use the map (or memory allocation file) generated by the Task Builder and the listings generated by the MACRO-11 assembler. These maps and listings provide information you need to access the locations whose contents you want to change. For information on Task Builder maps, see the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual*. For information on MACRO-11 listings, see the *PDP-11 MACRO-11 Language Reference Manual*.

9.1 ZAP OPERATING MODES AND SWITCHES

ZAP provides two addressing modes and two access modes. The addressing modes are task image mode and absolute mode. Task image mode is the default mode. The access modes are read/write mode and read-only mode. Read/write is the default mode. Either addressing mode can be used with either access mode. The modes and their associated switches are as follows:

- Task image mode is the default addressing mode for ZAP. In this mode, addresses in ZAP command lines refer to addresses in the task image file as they are shown in the Task Builder map for the file. Refer to Section 9.1.1.1 for more information on using task image mode.

- In absolute mode, specified with the /AB switch, ZAP processes the addresses you enter in ZAP command lines as absolute byte addresses within the file. You must use absolute mode for any files that are not task images. Refer to Section 9.2.1 for more information on using absolute mode.

- Read/write mode is the default access mode for ZAP. In this mode, ZAP opens a file for reading and/or modification.

- In read-only mode, specified with the /RO switch, ZAP opens a file for reading but not modification. That is, you can execute ZAP functions that change the contents of locations, but these changes are not actually made to the file. When ZAP exits, the original values in the file are still there.

9.1.1 The List Switch (/LI)

When using ZAP in task image mode (but not absolute), you can also specify the /List switch (/LI). The /List switch displays the overlay segment table for the on-disk task image file with which you are working. The table lists the starting disk block and address boundaries for each overlay segment in the file. The segment table lists are in a different format for each type of task image file. The table and Task Builder map allow you to locate the task segments being changed.

The /LI switch displays the overlay segment boundaries in the following format:

```
ssssss:  aaaaaa-bbbbbb [nnnnnn] [dddddd]
```

ssssss:

The starting block in octal.

aaaaaa

The lower address boundary in octal.

bbbbbb

The upper address boundary in octal.

nnnnnn

The segment name that appears for I- and D-space tasks; manually loaded overlays ($LOAD); memory-resident overlays; tasks that link to the library with memory-resident overlays; or for any combination of the previous conditions.

dddddd

The description of the segment type string which appears next to the segment name in the segment table.

The following sections describe the /List switch formats for the different kinds of task image files. Section 9.6.6 gives examples of the segment table lists.

9.1.1.1 The /LI Switch and Regular Task Image Files - For regular task image files, including those mapped to resident and supervisor mode libraries, the /LI switch displays the task's overlay segments in the order of their location in the file. Each segment begins on an even block boundary.

9.1.1.2 The /LI Switch and Multiuser Task Image Files - For multiuser tasks, the /LI switch lists the starting disk block number and address boundaries of each segment. In addition, the address boundaries of the shared read-only segment are listed. The block number that is used to reference the multiuser segment is the same as the root segment. The multiuser segment is an extension of the root segment. The segment list disk block numbers have a corresponding entry in the Task Builder map.

See the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual* for more information on multiuser tasks.

9.1.1.3 The /LI Switch and Resident Libraries - For resident libraries, the /LI switch displays each of the task's segments as beginning on a new block boundary. However, the segments may not actually begin on even boundaries because of compression by the Task Builder. Resident libraries can be overlaid, but each overlay segment must also be resident in memory.

To avoid the possibility that two or more segments in a single block could have the same virtual address, ZAP treats the resident library in the same way that the Task Builder does. The Task Builder builds the library with each segment beginning on an even block boundary, but then compresses the segments in the task file itself. The Task Builder map is generated before the segments are compressed, so the boundaries given in the map do not necessarily correspond to the actual location of the segments.

The disk block boundaries given in the Task Builder map file are the ones that ZAP uses to address locations in the resident library and that the /LI switch displays in its segment table. They do not use the actual starting blocks of the segments. (You should be aware of this when you are working with resident libraries in absolute mode. However, also remember that you cannot use the /LI switch in absolute mode.)

You should also note that ZAP cannot know the physical starting addresses for the segments of an overlaid resident library because its overlay structure is stored in the symbol definition (.STB) file, not with the task image file itself. For ZAP, each segment's starting address is 000000.

See the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual* for more information on resident libraries.

9.1.1.4 The /LI Switch and I- and D-Space Tasks - For I- and D-space tasks, the /LI switch lists the starting block number and the address boundaries of each segment. The I-space or D-space segment may be suppressed in the listing when, for example, a segment with only I-space code does not include a listing for a D-space segment. If the segments are not suppressed, the segment list disk block numbers have a corresponding entry in the Task Builder map.

9.2 ADDRESSING LOCATIONS IN FILES

To address locations in a file, ZAP provides two addressing modes (task image and absolute, described above) and a set of internal registers, which includes eight Relocation Registers. This section first introduces the concept of relocation biases and the use of the Relocation Registers, then explains how to use the addressing modes.

9.2.1 Relocation Biases

When MACRO-11 generates a relocatable object module, the base address of each program section of the module is 000000. In the assembler listing, all locations in the program section are shown relative to this base address.

The Task Builder links program sections to other program sections by mapping the relative addresses applied by the assembler to the physical addresses in memory (for unmapped systems) or to virtual addresses (for mapped systems).

Many values within the resulting task image are biased by a constant whose value is the absolute base address of the program section after the section has been relocated. This bias is called the relocation bias for the program section.

ZAP's eight Relocation Registers, 0R through 7R, are generally set to the relocation biases of the program sections to be examined. This allows you to refer to a location in a module by the same relative address that appears in the MACRO-11 listing. The addressing modes help you calculate the relocation biases.

9.2.2 ZAP Addressing Modes

As explained in Section 9.1, ZAP's two modes for addressing locations in a file are task image mode and absolute mode. Task

image mode is the default mode for ZAP. The next two sections explain how to use these modes.

The following examples show excerpts from a MACRO-11 listing of the module MYFILE and a Task Builder map. These excerpts and the accompanying text show how to use ZAP in task image mode. The following lines represent assembled instructions from a MACRO-11 source listing:

```
71 000574    032767  000000G 000000G BIT  #FE.MUP,$FMASK
72 000602    001002                  BNE  2$
73 000604    000167  000406          JMP  30$
74 000610    016700  000000G    2$:  MOV  $TKTCB,R0
75 000614    016000  000000G         MOV  T.UCB(R0),R0
76 000620    010067  177534          MOV  R0,UCB
77 000624    032760  000000G 000000G BIT  #U2.HLD,U.CW2(R0)
```

The following lines from a Task Builder map give the information you need to address locations in the task image file as they appear in the above MACRO-11 listing:

```
R/W MEM   LIMITS: 120000 123023 003024 01556.
DISK BLK  LIMITS: 000002 000005 000004 00004.

MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE IDENT     FILE

. BLK.:(RW,I,LCL,REL,CON) 120232 002546 01382.
                          120232 002244 01188. MYFILE 01    MCR.OLB;1
                          122476 000064 00052. FMTDV  01    MCR.OLB;1
$$RESL:(RW,I,LCL,REL,CON)  123000 000024 00020.
```

Using information in the Task Builder map, you can determine the block number and byte offset for the beginning of the file you want to change. The disk-block-limits line lists block 2 as the block where the program code begins. The synopsis lists byte offset 120232 as the beginning of the file MYFILE. To address location 574 in the MACRO-11 listing in task image mode, specify the following command line:

```
2:120232+574/<RET>
```

ZAP responds by opening the location and displaying its contents:

```
002:121026/ 032767
```

9.2.2.1 Using the Task Image Addressing Mode - In task image mode, ZAP allows you to address locations in a task image file by using the addresses the MACRO-11 assembler displays in its listing and the starting block number and byte offset listed in the Task Builder map. Unlike absolute mode, task image mode is useful for working with locations in an overlaid file because the Task Builder and ZAP perform the calculations necessary to relate the file's disk structure to its run-time memory structure.

9.2.2.2 Using the Absolute Addressing Mode - In absolute mode, ZAP processes the addresses you enter in the command lines as absolute byte addresses within the file. To use ZAP in absolute mode, invoke ZAP and enter the /AB switch with the file specification.

ZAP interprets the first address in the file you are changing as virtual block 1, location 000000. All other addresses you enter are interpreted using this address as the base location. Absolute mode allows you to access all the bytes in a data or task image file as well as the label and header blocks of a task image file on disk. However, to modify a disk task image, you must know the disk layout of the task image. Generally, absolute mode is practical only for data files or for task image files that are not overlaid.

9.3 INVOKING AND TERMINATING ZAP

To invoke ZAP type:

```
$ RUN $ZAP
```

ZAP will prompt you:

```
ZAP>
```

You cannot enter a file specification on the same line that you use to invoke ZAP unless the file is an indirect command file. Refer to Section 9.3. When ZAP prompts you, enter the file specification for the file you want to change. You enter the file specification in the format:

```
dev:[directory]filename.filetype;version[/sw...]
```

The default file type is .TSK. After you enter the file specification, ZAP prompts with an underscore (_).

You terminate ZAP by entering the X command (explained in Section 9.6.1). This command exits you from ZAP and returns control to your DCL.

9.3.1 Using Indirect Command Files with ZAP

An indirect command file contains the specification for the file you want to work with and the appropriate ZAP commands. You can specify the indirect command file in the same command line in which you invoke ZAP.

The following sample indirect command file (called CHANGE.CMD) contains ZAP commands. The commands will change the default priority of the despooler from 70 to 80 (120 octal). The V command (explained in Section 9.7.5) is used to verify that 70 (106 octal) is what is actually in the location to be changed. The command file has the following ZAP commands:

```
LPP.TSK/AB
0:346/
106V
120
X
```

To use the indirect command file type the following:

```
ZAP> @CHANGE
```

In this command, ZAP executes the commands in the file named CHANGE.CMD.

The commands being used first open the task image file (LPP.TSK) in absolute mode (/AB). The next two commands open the desired location (byte 346 in block 0) and verify its contents (106). The next command changes the contents to 120, which will be the new default priority for the despooler. The X command exits you from ZAP and returns control to DCL.

9.4 THE ZAP COMMAND LINE AND COMMAND LINE ELEMENTS

ZAP commands perform functions that allow you to examine and modify the contents of locations in a file. Command lines comprise combinations of the following elements:

- Commands
- Internal registers
- Arithmetic operators
- Command line element separators
- The current location symbol
- Location-specifier formats

The command elements can be combined to perform multiple functions. The function of a given command line depends not only on which elements you use, but also on the position of one element in relation to the next.

The following sections describe the ZAP command line elements. Sections 9.5 and 9.6 describe how to combine the command line elements to execute ZAP functions.

9.4.1 ZAP Commands

There are three types of ZAP commands:

- Open/close location commands
- General purpose commands
- <RETURN> key

The following sections describe each type of command.

9.4.1.1 Open/Close Location Commands - Open/close location commands are nonalphanumeric ASCII characters that direct ZAP to perform a sequence of functions. Open/close commands specify two general sequences of operations:

- Open a location, display its contents, and store the contents in the Quantity Register (see Section 9.6.2).
- Close the location after (optionally) modifying it and open another location as specified by the command.

Section 9.6 describes the format for specifying open/close location commands.

9.4.1.2 General Purpose Commands - ZAP provides six single-character, general purpose commands. You use these commands for calculating displacements, verifying location contents, and exiting from ZAP. You can enter some of the commands on the command line with no other parameters. Section 9.7 describes the format for specifying these commands.

9.4.1.3 <RETURN> Key - Unless there is another value or command on the line, the <RETURN> key closes the current location as modified and opens the next sequential location. ZAP commands take effect only after you press the <RETURN> key.

9.4.2 ZAP Internal Registers

ZAP internal registers are fixed storage locations that ZAP uses as registers. These registers contain values set by both you and ZAP. ZAP provides the following internal registers:

- Relocation Registers 0 through 7 (0R through 7R). These registers provide a means for indexing into a program section to change the contents of locations in the program section. You load the registers with the base address of the program section that has been relocated by the Task Builder.

- The Constant Register (C). You set this register to contain a 16-bit value, which you can specify as an expression in the ZAP command line.

- The Format Register (F). This register controls the format of the displayed address. If the value of the F Register is 0 (the initial value), ZAP displays addresses relative to the largest value of any Relocation Register whose value is less than or equal to the address to be displayed. If the value of the Format Register is not 0, ZAP displays addresses in absolute byte format.

- The Quantity Register (Q). ZAP sets the value in the register to be the last value displayed at your terminal.

To access the contents of a register, specify a dollar sign ($) preceding the register name (in this case, C) in the command line. For example:

```
_$C/<RET>
$C/ 000000
_
```

This command line directs ZAP to display the contents of the Constant Register. The slash (/) is an open command described in Table 9-3.

9.4.3 ZAP Arithmetic Operators

Operators are single-character command line elements that define an arithmetic operation. Generally, ZAP evaluates these expressions as addresses. Table 9-1 describes the operators.

You use the operators in expressions in command lines. For example, rather than manually adding all the displacements listed in the Task Builder map, you can specify a location using the following notation:

```
_2:120000+170/<RET>
```

This method for calculating such a displacement is faster and more accurate than doing it manually.

Table 9-1: ZAP Arithmetic Operators

Operator	Function
+	The plus sign adds a value to another value. Use it in an expression that ZAP then evaluates to be a command line element.
-	The minus sign subtracts a value from another value. Use it an expression that ZAP then evaluates to be a command line element.
*	The asterisk multiplies a value by 50 (octal) and adds it to another value. Use it to form a Radix-50 string.

The following example shows how to use the asterisk (*) to form Radix-50 strings. Section 9.5.4 explains the use of the colon and comma; the percent sign is an open command described in Table 9-3.

```
_0,40/<RET>
002:0,000040/  000001
_1*33<RET>
_/<RET>
002:0,000040/  000103
_%<RET>
002:0,000040%  A$
```

In this example, the first command opens the locations that is 40 bytes offset from the location address contained in Relocation Register 0 and displays in octal format the contents of the new location. The location contains the value 000001. The second command converts 000001 to Radix-50 and adds 33 to the Radix-50 value. The slash (/) command again displays in octal the value contained in the offset location. The value is now 103. The percent sign (%) command displays 103 in Radix-50 format.

9.4.4 ZAP Command Line Element Separators

ZAP provides separators to delimit one command line element from another. Different separators are required depending on the type of ZAP command being executed. See Table 9-2.

Table 9-2: ZAP Command Line Element Separators

Separator	Function
,	The comma separates a Relocation Register specification from another command line element.
;	The semicolon separates an address from an internal register specification. Used in expressions that set values for Relocation Registers.
:	The colon separates a block number base value from a byte offset into the block. Used in most of the references to locations in a file.

9.4.5 ZAP Command Line Location-Specifier Formats

ZAP has four formats for specifying locations in a command line.

Each provides a means of indexing into a file.

The formats are:

- Current location symbol
- Byte offset
- Block number:byte offset
- Relocation register, byte offset

9.4.5.1 The Current Location Symbol - In command line expressions that ZAP evaluates as addresses, a period (.) represents the last open location.

9.4.5.2 Byte Offset Format - You specify the byte offset format as follows:

```
location
```

If you are using ZAP in absolute mode, ZAP interprets this specification as a byte offset from block 1, location 000000. This format is generally useful only when you are using absolute mode.

The following ZAP command line opens absolute location 664 and displays its contents in octal format:

```
_664/<RET>
```

9.4.5.3 Block Number: Byte Offset Format - This format allows you to specify a byte offset from a specific block in the file. Specify the format as follows:

```
blocknum:byteoffset
```

You can use this format for addressing locations whether or not you enter the /AB switch with the file specification.

In task image mode, ZAP allows you to enter the block number and byte offset displayed in the Task Builder map. The map gives information on the overlay segments in a task image file. Refer to Section 9.3 for more information.

9.4.5.4 Relocation Register, Byte Offset Format - This format allows you to load a Relocation Register with the address of a location. The address is then used as a relocation bias. You specify this format for addressing locations in a task image file as follows:

```
relocreg,byteoffset
```

Specify relocreg in the form nR, where n is the number of the Relocation Register. You can then address byte offsets from the address loaded in the Relocation Register. For example:

```
_2:001254;3R<RET>
_3,64/<RET>
002:3,000064/ 037334
```

The first command loads the address 001254 into Relocation Register 3, then the second command opens the location that is 64 bytes offset from block 2, location 001254. The contents of that location are 037334.

9.5 USING ZAP OPEN AND CLOSE COMMANDS

This section gives examples of how to use the ZAP open and close commands. These commands allow you to open locations in a file, modify those locations, and close the locations.

Table 9-3 summarizes the open and close commands

Table 9-3: ZAP Open and Close Commands

Command	Name	Description
/	Slash	Opens a location, displays its contents in octal, and stores the contents of the location in the Quantity Register (Q). If the location is odd, it is opened as a byte.
"	Quotation mark	Opens a location, displays the contents of the location as two ASCII characters, and stores the contents of the location in the Quantity Register (Q).

Command	Name	Description
%	Percent sign	Opens a location, displays the contents of the location in Radix-50 format, and stores the contents of the location in the Quantity Register (Q).
\	Backslash	Opens a location as a byte, displays the contents of the location in octal, and stores the contents of the location in the Quantity Register (Q).
'	Apostrophe	Opens a location, displays the contents as one ASCII character, and stores the contents of the location in the Quantity Register (Q).
<RET>	RETURN key	Closes the current location as modified and opens the next sequential location if no other values or commands are on the command line. ZAP commands take effect only after you press the RETURN key.
^	Circumflex or Up-arrow	Closes the currently open location as modified and opens the preceding location.
_	Underscore or Back-arrow	Closes the currently open location as modified, uses the contents of the location as an offset from the current location, and opens the new location.
@	At sign	Closes the currently open location as modified, uses the contents of the location as an absolute address, and opens that location.

Command	Name	Description
>	Right angle bracket	Closes the currently open location as modified, interprets the low-order byte of the contents of the location as the relative branch offset, and opens the target location of the branch.
<	Left angle bracket	Closes the currently open location as modified, returns to the location from which the last series of underscore (__), at sign (@), and/or right angle bracket (>) commands began, and opens the next sequential location.

9.5.1 Opening Locations in a File

Use any of the ZAP open commands -- slash (/), quotation mark ("), percent sign (%), backslash (\), or apostrophe (') -- to open a location in a file. The format ZAP uses to display the contents of the open location depends on which operator you use.

Once you open a location in a given format, ZAP displays in that format any other locations you open. For example, if you enter the percent sign (%) command, the contents of the open location are displayed in Radix-50 format. If you continually press the <RETURN> key, consecutive locations are displayed in Radix-50 format until you change the format by entering a different special-character open command.

9.5.2 Changing the Contents of a Location

When you open a location with a special-character open command, you can change the contents of that location by entering the new value and pressing the <RETURN> key. The following example is a sequence of commands and ZAP responses that show how to open a location, change the value of the location, and close the location.

```
_/<RET>
002:120000/ 000000
_44444<RET>
_/<RET>
002:120000/ 44444
```

The first command (/) displays in octal format the contents (000000) of the current location. The contents are changed by entering the value 44444 and then the location is closed as modified by pressing the <RETURN> key. The slash (/) and <RETURN> key display the new contents of the location (last line of example).

9.5.3 Closing Locations in a File

ZAP uses the <RETURN> key and other special-character commands for closing a location in a file. The close commands perform three functions:

- Close the current location
- Direct ZAP to another location (such as the preceding location or a location referred to by the current location)
- Open the new location

The following sections give examples of how each command works.

9.5.3.1 Closing a Location and Opening the Preceding Location - Use the circumflex (^) or up-arrow (#) command (depending on the type of terminal you are using) to close the current location, to direct ZAP to the preceding location, and to open that location. The following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/<RET>
002:120100/ 000000
_<RET>
002:120102/ 000111
_<RET>
002:120104/ 000222
_<RET>
002:120106/ 000333
_^<RET>
002:120104/ 000222
```

The <RETURN> key closes the first three open locations and then opens the next location. The circumflex command closes location 120106 and directs ZAP to open the preceding location, 120104.

9.5.3.2 Closing a Location and Opening an Offset Location - Use the underscore (_) or back-arrow (#) command to close the current location, to direct ZAP to use the contents of the current location as an offset from the current location, and to open the new location. The following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/<RET>
002:120100/ 000000
_<RET>
002:120102/ 111111
_<RET>
002:120104/ 222222
_<RET>
002:120106/ 000022
__<RET>
002:120132/ 123456
```

The <RETURN> key closes the first three open locations. The underscore command closes location 120106, directs ZAP to use the contents (22) of the current location as the offset from the current location (120110), and then opens that offset location (120132).

9.5.3.3 Closing a Location and Opening an Absolute Location - Use the at sign (@) command to close the current location, to direct ZAP to use the contents of the just-closed location as the absolute address of a location, and to open that location.

The following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/<RET>
002:120100/ 000000
_<RET>
002:120102/ 111111
_<RET>
002:120104/ 120114
_@<RET>
002:120114/ 114114
```

The <RETURN> key closes the first three open locations. The at sign command closes 120104, directs ZAP to use the contents

(120114) of that location as the absolute address of the next location to open, and then opens that location.

9.5.3.4 Closing a Location and Opening a Branch Target Location - Use the right angle bracket (>) command to close the current location, to direct ZAP to use the low-order byte of the contents of the just-closed location as a branch offset for the address of the next location, and then to open that location. The following sequence of ZAP commands and responses shows how this command works:

```
_2:120100/<RET>
002:120100/ 005000
_<RET>
002:120102/ 005301
_<RET>
002:120104/ 001020
_><RET>
002:120146/ 052712
```

The <RETURN> key closes the first three open locations. The right angle bracket command closes location 120104, directs ZAP to use the low-order byte (020) of its contents as the branch offset for the address of the next location (120146), and then opens that location.

9.5.3.5 Closing a Location and Opening a Previous Location - Use the left angle bracket (<) command to close the current location; to direct ZAP to the location where the current series of underscore (_), at sign (@), and/or right angle bracket (>) 'HEADER'began; and then to open that location. The following sequence of ZAP commands and responses shows how this command works:

```
_1202;0R<RET>
_0,10/<RET>
002:0,000010/ 005212
__<RET>
002:0,005224/ 001020
_><RET>
002:0,005266/ 000000
_@<RET>
002:0,000000/ 000000
_<<RET>
002:0,000012/ 000430
```

The underscore command directs ZAP to location 005224. The right

angle bracket command directs ZAP to location 005266, and the at sign command directs ZAP to location 000000. The left angle bracket command then directs ZAP to location 000012, which is the next sequential address after the location where the sequence of commands began.

9.6 USING ZAP GENERAL PURPOSE COMMANDS

This section explains the functions of ZAP general purpose commands and shows the formats for specifying them. Table 9-4 describes the commands.

Table 9-4: ZAP General Purpose Commands

Command	Function
X	Exits from ZAP; returns control to DCL.
K	Calculates the offset in bytes between an address and the value contained in a Relocation Register, displays the offset value, and stores it in the Quantity Register (Q).
O	Displays the jump and branch displacements from the current location to a target location.
=	Displays in octal the value of the expression to the left of the equal sign.
V	Verifies the contents of the current location.
R	Sets the value of a Relocation Register.

9.6.1 The X Command

Use the X command to exit from ZAP and then return control to DCL.

Specify the X command in the format:

```
_X
```

9.6.2 The K Command

Use the K command to calculate the offset in bytes between an address and the value contained in a Relocation Register, to display the offset value, and to store it in the Quantity Register (Q).

You can enter the K command in the following formats:

K — Calculates the offset in bytes between the address of the currently open location and the value of the Relocation Register whose contents are equal to or closest to (but less than) the value of that address.

nK — Calculates the offset in bytes between the currently open location and Relocation Register n.

a;nK — Calculates the offset in bytes between address a and Relocation Register n.

ZAP responds to the K command by displaying the Relocation Register it used and the offset value it calculated in the format:

```
=reg,offset
```

The following example shows how to use the K command:

```
_2:1172;0R<RET>
_2:1232;1R<RET>
_2:1202/<RET>
002:000020/ 000111
_K<RET>
=0,000010
_0,100;1K<RET>
=1,000040
```

The first command sets the value of Relocation Register 0 to 001172. The second command sets the value of Relocation Register 1 to 001232. The slash command displays in octal format the contents of location 001202 (000111). The K command calculates the physical distance (offset) between the address of the currently open location (001202) and the value of the Relocation Register whose contents are equal to or closest to (but less than) the value of the address. ZAP then displays the number of the Relocation Register it used (0) and the offset (00010=001202-001172).

The last command adds 100 to the address in Relocation Register 0 (001172) and then calculates the offset between the new address 0 (001272) and the contents of Relocation Register 1 (001232). ZAP then displays the number of the specified Relocation Register (1) and the offset (000040=001272-001232).

9.6.3 The O Command

Use the O command to display the jump and branch displacements from the current location to a target location. A jump displacement is the offset between the open location and the target location. The jump displacement is used in the second word of a jump instruction if the instruction uses relative addressing. A branch displacement is the low-order byte of a branch instruction which, when executed, branches to the target location.

You can enter the O command in the following formats:

aO — Displays the jump and branch displacements from the current location to the target of the branch (a).

a;rO — Displays the jump and branch displacements from location a to target location r.

The following example shows how to use the O command:

```
_0,4534/<RET>
0,4534/ 1234
_4566O<RET>
_000030> 000014
_4534;4566O<RET>
_000030> 000014
```

The first number (000030) is the jump displacement; the second number (000014) is the branch displacement.

9.6.4 The Equal Sign (=) Command

Use the equal sign command (=) to display (in octal) the value of the expression to the left of the equal sign.

Specify the equal sign command in the format:

```
expression=
```

The following example shows how to use the equal sign command

(note that 177777 equals -1):

```
_2:30/<RET>
002:000030/ 000000
_.+177756=<RET>
000006
```

The first command displays in octal format the contents of location 000030, which are 000000. The next command adds 177756 to the address of the currently open location (000030). ZAP then displays the value of the specified expression (6=30+177756 or 6=30-22).

9.6.5 The V Command

Use the V command to verify that a location contains a specified value.

Specify the V command in the format:

```
contentsV
```

You use the V command to ensure that, before you have ZAP change them, the contents being changed are what they should be. The V command is mainly useful in indirect command files because ZAP issues an error message and exits if the contents do not match. That way, the contents are not changed incorrectly.

The following example shows how to use the V command; if you were using an indirect command file, you would include this sequence of ZAP commands in it.

```
0,1200/
6V
10
```

ZAP opens the location that is 1200 offset from the value of Relocation Register 0 and ensures that the value contained at the location is 6. If so, ZAP changes the 6 to 10. If the value is not 6, ZAP exits.

9.6.6 The R Command

Use the R command to specify the value for a Relocation Register. As explained in Sections 9.3 and 9.5.2, ZAP uses these registers to index into a program section so that you can change the contents of locations in the program section.

Specify the R command in the format:

```
_contents;nR
```

The variable n is the number of the Relocation Register (0 through 7).

For example:

```
_$3R/<RET>
$3R/ 177777
_125670;3R<RET>
_$3R/<RET>
$3R/ 125670
```

The first command accesses the contents of Relocation Register 3, which ZAP displays in octal format as specified by the slash. The contents of the register are 177777. The next command changes the contents of the register to 125670. The last command again displays the contents of the register, which have been changed correctly.

9.7 EXAMPLES

This section gives examples of ZAP usage. The examples show the /LI switch segment table format and how you would use some of the ZAP commands.

Some of the ZAP examples in this section are based on information contained in the following excerpts from a sample Task Builder memory allocation map and from the program code for some of the modules in the task. Each example follows the section of program code associated with it.

Excerpts from Task Builder map:

```
MAINMEO.TSK;1   Memory allocation map  TKB M40.10      Page 1
                      14-MAR-83   16:01

Task    name    : ...MEO
Partition name : GEN
Identification : M00
Task  UIC       : [200,200]
Stack    limits: 000300 001277 001000 00512.
PRG xfr address: 020520
Task attributes: ID
Total address windows: 2.
Task  image  size  : 9184. words, I-Space
```

```
                    3520. words, D-Space
Task Address limits: 000000 043647 I-Space
                     000000 015507 D-Space
R-W disk blk limits: 000002 000102 000101 00065.

MAINMEO.TSK;1 Overlay description:

Base    Top        Length
----    ---        ------
000000  023135  023136  09822. I    MAINO
000000  014123  014124  06228. D

022140  043645  021506  09030. I    INPUT
014124  015507  001364  00756. D

022140  022307  000150  00104. I    CALC
014124  014167  000044  00036. D

022310  022437  000130  00088. I    AADD
014170  014173  000004  00004. D

022310  022437  000130  00088. I    SUBB
014170  014173  000004  00004. D

022310  022437  000130  00088. I    MULL
014170  014173  000004  00004. D

022310  022441  000132  00090. I    DIVV
014170  014173  000004  00004. D

022140  023725  001566  00886. I    OUTPUT
014124  014251  000126  00086. D

MAINMEO.TSK;1   Memory allocation map  TKB M40.10     Page 2
MAINO                14-MAR-83   16:01

*** Root segment: MAINO

R/W mem  limits: 000000 023135 023136 09822. I-Space
                 000000 014123 014124 06228. D-Space

Disk blk limits: 000002 000024 000023 00019. I-Space
                 000025 000041 000015 00013. D-Space

Memory allocation synopsis:

Section                                          Title  Ident
File
-------                                          -----  -----
----
. BLK.:(RW,I,LCL,REL,CON)     000300 000216 00142.
```

```
                                  000300 000216 00142. CBTA   04.3
SYSLIB.OLB;7
                                       .
                                       .
                                       .


MAINMEO.TSK;1   Memory allocation map  TKB M40.10       Page 4
INPUT                 14-MAR-83   16:01

*** Segment: INPUT

R/W mem  limits: 022140 043645 021506 09030. I-Space
                 014124 015507 001364 00756. D-Space

Disk blk limits: 000042 000063 000022 00018. I-Space
                 000064 000065 000002 00002. D-Space
                   .
                   .
                   .
                                                        Page 5
*** Segment: CALC

R/W mem  limits: 022140 022307 000150 00104.  I-Space
                 014124 014167 000044 00036.  D-Space

Disk blk limits: 000066 000066 000001 00001.  I-Space
                 000067 000067 000001 00001.  D-Space
                   .
                   .
                   .
                                                        Page 7
*** Segment: AADD

R/W mem  limits: 022310 022437 000130 00088.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000070 000070 000001 00001.  I-Space
                 000071 000071 000001 00001.  D-Space
                   .
                   .
                   .
                                                        Page 8
*** Segment: SUBB

R/W mem  limits: 022310 022437 000130 00088.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000072 000072 000001 00001.  I-Space
                 000073 000073 000001 00001.  D-Space
                   .
```

```
                    .
                    .
                                                          Page 9
*** Segment: MULL

R/W mem  limits: 022310 022437 000130 00088.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000074 000074 000001 00001.  I-Space
                 000075 000075 000001 00001.  D-Space
                    .
                    .
                    .

MAINMEO.TSK;1   Memory allocation map  TKB M40.10      Page 10
DIVV                 14-MAR-83   16:01

*** Segment: DIVV

R/W mem  limits: 022310 022441 000132 00090.  I-Space
                 014170 014173 000004 00004.  D-Space

Disk blk limits: 000076 000076 000001 00001.  I-Space
                 000077 000077 000001 00001.  D-Space
                    .
                    .
                    .
                                                          Page 11
*** Segment: OUTPUT

R/W mem  limits: 022140 023725 001566 00886.  I-Space
                 014124 014251 000126 00086.  D-Space

Disk blk limits: 000100 000101 000002 00002.  I-Space
                 000102 000102 000001 00001.  D-Space

Memory allocation synopsis:

Section                                          Title  Ident
File
-------                                          -----  -----
----
. BLK.:(RW,I,LCL,REL,CON)      022140 000374 00252.
                               022140 000042 00034. SAVAL  00
SYSLIB.OLB;7
                               022202 000074 00060. CATB   03
SYSLIB.OLB;7
                               022276 000126 00086. CDDMG  00
SYSLIB.OLB;7
                               022424 000110 00072. C5TA   02
SYSLIB.OLB;7
```

Example 1:

In this example, the segment table for task MAINMEO is requested.

Note that the segment table corresponds exactly to the overlay description list given in the Task Builder map. The sequence of ZAP commands is as follows:

NOTE

This example was taken from a PDP-11 running RSX-11M-PLUS. The example serves to show users how to use ZAP. Note, however, that I and D space capability is not currently supported.

```
ZAP>MAINMEO/LI
ZAP Version V02.01 COPYRIGHT (c) DIGITAL EQUIPMENT
CORPORATION 1983
Segment Table
000002: 000000-022137 MAINO    I-space root
000025: 000000-014123 MAINO    D-space root
000042: 022140-043647 INPUT    I- and
000064: 014124-015507 INPUT    D-space
000066: 022140-022307 CALC     I- and
000067: 014124-014167 CALC     D-space
000070: 022310-022347 AADD     I- and
000071: 014170-014173 AADD     D-space
000072: 022310-022437 SUBB     I- and
000073: 014170-014173 SUBB     D-space
000074: 022310-022437 MULL     I- and
000075: 014170-014173 MULL     D-space
000076: 022310-022443 DIVV     I- and
000077: 014170-014173 DIVV     D-space
000100: 022140-023727 OUTPUT   I- and
000102: 014124-014253 OUTPUT   D-space
```

In Example 1, the first command line invokes ZAP and the second command line requests the segment table for the task MAINMEO. The /List switch directs ZAP to give the starting disk block for the root segment of the task (in this example, MAIN0) and for each segment overlaid on the root of the task. The /List switch also lists the base and top addresses, plus the segment text string for each segment.

Because this is an I- and D-space overlaid task, there is an I-space root segment and a D-space root segment and each is a part of the root segment of the task MAIN0. In this example, the I-space root segment begins at disk block 2. The addresses for the I-space root segment range from 000000 to 022137. The next line of numbers is for the D-space root segment which begins at disk block 25. The addresses for the D-space root segment range

from 000000 to 014123. The next line of numbers is for the segment INPUT. That I-space segment begins at disk block 42 and the D-space segment begins at disk block 64. The I-space addresses range from 022140 to 043647 and the D-space addresses range from 014124 to 015507. The table continues for the remaining overlaid segments in the task MAINMEO.

Example 2:

In this example, the contents of a location in another task module (TEST.MAC) are being changed. The following excerpt from the module shows the associated code.

```
TEST - TEST MACRO FILE MACRO M1113  18-MAR-81 07:48  PAGE 8-1
1269                    ; THIS IS A PART OF THE MODULE
1270                    ;           TEST.MAC
1271                    ; WHICH IS IN THE ROOT SEGMENT:
1272                    ;            TEST
1273                    ;
1274 010132  010146                    MOV     R1,-(SP)
1275 010134  012704    041114          MOV     #"LB,R4
1276 010140  005003                    CLR     R3
1277 010142                            CALL    $FNDUB
1278 010146  010146                    MOV     R1,-(SP)
1279 010150  012704    050123          MOV     #"SP,R4
```

The sequence of ZAP commands is:

```
_42:121244;0R<RET>
_0,10136"<RET>
042:131402" LB
_'<RET>
042:131402' L
_'<RET>
042:131403' B
_120<RET>
_0,10136"<RET>
042:131402" LP
```

The first command line loads the starting address of TEST.MAC (121244 in disk block 42) into Relocation Register 0. The second command line displays as an ASCII word the contents of location 10136 of the module. The contents are LB. The first apostrophe command (') displays the first byte of the word (L) and the second command displays the second byte (B). The following command line changes the contents of the second byte to 120, which is the ASCII code for the letter P. The last command displays the new contents of location 10136, which are now LP.

Example 3:

In this example, the contents of a location are also being changed. This time, the location is in the module TSTVB1.MAC. The following excerpt is the associated code.

```
TSTVB1 - TSTVB1 MACRO FILE  MACRO M1113  18-MAR-81 07:52  PAGE 4-2

153                                     ;   PART OF MODULE TSTVB1.MAC
154                                     ; WHICH IS ALSO IN THE ROOT SEGMENT:
155                                     ;           TEST
156                                     ;
157  000334  005702                            TST     R2
158  000336  001404                            BEQ     70$
159  000340  052767  000060  172516            BIS     #60,SR3
160  000346  000403                            BR      75$
```

The sequence of ZAP commands is:

```
_42:133470;1R<RET>
_1,342/<RET>
042:134032/ 000060
_100<RET>
_1,342/<RET>
042:134032/ 000100
```

The first command line loads the starting address of TSTVB1.MAC (133470 in disk block 42) into Relocation Register 1. The second command line displays in octal the contents of location 342 in the module. The third command line changes the contents of this location from 60 to 100. The last command line displays the new contents (again in octal).

Example 4:

In this example, the operation code (op code) for one of the instructions in another module is being changed. The module is TSTCM.MAC, and the following excerpt is the associated code.

```
TSTCM - TSTCM MACRO FILE    MACRO M1113  18-MAR-81 07:47  PAGE 3-7

402      ; PART OF THE MODULE TSTCM.MAC
403      ; WHICH IS IN THE SEGMENT: TSTCM
404      ;
405  001272  073127  177766            ASHC    #-10.,R1
406  001276  010037  00000G          MOV     R0,@#KISAR6
407  001302  062701  140002            ADD     #140000+2,R1
```

The sequence of ZAP commands is:

```
_113:154530;R2<RET>
_2,1302/<RET>
113:156032/ 062701
_162701<RET>
_2,1302/<RET>
113:156032/ 162701
_X
```

The first command line loads the starting address of TSTCM.MAC (154530 in disk block 113) into Relocation Register 2. The second command line displays in octal the current instruction contained in location 1302. The instruction includes the op code 06 for the ADD operation. The third command line changes the op code to 16, which signifies the SUBTRACT operation. The fourth command line displays the new contents of the location and the X command ends the ZAP session.

9.8 ZAP ERROR MESSAGES

This section lists the messages generated by ZAP, explains the condition that causes each message, and suggests a response to the message.

ZAP -- ADDRESS NOT WITHIN SEGMENT

Explanation: The address specified was not within the overlay segment specified.

User Action: Reenter the command line, specifying the correct address or overlay segment number.

ZAP -- CANNOT BE USED IN BYTE MODE

Explanation: The at sign (@), underscore (_), and right angle bracket (>) commands cannot be used when a location is opened as a byte.

User Action: If the location is an even address, open the location as a word.

ZAP -- ERROR IN FILE SPECIFICATION

Explanation: The file specification was entered incorrectly.

User Action: Reenter the command line, using the correct file specification.

ZAP -- ERROR ON COMMAND INPUT

Explanation: An I/O error occurred while a command line was being read. This could be a hardware error.

User Action: Ensure that the hardware is functioning properly. If it is, reenter the command line. If not, call your DIGITAL Customer Support Center.

ZAP -- I/O ERROR ON TASK IMAGE FILE

Explanation: An I/O error occurred while the file being modified was being read or written. This could be a hardware error.

User Action: Ensure that the hardware is functioning properly. If it is, reenter the command line. If not, call your DIGITAL Customer Support Center.

ZAP -- NO OPEN LOCATION

Explanation: You attempted to modify the contents of a closed location.

User Action: Open the location to perform the modification.

ZAP -- NO SUCH INTERNAL REGISTER

Explanation: The character following a dollar sign was not a valid specification for the internal register.

User Action: Reenter the command line, specifying the correct value.

ZAP -- NO SUCH RELOCATION REGISTER

Explanation: An invalid number was specified for a Relocation Register.

User Action: Relocation Registers are numbered 0 through 7. Any other numbers are illegal. Reenter the command line, specifying a valid Relocation Register number.

ZAP -- NO SUCH SEGMENT

Explanation: The starting disk block was not the start of any segment in the task image file on disk.

User Action: Reenter the command line, specifying the correct disk block address.

ZAP-- NOT A TASK IMAGE OR NO TASK HEADER

Explanation: An error occurred while the segment tables were being constructed. Possibly, the file is not a task image, the /AB switch was not specified, or the task image is defective.

User Action: Terminate the ZAP session, then try invoking ZAP with the /AB switch specified.

ZAP -- NOT IMPLEMENTED

Explanation: You entered a command that is recognized by ZAP, but not implemented.

User Action: Ensure that you entered the command correctly.

ZAP -- OPEN FAILURE FOR TASK IMAGE FILE

Explanation: The file to be modified could not be opened. Possibly, the file does not exist, the file is locked, the device is not mounted, or you do not have write-access to the file.

User Action: Check the file specification for errors, or check your file access privileges. Refer to Section 3.21 for an explanation of the DIRECTORY command and the /FULL qualifier.

ZAP -- SEGMENT TABLE OVERFLOW

Explanation: ZAP does not have enough room in its partition to construct a segment table.

User Action: You cannot use ZAP on the file.

ZAP -- TOO MANY ARGUMENTS

Explanation: You entered more arguments on the command line than are allowed.

User Action: Reenter the command line, specifying the correct syntax.

ZAP -- UNRECOGNIZED COMMAND

Explanation: ZAP did not recognize the command as entered.

User Action: Check the syntax of the command you are trying to execute, then reenter the command line, specifying the correct syntax.

ZAP -- VERIFY FAILURE

Explanation: The V command determined that the contents of a location did not match the expected value. ZAP terminates.

User Action: If applicable, check for errors in the indirect command file. Ensure that the contents of the file are what they should be. Locate the cause of the error and reenter the command line. Ensure that you are correcting the right file or file version.

CHAPTER 10

SOURCE LANGUAGE INPUT PROGRAM (SLP)

The Source Language Input Program (SLP) is a utility used to maintain and audit source files. The optional audit trail in the output files allows you to keep a record of maintenance changes.

SLP is invoked by edit command statements and switches. SLP edit command statements allow you to:

- Update (delete, replace, add) lines in an existing file
- Create source files
- Run indirect files containing SLP edit commands

Input to SLP is a file that you want updated and command input consisting of text lines and edit command lines that specify the update operations to be performed. To locate lines to be changed, SLP uses line numbers or character strings that you specify. Command input can come directly from your terminal or from an indirect command file containing commands and text lines to be inserted into the file. SLP accepts data from any FILES-11 volume.

Output from SLP is a listing file and the updated input file. SLP provides an optional audit trail that helps you keep track of the update status of each line in the file. If an audit trail is not suppressed, it is shown in the listing file and permanently applied to the output file.

You can control SLP processing with SLP control switches. These switches allow you to:

- Suppress audit trails

- Specify the length and beginning position of the audit trails
- Calculate the checksum value for the edit commands
- Generate a double-spaced listing

To invoke SLP type:

```
EDIT/SLP (refer to EDIT/SLP In Chapter 3)

or

$ RUN $SLP
```

SLP will prompt:

```
SLP>
```

You should not specify TI: as your output file, because when you finish editing, you will not have a copy of the output file and the input file will be the same as before you began editing.

10.1 SLP INPUT AND OUTPUT FILES

SLP requires two types of input, an input file and command input. The input file is the source file you want to update using SLP. Command input consists of SLP edit commands and, optionally, new lines of text to be placed in the file.

SLP output consists of an output file and a listing file. The output file is the updated input file. The listing file is a copy of the output file with line numbers added. Both show the changes SLP made to the file.

10.1.1 The Input File

The input file is the file to be updated by SLP. It can contain as many lines of text as are required. When SLP processes the input file, it makes the changes specified by SLP edit commands. If an audit trail is generated, these changes are noted in the output files.

10.1.2 Command Input

SLP uses command input to update files. Command input can be entered interactively after you invoke the SLP utility or indirectly by means of indirect command files.

You enter command input to SLP in two modes: command mode and edit mode. After it is invoked, SLP is in command mode, where the first line entered must be the command line defining the files to be processed. When SLP accepts this line, it initializes the files you want processed. Once these files are initialized, SLP enters edit mode, where it interprets the lines you enter as SLP edit commands or new input lines.

You terminate command input with a single slash as the first character of an edit command line.

The following example shows the general form of command input:

```
MYFILE.MAC;2/CS/AU:55:10,MYFILE.LST;1=MYFILE.MAC;1
-3,,/;BJ007/
CMP (R1)+,B
-4,4
DEC R2
/
```

NOTE

Numeric values given for switches default to octal. Decimal values must be followed by a period (.). The default position for the audit trail is 80 (decimal) and its default length is 8 (decimal); no more than 14 (decimal) characters may be specified. (See Section 10.4.2 for more information about the audit trail.)

The first line is the command line, where you define the output file, the listing file, and the input file. The next four lines comprise the SLP edit commands and input lines.

Note that the input and output files in the example have the same file name and file type; only the versions are different. To ensure that the correct files are processed, specify the version numbers explicitly when you enter the SLP command line. Wildcards cannot be used in any of the file specifications.

You can also calculate the checksum value for the edit commands. Specify the checksum switch with either the input or output file specification in the format:

/CS[:n]

The checksum value can be calculated for all SLP edit command lines. The checksum value cannot be calculated for the following:

- The command line specifying the input and output files.
- Comments in the edit command lines.
- Any spaces and/or tabs between characters included in the checksum calculation and those characters excluded from the calculation.
- The second comma and anything following it in an edit command line (that is, audit trail and/or comment).
- Comment delimiter (specified by the first character of the last audit trail string before the current delimiter) and any characters following it in an input line, whether or not it is being used in the line as a delimiter. The value is then reported in a message on your terminal. If you specify a value for the checksum and it is not the same as the calculated checksum, you will get a diagnostic error message. (The messages are described in Section 10.5.2.)

10.1.3 The SLP Listing File

The SLP listing file shows the updates made to the source file. Each line in the listing file is numbered. Updates are marked by means of the audit trail if one has been generated. The examples given throughout this chapter contain samples of listing files.

10.1.4 The SLP Output File

The SLP output file is the updated input file. All of the updates specified by command input are inserted in this file. The audit trail, if specified, is applied to lines changed by the update. The audit trail is included in the output file. The numbers generated by SLP for the listing file do not appear in the output file.

10.2 HOW SLP PROCESSES FILES

This section describes how SLP porcesses files when it receives the following command line and edit commands.

```
MYFILE.MAC;2/AU:55:10,MYFILE.LST/-SP=MYFILE.MAC;1
-3
CMP (R1)+,B
-4,4
DEC (R2)
/
```

This is the input file (MYFILE.MAC;1) before SLP processes the files:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (R0)+,A
BNE     20$
INC     R1
```

The following is the listing file (MYFILE.LST;1) resulting from SLP processing of these files:

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  CMP     (R1)+,B         ;**NEW**
 5.  DEC     (R2)            ;**NEW**
 6.  BEQ     END             ;**-1
 7.  CLR     R1
 8.  MOV     R2,NUMC
 9.  CMPB    (R0)+,A
10.  BNE     20$
11.  INC     R1
```

The audit trail shows the new lines (;**NEW**) and indicates where lines have been removed (;**-1). (The audit trails ;**NEW** and ;**-n are automatically generated by SLP if you have not suppressed audit trail generation or if you have not specified another audit trail string.) In this case, a line has been added after line 3, and line 4 has been deleted and a new line added in its place.

SLP processes an input file using command input. When processing begins, SLP writes each line from the input file into the output file until it reaches a line to be modified, as requested in the command input. When SLP reaches a line to be modified, it modifies the line, notes the change by means of the audit trail, and then continues writing lines to the output file until another command is encountered or until end-of-file is reached.

10.3 USING SLP

This section describes how to:

- Specify the SLP edit commands.
- Update files using the SLP edit commands.
- Enter SLP commands interactively and by means of indirect command files.
- Create a source file using SLP.

10.3.1 Specifying SLP Edit Commands

The SLP edit commands allow you to update source files by adding, deleting, and replacing lines in a file. SLP allows you to enter lines sequentially. Once past a given line in the file, you cannot return the line pointer to that line. To return the line pointer to that line, you must begin another SLP editing session. You enter SLP edit commands after invoking SLP and specifying an edit command line.

The general format of the SLP edit command line is as follows:

```
-[locator1][,locator2][,/audittrail/][; comment]
inputline
    .
    .
    .
```

- (dash)

Identifies a SLP edit command line.

locator1

A line locator that causes SLP to move the current line

pointer to a specified line. If you specify only locator1, the current line pointer is moved to that line and SLP reads the next line in the command input file. This field can be specified using any of the locator forms described later in this section.

locator2

A line locator that defines a range of lines (that is, the range beginning with locator1 and ending with locator2, inclusive) to be deleted or replaced. This field can be specified using any of the locator forms described later in this section.

/audittrail/

A character string used to keep track of the update status of each line in the file. The string must be enclosed within slashes (/). It consists of a comment delimiter as the first character and then a text string. The semicolon (;) is the default delimiter for audit trails automatically generated by SLP (;**NEW** and ;**-n). The comment delimiter specified in audittrail (usually a semicolon) is the new delimiter for all subsequent audit trails until redefined by a later audittrail.

inputline

A line of new text to be inserted into the file immediately following the current line. You can enter as many input lines as required.

comment

A line of text (delimited by a semicolon) at the end of the SLP edit command line that appears only in the command input file.

All fields in the SLP edit command line are positional and commas must be specified.

The locator fields can take one of the following forms:

```
-/string/[+n]
-/string...string/[+n]
-number[+n]
-.[+n]
```

string

A string of ASCII characters. SLP locates the line where

the string exists and moves the current line pointer to that line. If the locator is specified in the form /string...string/, SLP locates the line where the two character strings delimit a larger character string abbreviated by an ellipsis (...).

number

A decimal line number where the current line pointer is to be moved. The largest line number that can be specified is 9999.

. (period)

The current line.

n

A decimal value used as an offset from the line specified by the locator. You cannot use +n by itself. It must be specified with a number or string locator or a period. SLP moves the current line pointer n lines beyond the line specified in the locator field.

Although the values for number and n are taken as decimal, remember that all other SLP values are octal by default.

All forms of the line locator can be specified interchangeably in the SLP edit command lines.

10.3.2 Entering SLP Edit Commands

Once you have invoked SLP, you can enter SLP edit commands interactively or by specifying indirect command files. In both cases, the first command you must enter is the command line defining the files to be processed during this SLP session. This section gives examples of how to use both methods of entering SLP commands.

The following file (BASE.MAC;1) is used as the input file for the examples in this section:

```
MOV     #$SWTCH,R3
CLR     $ERFLG
CLR     $CRCVL
CLR     $CSSV
MOV     SPSAV,SP
MOV     #$CFNMB,R0
MOV     #<$HDSIZ-$CFNMB>/2+1,R1
```

```
CLR     (R0)+
DEC     R1
BNE     5$
```

10.3.2.1 Entering SLP Commands Interactively - To alter the example file interactively, invoke SLP. Once you have entered the SLP command mode, SLP does not display prompts. The first line you enter must always be the command line defining the files you want processed during this session:

```
BASE.MAC;2/AU:48./TR,BASE.LST=BASE.MAC;1
```

Then you enter the edit commands and input lines:

```
-3
TST     R1
-4,4
BEQ     10$
-6,,/;JM010/
CLR     R2
/
```

In this example, the edit commands instruct SLP to do the following: -3 inserts a new line after line 3; -4,4 deletes line 4 and replaces it with a new line; -6,,/;JM010/ inserts a line after line 6 with a new audit trail value.

When you have entered all the corrections, enter the slash (/) to terminate the edit session. SLP processes the files and returns control to you with the prompt:

```
SLP>
```

This returns SLP to command mode. You can then enter another input file and begin another editing session.

The listing file (BASE.LST;1) resulting from SLP processing appears as follows:

```
1.  MOV     #$SWTCH,R3
2.  CLR     $ERFLG
3.  CLR     $CRCVL
4.  TST     R1                          ;**NEW**
5.  BEQ     10$                         ;**NEW**
6.  MOV     SPSAV,SP                    ;**-1
7.  MOV     #$CFNMB,R0
8?  CLR     R2                          ;JM010
9.  MOV     #<$HDSIZ-$CFNMB>/2+1,R1
```

```
10.  CLR     (R0)+
11.  DEC     R1
12.  BNE     5$
```

The /TR switch (/TR in the command line) records the truncation of lines by the audit trail. In the listing file, a question mark (?) replaces the period (.) after the line number for the lines that were truncated. It is possible that audit-trail strings in the input file will be truncated by the new audit-trail string, although the commands or text strings will not be truncated.

10.3.2.2 Entering SLP Commands Using Indirect Command Files - To alter the example file by using the SLP edit commands in the indirect command file, BASE.SLP, you invoke SLP and SLP responds with the prompt:

```
SLP>
```

You then enter the file specification for the indirect command file containing the command line, the SLP edit commands, and the input lines:

```
@BASE.SLP
```

SLP processes the files just as if you entered the commands and input lines interactively, returning control to you with the prompt:

```
SLP>
```

The output listing resulting from indirect command file processing is exactly like the output listing resulting from the same changes made interactively.

Indirect command files can be nested to a maximum level of three. This permits indirect command files to reference a text file.

10.3.2.3 Using SLP Operators - In addition, you can enter special characters called operators, which perform specific functions. Table 10-1 lists the operators and the function each performs. Enter operators, in edit mode, as the first character of an input line.

Table 10-1: SLP Operators

Operator	Function
-	Identifies the first character of a SLP edit command line.
\	Suppresses audit-trail processing.
%	Reenables audit-trail processing.
@	Invokes an indirect command file for SLP processing.
/	Terminates the SLP edit session, and then returns to SLP command mode.
<	Enables you to enter characters in the input file that SLP otherwise would interpret as operators. For example, </ hides the slash character from SLP, thereby enabling you to enter the slash into the output file without terminating the SLP editing session. This character can be used with all SLP operators.

10.3.3 Updating Source Files With SLP

This section describes the procedure for generating a numbered listing for use in editing source files by line number. The section also describes how to use SLP to add, delete, and replace lines in a file.

10.3.3.1 Generating a Numbered Listing - SLP processes input by line number. However, line numbers appear only in the listing file; they are not written to the output file. To use SLP effectively, you should use a numbered listing when you prepare command input. To generate a numbered listing, first invoke SLP, then enter the command line in the format:

```
,listfile=infile
/
```

In this format, listfile is the name you assign to the listing file SLP will produce and infile is the name of the input file whose lines are to be numbered. The slash (/) terminates edit mode. For example, suppose the input file is:

```
MOV     R1,-(SP)
BIC     #177770,@SP
ADD     #60,@SP
MOVB    (SP)+,-(R0)
ASR     R1
ASR     R1
ASR     R1
DEC     R2
BNE     30$
MOV     #MSG,R0
```

SLP processes each line to generate a numbered list file (list file;1):

```
 1.  MOV     R1,-(SP)
 2.  BIC     #177770,@SP
 3.  ADD     #60,@SP
 4.  MOVB    (SP)+,-(R0)
 5.  ASR     R1
 6.  ASR     R1
 7.  ASR     R1
 8.  DEC     R2
 9.  BNE     30$
10.  MOV     #MSG,R0
```

10.3.3.2 Adding Lines to a File - The three SLP edit command formats for adding lines to a file are:

```
-locator1
inputline
    .
    .
    .
```

or

```
-locator1,,
inputline
    .
    .
    .
```

or

```
locator1,,/audittrail/
inputline
    .
    .
    .
```

The following example shows how to add lines to a file. The command input consists of the following lines:

```
MYFILE.MAC;2/AU:48.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1
-3
CMP     (R1)+,B
-4,4
DEC     R2
-6,,,/;JM010/
INC     R3
-9,,/;BJ008/
BEQ     10$
/
```

The next example uses text rather than line numbers to indicate where new lines should be added or deleted:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/BEQ/
CALL    WRITE
/
```

In this example, the edit command /BEQ/ instructs SLP to insert a line after the line with the first occurrence of BEQ.

SLP processing generates the following listing file (MYFILE.LST;1):

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  TST     R2
 5.  BEQ     END
 6.  CALL    WRITE                   ;**NEW**
 7.  CLR     R1
 8.  MOV     R2,NUMC
 9.  CMPB    (R0)+,A
10.  BNE     20$
11.  INC     R1
```

SLP has numbered the lines and applied an audit trail to the line following line 5, where SLP found the first occurrence of the string BEQ.

The next example uses the same input file and the following new command lines:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/#SIZ/+2
CMP     (R1)+,B
/
```

SLP processing generates the following listing file (MYFILE;1):

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  TST     R2
 5.  CMP     (R1)+,B                    ;**NEW**
 6.  BEQ     END
 7.  CLR     R1
 8.  MOV     R2,NUMC
 9.  CMPB    (R0)+,A
10.  BNE     20$
11.  INC     R1
```

Again, SLP has numbered the lines and this time the new input line is inserted so that it is two lines beyond the line containing the first occurrence of the string /#SIZ/.

10.3.3.3 Deleting Lines from a File - The SLP edit command format for deleting lines from a file is:

```
-[locator1],[locator2],[/audittrail/][; comment]
```

In this format, locator1 and locator2 can be any of the forms of the locator fields described in Section 10.3.1; locator1 specifies the line where SLP is to begin deleting lines; locator2 specifies the last line to be deleted. SLP deletes all lines from locator1 through locator2, inclusive.

Suppose an input file consists of the following lines:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (R0)+,A
BNE     20$
INC     R1
```

To delete lines from the file, the command input can consist of the following commands and text lines:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/MOV...R1/,/NUMC/
/
```

SLP processing generates the following listing file (MYFILE;1):

```
1.  MOV     #BUF1,R0
2.  CMPB    (R0)+,A
3.  BNE     20$
4.  INC     R1                      ;**-6
```

In this example, the ellipsis (...) abbreviates the larger string MOV #SIZ,R1. Assuming the two strings bracket a larger string, SLP searches for the first occurrence of the string MOV and then the first occurrence on the same line of the string R1, in this case the string MOV #SIZ,R1. SLP begins deleting lines at this line and continues deleting lines until it deletes the last line of the given range, specified here by the string NUMC. SLP applies the audit-trail count of the lines it deleted to the next line from the input file.

Using the same input file as used in the previous example, the following example shows how to delete a single line using the period locator. The command input for this example is:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-/MOV #SIZ,R1/,.
/
```

SLP processing generates the following listing file (MYFILE;1):

```
1.  MOV     #BUF1,R0
2.  CALL    READ                    ;**-1
3.  TST     R2
4.  BEQ     END
5.  CLR     R1
6.  MOV     R2,NUMC
7.  CMPB    (R0)+,A
8.  BNE     20$
9.  INC     R1
```

SLP moves the current line pointer to the line containing the string MOV #SIZ,R1 and then finds the period as the second locator field. Since the second locator field is specified as the current line, SLP deletes the current line.

10.3.3.4 Replacing Lines in a File - A replacement is the deletion of old text followed by the insertion of new text. The number of lines deleted need not match the number of lines added. To replace lines in a file, use the same SLP edit command format as used in the delete command. The first line locator field specifies the first line to be deleted. The second line locator field defines the last line in the range to be deleted and where the new text is to be inserted. For example:

```
-4,.+4
```

This command instructs SLP to move the line pointer to line 4, and replace line 4 and the next four lines with new input lines.

The following example shows how to delete lines from a file and replace them with new lines. The input file consists of the following lines:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
```

The command input is:

```
MYFILE.MAC;2/AU:50,MYFILE.LST=MYFILE.MAC;1
-2,.+1
CMP     (R1)+,B
INC     R2
/
```

In this example, the edit command, -2,.+1, instructs SLP to delete lines 2 and 3 and insert two new lines.

SLP processing generates the following listing file (LISTING;1):

```
1.  MOV     #BUF1,R0
2.  CMP     (R1)+,B                 ;**NEW**
3.  INC     R2                      ;**NEW**
4.  TST     R2                      ;**-2
5.  BEQ     END
6.  CLR     R1
7.  MOV     R2,NUMC
```

10.3.4 Creating Source Files Using SLP

Using SLP to create source files is possible, but not recommended. SLP does not have an intraline editing mode and you cannot return to a line once you have passed it. An interactive editor, EDT, is better for creating source files.

To create source files using SLP, invoke SLP and enter the command line in the format:

```
outfile/-AU[/sw][,listfile][/sw]=[primary_input_device:[/sw]]
```

outfile

The file specification for the output file. The default device is SY0:.

/-AU

Specifies that an audit trail is not to be generated. Otherwise, you will get the ;**NEW** audit trail on every line of the output files.

listfile

The file specification for the listing file (optional). The default device is implied by the output file specification.

primary_input_device:

Specifies that input for the file being created is coming from this device, for example, a terminal. The default device is your primary input device.

/sw

Specifies any optional SLP switches.

The following file specification creates a new file called MYFILE.MAC from the terminal and puts it on SY0:.

```
MYFILE.MAC/-AU=TI:
```

Once you have entered the file specification, SLP accepts each line as a variable-length record of up to 132 (decimal) characters. Trailing blanks and tabs on input lines are deleted. SLP expects input to the file to come from the primary input device. End the SLP session with a slash (/) and then a CTRL/Z.

10.4 CONTROLLING SLP

The SLP switches allow you to calculate the checksum value for the edit commands and to control the generation and format of the listing file and the output file.

10.4.1 SLP Switches

SLP output consists of two files--a listing file and the output file, which is the modified version of the input file. You can use the SLP switches to control the audit trail and print options associated with the two files.

The effects of SLP switches are the same whether you apply them to input or output files. Table 10-2 lists the SLP switches and gives a brief description of the functions each performs.

Table 10-2: SLP Switches

Switch	Function
/AU /-AU	Allows you to generate an audit trail or suppress audit-trail generation and specify the beginning field and length of the audit trail. /AU is the default value. See the following sections for more information about the /AU switch.
/BF /-BF	Positions the audit trail by inserting spaces instead of tabs at the end of text information. /BF is the default value.
/CM[:n]	Deletes audit trails and any trailing spaces or tabs, and truncates the text at a specified horizontal position. The value given for the beginning position of the audit trail is the default value for this switch. See Section 10.4.6 for more information about the /CM switch.
/CS[:n]	Calculates the checksum value for the edit commands. If you do not specify n, SLP reports the value in a message on your terminal. If you do specify n and the checksum value that SLP calculates is not the same as the one you specified, SLP displays a diagnostic error message.

Switch	Function
	The procedure SLP uses to calculate the checksum value for the edit commands is described in Section 10.1.2.
/DB /-DB	Generates the listing file in double-space format. /-DB is the default value.
/TR	Reports truncation of lines by the audit trail. If line truncation occurs, you will get a diagnostic error message. There is no default value for this switch. In the listing file, a question mark (?) replaces the period (.) in the line number of the lines that were truncated.
/SQ	Sequences the lines in the output file so that the numbers reflect the line numbers of the original input file. New lines added to the file have the same number as the preceding line. This allows the MACRO Relocatable Assembler to output listing files that contain the original line numbers, thus easing the process of updating correction files. If you specify a listing file, SLP preserves the line numbers of the input file but does not display numbers for the new lines that have been inserted.
/RS	Resequences the lines in the output file so that the line numbers are incremented for each line written to the output file. The /RS switch overrides the /SQ switch.
/NS	Does not sequence the lines in the output file. New lines are indicated by the audit trail (if specified). The /NS switch is the default condition and overrides the /SQ and /RS switches.

10.4.2 Controlling the Audit Trail

The /AU switch allows you to generate, suppress, and set the length and contents of the audit trail. To suppress generation of the audit trail, specify the /-AU switch in either the input or output file specification. For example, either of the following command lines generates an output file with no audit trail:

```
DW1:MYFILE.MAC;3/-AU,MYFILE.LST:=MYFILE.MAC;2

DW1:MYFILE.MAC;3,MYFILE.LST:=MYFILE.MAC;2/-AU
```

By default, SLP automatically generates an audit trail; that is, you need not explicitly specify the /AU switch in your command line (unless you want to specify the beginning position and length of the audit trail).

10.4.3 Setting the Position and Length of the Audit Trail

You can set the beginning position of the audit trail and the length of the audit trail using the /AU switch in the format:

```
/AU:position:length
```

position

A number, less than or equal to 132 (decimal), designating the beginning character position of the audit trail on the line. SLP rounds this value to the next highest tab stop (a multiple of 8). The default value for position is 80 (decimal).

NOTE

Numeric values given for switches default to octal. Decimal values must be followed by a period (.). The default position for the audit trail is 80 (decimal) and its default length is 8 (decimal); no more than 14 (decimal) characters may be specified. (See Section 10.4.2 for more information about the audit trail.)

length

The length of the audit trail. The default value for length is 8 (decimal) characters; no more than 14 (decimal) characters may be specified.

The following example shows how to specify the beginning position and length of the audit trail. The input file for this example is:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
```

The command input is:

```
MYFILE.MAC;2/AU:30.:10./TR,MYFILE.LST=MYFILE.MAC;1
-2,.+1,/;CHANGE001/
CMP     (R1)+,B
DEC     R2
/
```

The listing file MYFILE.LST;1 resulting from SLP processing is:

```
1.  MOV     #BUF1,R0
2.  CMP     (R1)+,B         ;CHANGE001
3.  DEC     R2              ;CHANGE001
4.  TST     R2              ;**-2
5.  BEQ     END
```

10.4.4 Changing the Value of the Audit Trail

To change the value of the audit trail, specify:

```
-[locator1],[locator2],/;new value/
```

The following example shows how to change the audit trail values. The input file consists of the following lines:

```
MOV     #BUF1,R0
MOV     #SIZ,R1
CALL    READ
TST     R2
BEQ     END
CLR     R1
MOV     R2,NUMC
CMPB    (R0)+,A
BNE     20$
INC     R1
```

The command input consists of the following commands and text lines:

```
MYFILE.MAC;2/AU:48.:10./TR,MYFILE.LST/-SP=MYFILE.MAC;1
-3
CMP     (R1)+,B
-4,4
DEC     R2
-6,,/;JM010/
INC     R3
-9,,/;BJ008/
BEQ     10$
/
```

In this example, the edit commands instruct SLP to insert a line after line 3, to delete and replace line 4, and to insert new lines after lines 6 and 9 with new audit trail values.

The listing file (MYFILE.LST) resulting from SLP processing appears as follows:

```
 1.  MOV     #BUF1,R0
 2.  MOV     #SIZ,R1
 3.  CALL    READ
 4.  CMP     (R1)+,B                                 ;**NEW**
 5.  DEC     R2                                      ;**NEW**
 6.  BEQ     END                                     ;**-1
 7.  CLR     R1
 8.  INC     R3                                      ;JM010
 9.  MOV     R2,NUMC
10.  CMPB    (R0)+,A
11.  BNE     20$
12.  BEQ     10$                                     ;BJ008
13.  INC     R1
```

10.4.5 Temporarily Suppressing the Audit Trail

You can temporarily suppress the generation of the audit trail by using the backslash (\) operator. You can then reenable audit-trail processing with the percent sign (%) operator. (You cannot enable audit trail processing with this operator if you have specified the /-AU switch in the SLP command line.)

Both operators are entered in the command input. The backslash (\) is specified in column 1 of the line that precedes those commands and/or input files for which you do not want audit-trail processing. The percent sign (%) is specified in column 1 of the line that precedes the lines for which you do want processing. For example:

```
BAK.MAC;26/AU/-BF=BAK.MAC;25
\
-2,2
          .IDENT  /05.03/
-23,23
; VERSION 05.03
-37,,
;  J. MATTHEWS      11-NOV-80
;
;       JM011      CORRECT OUT-OF-BOUNDS CONDITION FOR
INPUT-BUFFER
;                  SIZE
;
```

```
%
-106,106,/;JM011/
   CMP     #132.,R3          ; IS INPUT-BUFFER SIZE IN RANGE?
   BLT     30$               ; IF LT, NO
   .
   .
   .
/
```

The lines between the backslash (\) and the percent sign (%) are not affected by audit-trail processing. The lines following the percent sign (%) are affected.

10.4.6 Deleting the Audit Trail

The /CM switch allows you to delete audit trails and trailing spaces and tabs from a file. The /CM switch applied to the output or input file specification accepts a numeric argument that specifies the beginning position of an audit trail or other text string to be deleted. The default for this argument is the position argument given for the /AU switch (or its default, decimal 80). This value is rounded to the next highest tab stop before use.

When processing an input line, SLP first truncates the text to the next highest tab stop after the position specified, and then deletes any trailing spaces or tabs. The remaining text is copied to the output file.

The /CM switch is specified in the form:

```
/CM:[n]
```

n

A number designating the beginning character position of the audit trail (or other text) to be deleted.

For example:

```
SLP>SLPR11.MAC;12/CM:119.=SLPR11.MAC;11
/
```

In this case, the input lines are truncated to a length of 120 (decimal) characters. The specified length is rounded up to the next highest tab stop and the audit trail begins at column 121 (decimal). Trailing spaces and tabs are deleted before each line is copied to the output file.

In the following example, SLP truncates input lines to the default position of the audit trail, column 80 (decimal).

```
SLP>SLPR11.MAC;12=SLPR11.MAC;11/CM
/
```

10.5 SLP MESSAGES

SLP messages are divided into two groups: information and error. The messages and suggested responses are given in the following sections. Section 10.5.1 describes the information message and Section 10.5.2 describes the error messages.

10.5.1 SLP Information Message

```
SLP -- COMMAND FILE CHECKSUM IS ######
```

Explanation: By specifying the /CS[:n] switch in the command line, you requested SLP to calculate the checksum value for the edit commands.

User Action: This message is for your information only. No action is required.

10.5.2 SLP Error Messages

This section lists the SLP error messages. Following each message is an explanation of the error and recommended user action to correct the error.

SLP error messages are issued in two formats:

- SLP followed by a dash, the type of error message, and the error message. If applicable, the command line or command line segment that caused the message is printed on the next line. For example:

```
SLP -- *FATAL*-ILLEGAL SWITCH OR FILESPEC
SHIRLEY.MAC;2/CF
```

- SLP followed by a dash, the type of error message, the error message, and the name of the file with which the error is associated. For example:

```
SLP -- *FATAL*-OPEN FAILURE LINE LISTING FILE filename
```

Note that all but two of the SLP error messages are fatal. The two exceptions are diagnostic messages, which are described at the end of this section.

SLP -- *FATAL*-COMMAND SYNTAX ERROR
command line

Explanation: The command line format did not conform to syntax rules. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line, specifying the proper syntax.

SLP -- *FATAL*-ILLEGAL DEVICE NAME
command line

Explanation: The device specified was not a legal device. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line, specifying a legal device.

SLP -- *FATAL*-ILLEGAL DIRECTORY
command line segment

Explanation: The directory was not legally specified. Open files were closed and SLP was reinitialized.

User Action: Reenter the command line, specifying a legal directory.

SLP -- *FATAL*-ILLEGAL ERROR/SEVERITY CODE p1 p2 p3

Explanation: This error message indicates an error in the SLP program.

User Action: Reenter the command line. If this does not correct the problem, call your DIGITAL Customer Support Center.

SLP -- *FATAL*-ILLEGAL FILE NAME
command line segment

Explanation: A file specification was greater than 19 (decimal) characters in length or contained a wildcard (that

is, an asterisk in place of a file specification element). Open files were closed and SLP was reinitialized.

User Action: Reenter the command line, specifying a legal filename.

SLP -- *FATAL*-ILLEGAL GET COMMAND LINE ERROR

Explanation: The system was unable to read a command line. This error message indicates an internal system failure or an error in the SLP program.

User Action: Reenter the command line. If this does not correct the problem, call your DIGITAL Customer Support Center.

SLP -- *FATAL*-ILLEGAL SWITCH OR FILESPEC
command line segment

Explanation: The switch was not a legal SLP switch or a legal switch was used in an illegal manner, or a file specification could not be parsed.

User Action: Reenter the command line, specifying the legal switch, or correct the file specification.

SLP -- *FATAL*-INDIRECT COMMAND SYNTAX ERROR
command line

Explanation: The command line format specified for the indirect command file did not conform to syntax rules. Open files are closed and SLP was reinitialized.

User Action: Reenter the command line, specifying the proper syntax.

SLP -- *FATAL*-INDIRECT FILE DEPTH EXCEEDED
command line

Explanation: More than three levels of indirect command files were specified in an indirect command file. Open files were closed and SLP was reinitialized.

User Action: Correct the indirect command file and reenter the command line.

SLP -- *FATAL*-I/O ERROR COMMAND INPUT FILE

or

SLP -- *FATAL*-I/O ERROR COMMAND OUTPUT FILE

or

SLP -- *FATAL*-I/O ERROR CORRECTION INPUT FILE filename

or

SLP -- *FATAL*-I/O ERROR LINE LISTING FILE filename

or

SLP -- *FATAL*-I/O ERROR SOURCE OUTPUT FILE filename

Explanation: One of the following conditions may exist:

- A problem exists on the physical device.
- The length of the command line was greater than the allowed number of characters.
- The file is corrupted or the format is incorrect.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

SLP -- *FATAL*-INDIRECT FILE OPEN FAILURE
command line

or

SLP -- *FATAL*-OPEN FAILURE CORRECTION INPUT FILE filename

or

SLP -- *FATAL*-OPEN FAILURE LINE LISTING FILE filename

or

SLP -- *FATAL*-OPEN FAILURE SOURCE OUTPUT FILE filename

Explanation: One of the following conditions may exist:

- The file is protected against an access.
- A problem exists with the physical device
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.
- The available Executive dynamic memory is insufficient for the operation.

These errors cause open files to be closed and SLP to be reinitialized.

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

SLP -- *FATAL*-LINE NUMBER ERROR
command line

Explanation: The command line printed contained an illegally specified numeric line locator.

User Action: Terminate the SLP edit session and refer to the rules for specifying numeric line locators in Section 10.3.1. Correct the error and reenter the command line.

SLP -- *FATAL*-PREMATURE EOF CORRECTION INPUT FILE filename

Explanation: An out-of-range line locator was specified in an indirect command file or from the terminal; for example, -990 was specified for an 800-line file.

User Action: Terminate the current editing session. Restart the editing session, and enter the edit command line, specifying the correct line number.

SLP -- *FATAL*-PREMATURE EOF COMMAND INPUT FILE

Explanation: This is caused by not terminating SLP command input with a slash (/) or by inadvertently typing CTRL/Z at the terminal, which sends an end-of-file to SLP before the slash (/) character is read. SLP prompts (SLP>), indicating that a new file specification is expected.

User Action: Restart the editing session at the point where the CTRL/Z was typed.

SLP -- *DIAG*-ERROR IN COMMAND FILE filespec CHECKSUM

Explanation: An incorrect value was specified for the command file checksum. If you enter the edit command lines directly from the terminal, the command file in the error message is CMI.CMD. Thus, the error message reads:

SLP -- *DIAG*-ERROR IN COMMAND FILE CMI.CMD CHECKSUM

User Action: This is a warning message only. The specified output file is still created, although possibly not as intended.

SLP -- *DIAG*-n LINES TRUNCATED BY AUDIT TRAIL
command line

Explanation: Line truncation by the audit trail was detected.

User Action: This is an informational message only. The specified output file is still created. (In the listing file, a question mark (?) replaces the period (.) in the line number of the lines that were truncated. It is possible that audit-trail strings from the input file will be truncated by the new audit-trail string although text strings will not be truncated.) Determine where the truncation(s) occurred. If necessary, modify the command file so that it contains commands that do not cause truncation.

CHAPTER 11

CONVERT UTILITY (CVT)

The Convert Utility (CVT) provides you with a convenient means of converting an integer value into several different radixes.

To invoke CVT, enter the following command from DCL:

```
$ RUN $CVT
```

You receive the following prompt:

```
CVT>
```

Once you receive the CVT prompt, you can enter an integer expression. Figure 11-1 illustrates the output of CVT when you enter the value 15 (octal).

```
$ RUN $CVT   ! Execute the Convert Utility.
CVT> 15
13.  13.,0.   000015   015,000  $000D  %  M  "<CR ><NUL>
 ↑     ↑       ↑        ↑       ↑     ↑     ↑
Decimal Decimal Octal    Octal   Hex   RAD50 ASCII
Word    Bytes   Word     Bytes   Word  Word  Bytes
```

Figure 11-1: Output of the CVT Utility

CVT displays a row of numbers that represent the value 15 (octal) converted into a decimal word, decimal bytes, octal word, octal bytes, hexadecimal word, RAD50 word, and ASCII bytes.

By default, CVT interprets the expression you specify as an octal value. However, you can force CVT to interpret the expression as decimal, hexadecimal, RAD50, or ASCII. To do this, you use the punctuation described in Table 11-1. Note that CVT uses the same punctuation in its output display to indicate the radix of each value.

Table 11-1: Punctuation to Specify Radix

Name	Example	Description
Period	13.	Forces decimal interpretation. Place the period immediately after the value you want interpreted as decimal.
Dollar Sign	$D	Forces hexadecimal interpretation. Place the dollar sign immediately before the value you want interpreted as hexadecimal.
Percent Sign	%M	Forces RAD50 interpretation. Place the percent sign immediately before the value you want interpreted as RAD50. You can enter from one to three RAD50 characters at a time.
Apostrophe	'x	Forces ASCII interpretation of 1 byte. Place the apostrophe immediately before the value you want interpreted as an ASCII byte. You can enter no more than one character after an apostrophe.
Quotation Mark	"xx	Forces ASCII interpretation of two bytes. Place the quotation mark immediately before the value you want interpreted as two ASCII bytes.

CHAPTER 12

PERIPHERAL INTERCHANGE PROGRAM (PIP)

The Peripheral Interchange Program (PIP) is a file utility program that transfers data files from one standard FILES-11 device to another. PIP also performs file control functions.

Some of the functions PIP performs are:

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

12.1 INVOKING PIP

You can invoke PIP in two ways:

1. Invoke the DCL commands that manipulate files; these commands in turn invoke PIP. See Section 1.1.1 for a list of file-handling commands.

2. Invoke PIP directly from the DCL command level.

To invoke PIP directly from the DCL command level, enter the following command:

```
$ RUN $PIP
```

You receive a new prompt to indicate that you are in the PIP environment:

```
PIP>
```

Once you are in the PIP environment, PIP waits for your command. The following section describes the command format.

12.2 FORMAT OF PIP COMMANDS

The simplest format for the PIP command line is:

```
outfile=infile
```

outfile

The output file specification. If the output filename, file type, and version are either defaulted or *.*;*, the input filename, file type, and version are used for the output file (see /NV and /SU subswitches).

If you explicitly specify any portion of the output file specification (filename, file type, or version), wildcards cannot be used in this specification.

Similarly, for a copy command, if you enter any portion of the output specification, you can enter only one file as the input file.

infile

The input file specification. If the filename, file type, and version fields are not specified, then *.*;* is the default.

One switch that you can specify when copying FILES-11 files is the Merge switch (/ME). The Merge switch creates a new file from two or more existing files. PIP assumes /ME when you explicitly specify an output file, two or more input files, and no switches.

Because the basic copy function and the Merge switch are logically related, the Merge switch is described here rather than with the other switches.

The general format of the PIP command line is:

```
outfile=infile1[,infile2...,infilen][[/ME][/subswitch]]
```

outfile

The output file specification.

infile

The input file specification.

/ME

The Merge switch.

/subswitch

Specifies any of the subswitches that you can enter as part of the basic command line or with the Merge switch. (Table 12-2 describes these subswitches.)

Subswitches can appear in either the output or input file specification. If you place the subswitch in an input file specification, it applies only to that file. If you place the subswitch in the output file specification, it applies to the entire list of input specifications.

12.3 PIP DEFAULTS FOR FILE SPECIFICATION FIELDS

With the exception of the version number, PIP generally uses the last value encountered in the command line as the default. That is, PIP uses values you enter to set defaults and changes the default when you change the value. Exceptions to this are noted in the descriptions of each switch.

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

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

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

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

Table 12-1 summarizes the rules PIP uses to set defaults.

Table 12-1: PIP Default File Specifications

Field	Default Value
dev:	For the first file specification, the unit on which the user's system disk is mounted (SY0:) or the default that you specify with the /DF switch. For subsequent file specifications, either you explicitly specify a new device or PIP assumes the device from the previous specification.
[ufd]	For the first file specification, your current User Identification Code (UIC)--the UIC under which you log on, the UIC you specify with the SET command, or the default you specify with the /DF switch. For subsequent file specifications, either you explicitly specify a new User File Directory or PIP assumes the UFD from the previous specification. Only the asterisk specification is valid as a wildcard.
filename	No default for the first file specification. For subsequent file specifications, the last file name that you explicitly specified. Asterisk and percent sign specifications are valid as wildcards.
.filetype	No default for the first file specification. For subsequent file specifications, the last file type that you explicitly specified. Asterisk and percent sign specifications are valid as wildcards.

PIP DEFAULTS FOR FILE SPECIFICATION FIELDS

Field	Default Value
;version	The default for input files is the most recent version number. The default for output files is the next higher version number, or version 1 if the file does not exist in the output directory. An exception is the PIP file delete function, which requires that a version number be specified. An explicit version number is defined to be of the form ;n where n is greater than 0. A version number of ;-1 may be used to specify the oldest version of a file. A version number of ;0 or ; may be specified to signify the most recent version. In certain cases, just the asterisk (wildcard) may be specified.

12.4 PIP SWITCHES AND SUBSWITCHES

PIP provides several file control switches and subswitches. A switch specification consists of a slash (/) followed by a two- or three-character switch name.

The switch specification is optionally followed by a subswitch name separated from the switch name by a slash. The switch or subswitch can have arguments that are separated from the switch or subswitch name by a colon (:).

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

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

Table 12-2 lists PIP switches and subswitches and summarizes the functions performed by them. The subswitches are listed with their respective switches. The switches and subswitches are described in detail in Section 12.4.1.

Table 12-2: PIP Switches and Subswitches

Switch	Subswitch	Function
/AP		Appends file(s) to the end of an existing file.
	FO	Specifies the file owner for a file.
/CD		Allows the output file to take the creation date of the input file rather than the date of transfer.
DE		Deletes one or more files.
	LD	Lists the deleted files.
DD		Restricts file searches to files created during a specified period of time.
DF		Changes PIP's default device and/or UFD.
EN		Enters a synonym for a file in a directory file.
	NV	Forces the version number of a file to one greater than the latest version.
EOF[:block:byte]		Specifies the end-of-file pointer for a file.
EX		Excludes one file specification from the PIP operation.
/FI:filenum:seqnum		Accesses a file by its file identification number (file-ID).
FR		Displays the amount of available space on the specified volume, the largest contiguous free space on that volume, and the number of available file headers.
ID		Identifies the version of PIP being used.

PIP SWITCHES AND SUBSWITCHES

Switch	Subswitch	Function
LI		Lists directory files.
	BR	Lists a directory file in brief format (an alternative mode for the /LI switch).
	FU[:n[.]]	Lists a directory file in full format (an alternative mode for the /LI switch).
	TB	Lists the total number of blocks used for a directory, along with the total number of blocks allocated and the number of files in that directory (an alternative mode for the /LI switch).
ME		Concatenates two or more files into one file.
	BL:n[.]	Allocates a number (n) of contiguous blocks.
	CO	Specifies that the output file(s) be contiguous.
	FO	Specifies the file ownership for a file.
	NV	Forces the version number of a file to one greater than the latest version.
	SU	Supersedes (replaces) an existing file.
NM		Suppresses certain PIP error messages.
/PR		Changes the protection status of a file.
	FO	Specifies the ownership for a file.
	GR[:RWED]	Sets the read/write/extend/delete protection at the group level.
	OW[:RWED]	Sets the read/write/extend/delete protection at the owner level.
	SY[:RWED]	Sets the read/write/extend/delete protection at the system level.

PIP SWITCHES AND SUBSWITCHES

Switch	Subswitch	Function
	WO[:RWED]	Sets the read/write/extend/delete protection at the world level.
/PU[:n[.]]		Deletes obsolete version(s) of a file.
	LD	Lists the deleted files.
RE		Renames a file.
RM		Removes a file entry from a directory.
SD		Selectively deletes files by prompting for your response before deleting.
SR		Allows shared reading of a file that has already been opened for writing by another user or task.
TD		Restricts file searches to files created on the current day.
TR		Truncates files to logical end-of-file.
UF		Creates a User File Directory entry on the volume to which a file is being transferred.
UN		Unlocks a file.
UP		Updates (rewrites) an existing file.
	FO	Specifies the owner for a file.

12.4.1 Switches

PIP accepts some switches with no file specification. However, when you use a switch in a command line, it must follow the file or directory specification. It cannot come before the device name, the directory, the filename, file type, or version of the file on which it is to operate.

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

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

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

You specify switch arguments as octal (default), decimal, or alphabetic characters, depending on the switch. The sections that explain the individual PIP switches discuss these values.

12.4.2 Subswitches

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

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

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

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

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

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

PIP renames the files PRT2.QRT and KG.MAC, but they maintain their associated version numbers. File ASDG.MAC is also renamed, but the version number is forced to a number one greater than the latest version of file ASDG.SMP (assuming a version of ASDG.SMP already exists).

When you explicitly apply a subswitch to a file specification, you implicitly apply the switch with which the subswitch is associated. On a command line with more than one file specification, the explicit subswitch affects only the file to which it is applied. The implicit switch affects all the files on the command line.

Example

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

This command is equivalent to:

```
PIP>FILE1.CMD/GR:R/WO,FILE2.MAC/GR:RW/PR
```

The command results in the following file protection:

```
a. FILE1    SYSTEM  --  Unchanged
            MEMBER  --  Unchanged
            GROUP   --  Read access
            WORLD   --  No access

b. FILE2    SYSTEM  --  Unchanged
            MEMBER  --  Unchanged
            GROUP   --  Read/write access
            WORLD   --  Unchanged
```

12.5 SPECIFYING WILDCARDS

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

The asterisk can denote zero or more characters in the field you specify it in, while the percent sign character denotes a single character in a field. (Correct syntax must be followed, however.)

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

12.5.1 Wildcards in Output File Specifications

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

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

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

```
PIP>LIST.*=[USERFILES]/LI
```

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

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

For the Rename (/RE) and Enter (/EN) switches, the output specification may have wildcards (asterisk only) mixed with specified fields. For either switch, the equivalent field of the input file specification is used.

For all cases in which wildcards are allowed in the output file specification, the wildcard directory form [*] is used to indicate that the output directory is to be the same as the input directory.

NOTE

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

12.5.2 Wildcards in Input Specifications

PIP provides the following wildcard features for input file specifications:

- *.*;* means all versions of all files.
- *.DAT;* means all versions of all files of file type .DAT.

- *.D*;* means all versions of all files with file types beginning with D.
- TEST.*;* means all versions of all types of files named TEST.
- T*.*;* means all versions of all types of files with names beginning with T.
- TEST.DAT;* means all versions of file TEST.DAT.
- TEST.D%T;* means all versions of files named TEST with three-character file types beginning with D and ending with T.
- T%N.*;* means all versions of all file types of all three-character filenames beginning with T and ending with N.
- *.* means the most recent version of all files.
- *.DAT means the most recent version of all files of file type .DAT.
- *%.DAT means the most recent version of all files that have at least one character in their names and have the file type of .DAT.
- TEST.* means the most recent version of all file types for files named TEST.

PIP also provides the following wildcard directory features:

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

NOTE

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

12.6 PIP ERROR MESSAGES

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

PIP -- ALLOCATION FAILURE - NO CONTIGUOUS SPACE

Explanation: Not enough contiguous space was available on the output volume for the file being copied.

User Action: Delete all files that are no longer required on the output volume, then reenter the command line.

PIP -- ALLOCATION FAILURE ON OUTPUT FILE

or

PIP -- ALLOCATION FAILURE - NO SPACE AVAILABLE

Explanation: Not enough space was available on the output volume for the file being copied.

User Action: Delete all files that are no longer required on the output volume, then reenter the command line.

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

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

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

PIP -- CANNOT EXCLUDE *.*;*

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

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

PIP -- CANNOT FIND DIRECTORY FILE

Explanation: The specified directory does not exist on the volume.

User Action: Reenter the command line, specifying the correct directory or the correct volume.

PIP -- CANNOT FIND FILE(S)

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

User Action: Check the file specification and reenter the command line.

PIP -- CANNOT RENAME FROM ONE DEVICE TO ANOTHER

Explanation: You attempted to rename a file across devices.

User Action: Reenter the command line, renaming the file on the input volume, then enter another command to transfer the file to the intended volume.

PIP -- CANNOT TRUNCATE THIS FILETYPE

Explanation: PIP can only truncate files containing fixed-length, variable-length, and sequenced records.

User Action: Check the file specification and reenter the command line.

PIP -- CLOSE FAILURE ON INPUT FILE

or

PIP -- CLOSE FAILURE ON OUTPUT FILE

Explanation: The input or output file could not be properly closed. If the failure is on the output file, the output file is then locked to indicate possible corruption.

User Action: Reenter the command line. If the error recurs, run a validity check of the file structure using the File Structure Verification Utility (VFY) on the volume in question to determine if it is corrupted.

PIP -- COMMAND SYNTAX ERROR

Explanation: Command did not conform to syntax rules.

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

PIP -- DEVICE NOT MOUNTED/ALLOCATED

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

User Action: Allocate the drive and/or mount the device, then reenter the command line.

PIP -- DIRECTORY WRITE PROTECTED

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

User Action: Enable the device for write operations or have the owner of the directory change its protection.

PIP -- ERROR FROM PARSE

Explanation: The specified directory file does not exist.

User Action: Reenter the command line with the correct UIC specified.

PIP -- EXPLICIT OUTPUT FILENAME REQUIRED

Explanation: Self-explanatory.

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

PIP -- FAILED TO ATTACH OUTPUT DEVICE

or

PIP -- FAILED TO DETACH OUTPUT DEVICE

Explanation: An attempt to attach/detach a record-oriented output device failed. This is usually caused by the device being off-line or nonresident.

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

PIP -- FAILED TO ATTACH TERMINAL

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

User Action: Determine the cause of the failure and correct it. Reenter the command line.

PIP -- FAILED TO CREATE OUTPUT directory

Explanation: PIP could not create an entry in a directory because the device was write-protected or because of a privilege violation.

User Action: Enable the unit for write operations or have the owner of the directory change its protection.

PIP -- FAILED TO DELETE FILE

or

PIP -- FAILED TO MARK FILE FOR DELETE

Explanation: You attempted to delete a protected file.

User Action: Change the protection of the file if possible, and reenter the command line.

PIP -- FAILED TO ENTER NEW FILE NAME

Explanation: You specified a file that already exists in the directory file, or you did not have the necessary privileges to make entries in the specified directory file.

User Action: Reenter the command line, ensuring that the filename and directory are specified correctly.

PIP -- FAILED TO FIND FILE(S)

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

User Action: Check the file specification and reenter the command line.

PIP -- FAILED TO GET TIME PARAMETERS

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

User Action: Reenter the command line. If the problem persists, submit a Software Performance Report (SPR).

PIP -- FAILED TO OPEN INDEX FILE

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

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

PIP -- FAILED TO OPEN STORAGE BITMAP FILE

Explanation: PIP could not read the specified volume's storage bitmap, probably because of a privilege violation.

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

PIP -- FAILED TO READ ATTRIBUTES

Explanation: The volume you specified was corrupted or you did not have the necessary privileges to access the file.

User Action: Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check using the File Structure Verification Utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted.

PIP -- FAILED TO REMOVE DIRECTORY ENTRY

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

User Action: Enable the unit for write operations or have the owner of the directory change its protection.

PIP -- FAILED TO RESTORE ORIGINAL DIRECTORY ENTRY - FILE IS LOST

Explanation: PIP has removed a file from a directory, failed to enter it (using /RE) into another directory, and failed to replace the original directory entry.

User Action: Run the lost check of the File Structure Verification Utility (VFY) to recover the filename.

PIP -- FAILED TO TRUNCATE FILE

Explanation: The volume you specified is corrupted or you did not have the necessary privileges (write, extend) to truncate this file.

User Action: Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check of the File Structure Verification Utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted.

PIP -- FAILED TO WRITE ATTRIBUTES

Explanation: The volume you specified is corrupted or you did not have the necessary privileges to write the file attributes.

User Action: Ensure that PIP is running under the correct UIC. If the UIC is correct, then run the validity check of the File Structure Verification Utility (VFY) against the volume in question to determine where and to what extent the volume is corrupted.

PIP -- FILE IS LOST

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

User Action: Run the lost check of the File Structure Verification Utility (VFY) to recover the filename.

PIP -- FILE NOT LOCKED

Explanation: The /UN switch was entered for a file that was not locked.

User Action: Reenter the command line, specifying the correct file.

PIP -- GET COMMAND LINE - BAD @ FILE NAME

Explanation: An illegal indirect command filename was specified.

User Action: Reenter the command line, specifying the correct name for the indirect command file.

PIP - GET COMMAND LINE - FAILED TO OPEN @ FILE

Explanation: PIP could not find the specified indirect command file.

User Action: Check the specification for the indirect command file and reenter the command line.

PIP -- GET COMMAND LINE - I/O ERROR

Explanation: An I/O error occurred during an attempt to read a command line.

User Action: Check the command to ensure that you entered it correctly, then reenter the command line. If the error persists, submit a Software Performance Report (SPR).

PIP -- GET COMMAND LINE - MAX @ FILE DEPTH EXCEEDED

Explanation: The maximum level of nesting for indirect command files (4) was exceeded.

User Action: Reduce the level of nesting.

PIP -- ILLEGAL COMMAND

Explanation: The command was not recognized by PIP.

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

PIP -- ILLEGAL EOF VALUE

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

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

PIP -- ILLEGAL RESPONSE - TRY AGAIN

Explanation: Self-explanatory.

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

PIP -- ILLEGAL SWITCH

Explanation: The specified switch was not a legal PIP switch.

User Action: Reenter the command line with the correct switch specification.

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

Explanation: You attempted to copy all versions of a file into the same directory that is being scanned for input files. This would result in an infinite number of versions of the same file, so is not allowed.

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

PIP -- ILLEGAL USE OF WILDCARD VERSION OR LATEST VERSION

Explanation: The use of either a wildcard version number or a latest version number in the attempted operation would result in inconsistent or unpredictable output.

User Action: Reenter the command line with different options or with an explicit or default version number.

PIP -- INPUT FILES HAVE CONFLICTING ATTRIBUTES

Explanation: The input files specified in a Merge, Update, or Supersede command had conflicting attributes or the attributes of the input file(s) specified in an Append command conflicted with those of the output file.

User Action: The message is a warning only. The specified action was completed despite the conflict. With a Merge, Update, or Supersede command, the attributes of the output file will be those of the first input file. With an Append command, the attributes of the output file are unchanged. The resulting file should, however, be suspect because its attributes may not correctly represent all the records in the file.

PIP -- I/O ERROR ON INPUT FILE

or

PIP -- I/O ERROR ON OUTPUT FILE

Explanation: One of the following conditions may exist:

- The device is not on-line
- The device is not mounted
- The hardware has failed
- The volume is full (output only)
- The input file is corrupted

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

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

PIP -- NOT A DIRECTORY DEVICE

Explanation: A directory-oriented command was issued to a device that does not have directories (such as a printer).

User Action: Reenter the command line without specifying a directory.

PIP -- NOT ENOUGH BUFFER SPACE AVAILABLE

Explanation: PIP did not have enough I/O buffer space to perform the requested command.

User Action: You cannot use PIP to perform the requested operation. Submit a Software Performance Report to your local DIGITAL representative.

PIP -- NO SUCH FILE(S)

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

User Action: Check the file specification and reenter the command line.

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

Explanation: A directory other than [*] was specified as the output file directory for a copy operation.

User Action: Reenter the command line with [*] specified as the output directory.

PIP -- OPEN FAILURE ON INPUT FILE

or

PIP -- OPEN FAILURE ON OUTPUT FILE

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

- The file is protected against access.

- A problem on the physical device (for example, device down).
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

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

User Action: Determine which condition caused the message and correct that condition. Reenter the command line.

PIP -- OUTPUT FILE ALREADY EXISTS -- NOT SUPERSEDED

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

User Action: Retry the copy with /NV to assign a new version number or use /SU to supersede the output file.

PIP -- TOO MANY COMMAND SWITCHES - AMBIGUOUS

Explanation: Too many switches were specified or the switches conflict.

User Action: Reenter the command line, specifying the correct set of switches.

PIP -- VERSION MUST BE EXPLICIT OR "*"

Explanation: The version number of the specified file must be expressed explicitly or as a wildcard (*).

User Action: Reenter the command line with the version number correctly expressed.

12.7 PIP ERROR CODES

Table 12-3 identifies error codes PIP issues when it cannot access the message file. The descriptions and suggested user actions are identical to those described in Section 12.6.

Table 12-3: PIP Error Codes and Messages

Code	Message
1	Command syntax error
2	Invalid switch
3	Too many command switches--ambiguous
4	Only [*,*] is legal as destination UIC
5	Invalid command
6	Invalid "*" copy to same device and directory
7	Bad use of wildcards/characters in destination filename
8	Explicit output filename 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	Invalid use of wildcard version or latest version
15	Failed to create output directory
16	Input files have conflicting attributes
17	Open failure on input file
18	Open failure on output file
19	Close failure on input file
20	Close failure on output file
21	Failed to detach output device
22	Device not mounted/allocated
23	Output file already exists--not superseded
24	Failed to mark file for delete
25	File is lost
26	Version must be explicit or "*"
27	Error from parse
28	Failed to delete file
29	Failed to attach terminal
30	Invalid response--try again
31	Cannot exclude *.*;*
32	Cannot find directory file
33	Failed to attach output device
34	Failed to get time parameters
35	Not a directory device
36	Failed to write attributes
37	Failed to read attributes
38	File not locked
39	Failed to enter new filename
40	Failed to restore original directory entry--file lost
41	Cannot rename from one device to another
42	Failed to spool file for printing

PIP ERROR CODES

Code	Message
43	Cannot spool by file ID
44	Failed to open storage bitmap file
45	Failed to open index file
46	Failed to find file(s)
47	Cannot find file(s)
48	No such file(s)
49	Failed to remove directory entry
50	Directory write protected
51	Not enough buffer space available
52	Failed to truncate file
53	Cannot truncate this filetype
54	Invalid EOF value

CHAPTER 13

OBJECT MODULE PATCH UTILITY (PAT)

The Object Module Patch Utility (PAT) allows you to update, or patch, code in a relocatable binary object module.

Input to PAT is two files, an input file and a correction file. The input file consists of one or more concatenated object modules. You can correct only one of these object modules with a single execution of PAT. The correction file consists of object code that, when linked by the Task Builder, either overlays or is appended to the input object module.

Unlike the Task Builder and ZAP patching options, PAT allows you to increase the size of the object module because the changes are applied before the module is linked by the Task Builder.

PAT uses the correction file, which contains corrections and/or additional instructions, to update the object module. Correction input is prepared in source form and then assembled by the MACRO-11 assembler.

Output from PAT is the updated input file.

Using PAT to update a file involves several steps. First, you create the correction file using a text editor. Once created, the correction file must be assembled to produce an object module. The correction file and the input file (both in object module format) are then submitted to PAT for processing.

Finally, the updated input object module is submitted to the Task Builder to resolve global symbols and to create an executable task. Figure 13-1 shows the processing steps involved in generating an updated task file using PAT.

13.1 INVOKING PAT

To invoke PAT, enter the following command:

```
$ RUN $PAT
```

PAT can be used interactively or by means of indirect command files. If you use indirect command files, PAT allows a maximum nesting level of 2.

13.2 PAT COMMAND LINE FORMAT

Specify the PAT command line in the following format:

```
[outfile]=infile[/CS[:number]],correctfile[/CS:[number]]
```

outfile

The file specification for the output file. If you do not specify an output file, PAT does not generate one.

infile

The file specification for the input file. This file can contain one or more concatenated object modules.

correctfile

The file specification for the correction file. This file contains the updates to be applied to one module in the input file.

/CS[:number]

Specifies the Checksum switch. This switch directs PAT to calculate the checksum for all the binary data that constitutes the module. PAT displays this checksum in octal. (Refer to Section 13.3.4 for information on how to use /CS.)

You can optionally specify an octal number with /CS. Then, after PAT calculates the checksum value, it compares that value with the number you specified. If the values are not the same, PAT informs you with an error message. You must then rerun PAT, specifying the correct checksum.

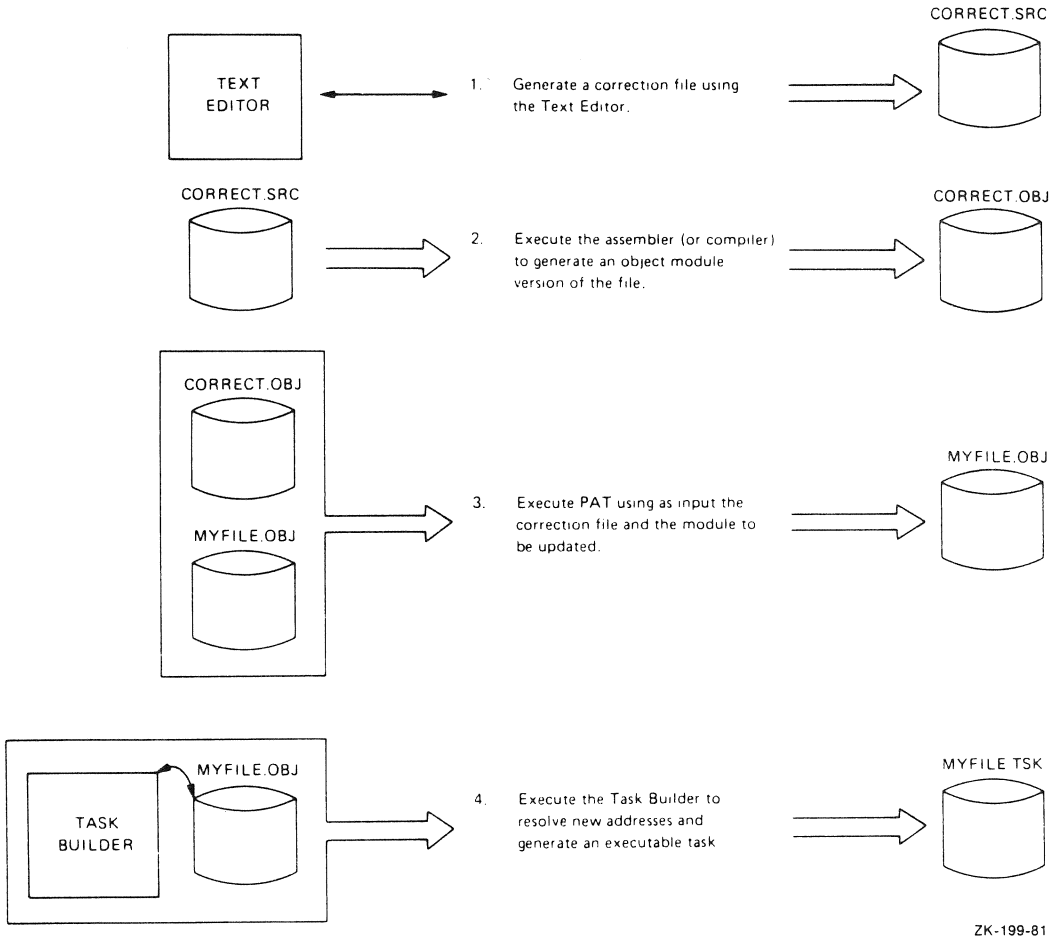


Figure 13-1: Using PAT

13.3 HOW PAT APPLIES UPDATES

This section describes the PAT input and correction files, gives information on how to create the correction file, and gives examples of how PAT applies the corrections to a module.

13.3.1 The Input File

The input file is the file to be updated; it is the base for the output file. The input file must be in object module format. When you execute PAT, the correction file is applied to one of the object modules in the file. PAT assumes a file type of .OBJ for the input file. If you use a file type other than .OBJ, you must specify it explicitly in the command line.

13.3.2 The Correction File

The correction file contains the patches to be applied to the input file. PAT assumes a file type of .OBJ for the correction file. If you use a file type other than .OBJ, you must specify it explicitly in the command line.

As shown in Figure 13-1, the first step in using PAT to update an object file is to generate the correction file. Use any text editor to create this source file, which is usually in the following format:

```
.TITLE inputname
.IDENT updatenum
inputline
inputline
    .
    .
    .
```

inputname

The name of the module to be corrected by the PAT update. You must specify the module that you are updating for inputname.

updatenum

Any value acceptable to the MACRO-11 .IDENT assembler directive. Generally, this value reflects the updated version of the file to be processed by PAT.

NOTE

The .IDENT assembler directive is a required part of the correction file. Failure to include an .IDENT directive in the file produces unusable output.

inputline

Lines of input to be used to correct and update the input file.

Once you have created the source version of the correction file, you assemble it to produce an object module that can be processed by PAT.

During PAT execution, new global symbols defined in the correction file are added to the module's symbol table. A symbol definition that is already being used in the input file can be superseded by the definition in the correction file. For a symbol definition to be superseded, both definitions must be either relocatable or absolute.

A duplicate program section supersedes the previous program section, provided:

- Both have the same relocatability attribute (ABS or REL)
- Both are defined with the same directive (.PSECT or .CSECT)

If PAT encounters duplicate program section names, the length attribute for the program section is set to the length of the longer program section and a new program section is appended to the module.

If you specify a transfer address, it supersedes the transfer address of the module being patched.

13.3.3 How PAT and the Task Builder Update Object Modules

The examples in the following sections show an input file and a correction file (both in object module format) to be processed by PAT and the Task Builder, along with a source-like representation of how the output file looks once PAT and the Task Builder complete processing. Two techniques are described: one for overlaying lines in a module, and the other for adding a subroutine to a module.

13.3.3.1 Overlaying Lines in a Module - The following example illustrates a technique using a patch file to overlay lines in a module. First, PAT appends the correction file to the input file. Then, the Task Builder generates a task image from the patched object modules.

The input file for this example is:

```
        .TITLE  ABC
        .IDENT  /01/
ABC::
        MOV     A,C
        CALL    XYZ
        RETURN
        .END
```

To add the instruction ADD A,B after the CALL instruction, you can use the following patch in the correction file:

```
        .TITLE  ABC
        .IDENT /01.01/
.=.+12
        ADD     A,B
        RETURN
        .END
```

You use the MACRO-11 assembler to assemble the correction file. After assembly, PAT processes the resulting object module and the input object module. The result of PAT processing appears as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        MOV     A,C
        CALL    XYZ
        RETURN
.=ABC
.=.+12
        ADD     A,B
        RETURN
        .END
```

You then use the Task Builder to produce the patched object module as a task image. This task image looks the same as the source code would have looked if it had originally been written as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        MOV     A,C
        CALL    XYZ
        ADD     A,B
        RETURN
        .END
```

PAT uses the .=.+12 in the program counter field to determine where to begin overlaying instructions in the program. It overlays the RETURN instruction with the patch code:

```
        ADD     A,B
        RETURN
```

13.3.3.2 Adding a Subroutine to a Module - The second example illustrates a technique for adding a subroutine to an object module. A patch often requires that more than a few lines be added to correct the file. A convenient technique for adding new code is to append it to the end of the module as a subroutine. That way, you insert a CALL instruction at an appropriate location in the subroutine. The CALL instruction directs the program to branch to the new code, execute that code, and then return to in-line processing.

The input file for this example is:

```
        .TITLE  ABC
        .IDENT  /01/
ABC::
        MOV     A,B
        CALL    XYZ
        MOV     C,R0
        RETURN
            .
            .
            .
        .END
```

The correction file for this example is:

```
        .TITLE  ABC
        .IDENT  /01.01/
        CALL    PATCH
        NOP
        .PSECT  PATCH
PATCH:
        MOV     A,B
        MOV     D,R0
        ASL     R0
        RETURN
        .END
```

PAT merges the correction file with the input file, as in the first example. The Task Builder then processes the files and produces a task image that looks the same as the source file would have looked if it had originally been written as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        CALL    PATCH
        NOP
        CALL    XYZ
        MOV     C,R0
        RETURN
            .
            .
            .
        .PSECT  PATCH
PATCH:
        MOV     A,B
        MOV     D,R0
        ASL     R0
        RETURN
        .END
```

In this example, the CALL PATCH and NOP instructions overlay the 3-word MOV A,B instruction. (The NOP is included because this is a case where a 2-word instruction replaces a 3-word instruction and NOP is required to maintain alignment.) The Task Builder allocates additional storage for .PSECT PATCH, writes the specified code into this program section, and binds the CALL instruction to the first address in this section. The MOV A,B instruction, replaced by the CALL PATCH instruction, is the first instruction executed by the PATCH subroutine.

13.3.4 Determining and Validating the Contents of a File

You use the Checksum switch (/CS) to determine or validate the contents of a module. The switch directs PAT to calculate the checksum (in octal) for all the binary data that constitutes the module and to then inform you of the checksum by means of a diagnostic message.

To determine the checksum of a file, enter the PAT command line with the /CS switch applied to that file's specification. For example:

```
=MYFILE/CS,CORRECT.POB
```

The command directs PAT to calculate the checksum for the input file, MYFILE.PAT then responds with the message:

```
INPUT MODULE CHECKSUM IS checksum
```

PAT generates a similar message when you request the checksum for the correction file. For example:

```
=MYFILE,CORRECT.POB/CS
```

After calculating the checksum for the correction file, PAT responds with the message:

```
CORRECTION INPUT FILE CHECKSUM IS checksum
```

If you specify /CS:number to validate the size of a file, PAT calculates the checksum for the file and then compares that checksum with the value you specified as number. If the two values do not match, PAT displays the following message to report the checksum error:

```
ERROR IN FILE filename CHECKSUM
```

For example, you might specify:

```
=MYFILE,CORRECT.POB/CS:432163
```

When PAT calculates the checksum for the correction file, the number is different. PAT then displays the message:

```
ERROR IN FILE CORRECT.POB CHECKSUM
```

Checksum processing always results in an octal, nonzero value.

13.4 PAT MESSAGES

PAT generates messages that state checksum values and messages that describe error conditions. For checksum values and nonfatal error messages, PAT prefixes the messages with:

```
PAT -- *DIAG*-error message
```

For messages that describe fatal errors (errors that caused PAT to terminate), PAT uses the prefix:

```
PAT -- *FATAL*-error message
```

The following messages are grouped according to message type, as follows:

- Information messages

- Command line errors
- File specification errors
- Input/output errors
- File content or format errors
- Internal software error
- Storage allocation error

13.4.1 Information Messages

The following messages describe results of checksum processing.

CORRECTION INPUT FILE CHECKSUM IS checksum

Explanation: When you specify /CS in the correction file specification, PAT informs you of the file's checksum value. The value is given in octal.

User Action: No response necessary.

INPUT MODULE CHECKSUM IS checksum

Explanation: When you specify /CS in the input file specification, PAT informs you of the file's checksum value. The value is given in octal.

User Action: No response necessary.

13.4.2 Command Line Errors

The following error messages result from failure to adhere to the command line syntax rules.

COMMAND LINE ERROR command line

Explanation: The system standard command line processor (.GCML) detected an error in the command line.

User Action: Reenter the command line using the correct information.

COMMAND SYNTAX ERROR command line

Explanation: The command line contained a syntax error.

User Action: Reenter the command line using the correct syntax.

ILLEGAL INDIRECT FILE SPECIFICATION command line

Explanation: You specified an indirect command file that contains one of the following errors:

- A syntax error
- A specification for a nonexistent indirect command file

User Action: Check for file specification syntax errors or ensure that the specified file is contained in the specified User File Directory. Reenter the command line.

MAXIMUM INDIRECT FILE DEPTH EXCEEDED command line

Explanation: In the command line, you specified an indirect command file that exceeds the maximum nesting level of 2 that is permitted by PAT.

User Action: Reorder your files so that they do not exceed PAT's nesting limit.

13.4.3 File Specification Errors

The following error messages are caused by errors in the specification of input or output files or related file switches.

CORRECTION INPUT FILE MISSING command line

Explanation: The mandatory correction file was not specified.

User Action: Reenter the command line specifying the correction file.

ILLEGAL DEVICE/VOLUME SPECIFIED device name

Explanation: The device or volume name specification contained a syntax error.

User Action: Check the rules for specifying devices and volumes, then reenter the command line using the correct syntax for the device or volume specification.

ILLEGAL DIRECTORY SPECIFICATION directory name

Explanation: The directory specification contained a syntax error.

User Action: Check the rules for specifying a directory and reenter the command line using the correct syntax for the directory specification.

ILLEGAL FILE SPECIFICATION filename

Explanation: The file specification contained a syntax error.

User Action: Reenter the command line using the correct syntax for the file specification.

ILLEGAL SWITCH SPECIFIED filename

Explanation: An unrecognized switch or switch value was specified with the file.

User Action: Check the rules for specifying the switch and reenter the command line using the correct switch or switch value.

INVALID FILE SPECIFIED filename

Explanation: You specified a file that contains one of the following errors:

- Nonexistent device

- Nonexistent directory - The directory in the filename specification does not exist on the specified device (or on the default device if no device was specified).

User Action: Reenter the command line specifying the correct device or directory.

MULTIPLE OUTPUT FILES SPECIFIED command line

Explanation: PAT accepts only one output file specification.

User Action: Reenter the command line specifying only one output file.

REQUIRED INPUT FILE MISSING command line

Explanation: The mandatory input file was not specified in the command line.

User Action: Reenter the command line specifying an input file.

TOO MANY INPUT FILES SPECIFIED command line

Explanation: Too many input files were specified in the command line. PAT accepts only the input and correction file specifications.

User Action: Reenter the command line specifying the correct files.

UNABLE TO FIND FILE filename

Explanation: PAT could not locate the specified input or correction file.

User Action: Check the directory to ensure that the file exists. Reenter the command line specifying the correct filename.

13.4.4 Input/Output Errors

The following error messages are caused by faults detected while PAT was performing I/O to the specified file.

ERROR DURING CLOSE: FILE: filename

Explanation: This error is most likely to occur while PAT is attempting to write the remaining data into the output file before deaccessing it. The most likely causes of this error are the following conditions:

- The device is full
- The device is write-locked
- A hardware error occurred

User Action: Perform the appropriate corrective action and reenter the command line: if the device is full, delete all unnecessary files; if the device is write-locked, write-enable it; if the problem is a hardware error, contact your DIGITAL Field Service representative.

ERROR POSITIONING FILE filename

Explanation: PAT attempted to position the file beyond end-of-file.

User Action: Submit a Software Performance Report along with the related console dialog and any other pertinent information.

I/O ERROR ON INPUT FILE filename

Explanation: An error was detected while PAT was attempting to read the specified input file. The principal cause of this error is a device hardware error.

User Action: Reenter the command.

I/O ERROR ON OUTPUT FILE filename

Explanation: An error occurred while PAT attempted to write into the named output file. The most likely causes of this error are the following conditions:

- The device is full

- The device is write-locked
- A device hardware error occurred

User Action: Perform the appropriate corrective action and reenter the command line: if the device is full, delete all unnecessary files; if the device is write-locked, write-enable it; if the problem is a hardware error, contact your DIGITAL Field Service representative.

13.4.5 Errors in File Contents or Format

The following errors represent inconsistencies detected by PAT in the format or contents of the input or correction files.

ERROR IN FILE filename CHECKSUM

Explanation: The checksum that PAT calculated for the named file does not match the one that you specified with /CS:number.

User Action: Ensure that you specified the correct checksum. If the checksum is correct, then you specified an invalid version of the file. Rerun PAT specifying the correct version of the file.

FILE filename HAS ILLEGAL FORMAT

Explanation: The format of the named file is not compatible with the object files produced by the standard DIGITAL language processors or accepted by the Task Builder. The principal causes are:

- Truncated input file
- Input file that consists of text

User Action: Ensure that the file is in the correct format and resubmit it for PAT processing.

INCOMPATIBLE REFERENCE TO GLOBAL SYMBOL symbol name

Explanation: The correction file contains a global symbol whose attributes do not match one or more of the following input file symbol attributes:

- Definition or reference
- Relocatable or absolute

User Action: Update the correction file by modifying the symbol attributes. Reassemble the file and resubmit it for PAT processing.

INCOMPATIBLE REFERENCE TO PROGRAM SECTION section name

Explanation: The correction file contains a section name whose attributes do not match one or both of the following input file section attributes:

- Relocatable or absolute
- Defined with the same directive (.PSECT or .CSECT)

User Action: Update the correction file by modifying the section attribute or changing the section type. Reassemble the file and resubmit it to PAT for processing.

UNABLE TO LOCATE MODULE module name

Explanation: PAT could not find the module name that was specified in the correction file in the file of concatenated input modules.

User Action: Update the input file specification to include the missing module. Reenter the command line.

13.4.6 Internal Software Error

This error reflects internal software error conditions.

ILLEGAL ERROR-SEVERITY CODE error data

Explanation: An error message call, containing an illegal parameter, has been generated.

User Action: If these messages persist, submit a Software Performance Report along with related console dialog and any other pertinent information.

13.4.7 Storage Allocation Error

The following error message indicates that not enough task memory was available for storing global symbol and program section data.

NO DYNAMIC STORAGE AVAILABLE storage-listhead

Explanation: Not enough contiguous task memory was available to satisfy a request for the allocation of storage.

PAT displays the contents of the 2-word dynamic storage listhead in octal.

User Action: You cannot use PAT to correct the input file.

UNABLE TO OPEN FILE filename

Explanation: There is insufficient work space in the internal File Storage Region (FSR) of the PAT Utility.

User Action: You cannot use PAT to correct the input file.

CHAPTER 14

FILE STRUCTURE VERIFICATION UTILITY (VFY)

The File Structure Verification Utility (VFY) for FILES-11 volumes provides the ability to:

- Check the readability and validity of a file-structured volume (default function).
- Print the number of available blocks on a file-structured volume (/FR).
- Search for files in the index file that are not in any directory; that is, files that are "lost" in the sense that they cannot be accessed by filename (/LO).
- Validate directories against the files they list (/DV).
- List all files in the index file, showing the file ID, filename, and owner (/LI).
- Perform a read check on every allocated block on a file-structured volume (/RC).

The volume to be verified must be mounted as a FILES-11 device.

There should be no other activity on the volume while VFY is executing. In particular, activities that create new files, extend existing files, or delete files should not be attempted while VFY is executing a function.

14.1 INVOKING VFY

To invoke VFY type:

```
$ RUN $VFY
```

VFY will prompt:

```
VFY>
```

14.2 VFY COMMAND FORMAT

The command line for VFY uses the format:

```
VFY>listfile,scratchdev=indev/switch
```

The parameters of this command format are:

Output Parameters

listfile

Specifies the output listing file in the following format:

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

If you do not specify a device, the default for the output listing device is the issuing terminal (TI:). The [ufd] is the UIC under which VFY is currently running. You must, however, specify the filename and file type of the output file. The default version number will be the latest version plus one.

scratchdev

Specifies the device on which the scratch file produced by VFY is to be written. This parameter is in the following format:

```
dev:
```

The scratch file is used by VFY during the verification scan and during the lost file scan. It is created but not entered in a directory. Therefore, it is transparent to you. The scratch file is automatically deleted when VFY is terminated. If you do not specify a scratch device the default device is SY0:.

If the user's default system disk is faulty or full, use this parameter to direct the scratch file to another device. The scratch file should always be assigned to a volume other than the indev volume. The scratch file is not used with the /FR and /LI switches.

Input Parameters

indev

Specifies the volume to be verified in the format dev:. If you do not specify the volume, the default is SY0:.

/switch

Specifies the function to be performed by VFY.

14.3 VFY MODE OF OPERATION

VFY normally operates in read-only mode, where the scratch file, if required, is on another device.

If the /LO switch is specified and lost files are found, VFY requires write access to [1,3], which is the directory containing lost files.

14.4 VFY VALIDITY CHECK

VFY checks the readability and validity of the volume mounted on the specified device. This function is the default function and entails reading all the file headers in the index file and ensuring that all the disk blocks referenced in the map area of each file header are allocated to that file in the volume bitmap.

The volume may be write-protected if it is not the system volume, or if the required scratch file is directed to another file-structured volume.

A validity check is specified in the following format:

```
listfile,scratchdev=indev<RET>

     or

indev<RET>
```

Example

```
$ RUN $VFY
VFY>DR0:

CONSISTENCY CHECK OF INDEX AND BITMAP ON DR0:
```

```
INDEX  INDICATES 114524. BLOCKS FREE, 17156. BLOCKS USED OUT OF 131680.
BITMAP INDICATES 114524. BLOCKS FREE, 17156. BLOCKS USED OUT OF 131680.
```

14.5 VFY SWITCHES

VFY functions are specified with switches appended to the VFY command line. The switches and their functions are summarized in Table 14-1.

Table 14-1: VFY Switches and Functions

Switch	Format	Description
Directory Validation	/DV	Validates directories against the files they list.
Free	/FR	Prints out the available space on a volume.
Identify	/ID	Identifies the VFY version. This switch may be specified on a command line by itself at any time.
List	/LI	Lists the index file by file ID.
Lost	/LO	Scans the file structure looking for files which are not in any directory.
Read Check	/RC	Checks the volume to see if every block of every file can be read.

14.5.1 Directory Validation Switch (/DV)

The Directory Validation Switch (/DV) examines each directory on the volume. (VFY considers any file on the volume with the file

type .DIR and a fixed record length of 16 bytes to be a directory.) It then reports any errors found that could be attributed to a corrupt directory or a nonexistent file listed in the directory. For example:

```
$ RUN $VFY
VFY>DX:/DV
THE FOLLOWING DIRECTORY ENTRIES WERE INVALID
[301,333] FILE ID 13,2,0 DELETED.FIL;1 - FILE NOT FOUND
[301,333] FILE ID 12345,3,0 CORRUPTED.FID;1 - FILE NOT FOUND
[301,333] FILE ID 14,2,0 GARBAGE.VER;123456 - INVALID VERSION NUMBER
[301,333] FILE ID 15,1,444 RELVOLNEZ.ERO;1 - RESERVED FIELD WAS NON-ZERO

        4. INVALID DIRECTORY ENTRIES WERE FOUND
```

Directory entries may be invalid due to the following conditions:

FILE NOT FOUND

> The file was either deleted without the corresponding directory entry being removed or the file ID field in the directory entry was corrupted. If the file does exist, it cannot be accessed with this directory entry.
>
> Remove the directory entry using the PIP /RM command.

INVALID VERSION NUMBER

> The directory entry was corrupted. If the file does exist, it cannot be accessed with this directory entry.
>
> Remove version zero of the file with the PIP /RM command.

RESERVED FIELD WAS NON-ZERO

> The third word of the file ID field in a directory entry is a reserved field and should always be zero. Remove the directory entry with PIP /RM and then reenter it with the PIP /EN command.

14.5.2 Free Switch (/FR)

The Free switch (/FR) displays the available space on a specified volume with the following message:

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

14.5.3 List Switch (/LI)

The List switch (/LI) lists the index file.

The output for each file specifies the file number, file sequence number, filename, and owner UIC, as shown in the following example:

```
VFY>DK:/LI
LISTING OF INDEX ON DK0:

FILE ID 000001,000001 INDEXF.SYS;1    OWNER [1,1]
FILE ID 000002,000002 BITMAP.SYS;1    OWNER [1,1]
FILE ID 000003,000003 BADBLK.SYS;1    OWNER [1,1]
FILE ID 000004,000004 000000.DIR;1    OWNER [1,1]
FILE ID 000005,000005 CORIMG.SYS;1    OWNER [1,1]
FILE ID 000006,000006 001001.DIR;1    OWNER [1,1]
FILE ID 000007,000007 001002.DIR;1    OWNER [1,2]
FILE ID 000010,000010 EXEMC.MLB;1     OWNER [1,1]
FILE ID 000011,000011 RSXMAC.SML;1    OWNER [1,1]
FILE ID 000012,000012 NODES.TBL;1     OWNER [1,1]
FILE ID 000013,000036 QIOSYM.MSG;311 OWNER [1,2]
FILE ID 000014,000037 F4PCOM.MSG;1    OWNER [1,2]
```

14.5.4 Lost Switch (/LO)

The Lost switch (/LO) scans the file structure looking for files that are not in any directory and cannot be referenced by filename. (VFY considers any file on the volume with the filetype .DIR and a fixed record length of 16 bytes to be a directory.) A list of the files is produced, and if the "lost file directory" [1,3] exists on that volume, the files will be entered in that directory. If an I/O error occurs, however, on a directory file operation, the files will not be entered into [1,3]. The following error message will appear:

```
FAILED TO OPEN DIRECTORY FILE
ERROR CODE -16.  - DIRECTORY [301,333]
AS A RESULT, NO FILES WILL BE ENTERED IN [1,3]
```

14.5.5 Read Check Switch (/RC)

The Read Check switch (/RC) checks to ensure that every block of every file on a specified volume can be read.

The optional parameter [:n] is the blocking factor that indicates the number of file blocks to be read at a time. The default value is the maximum number of blocks available in VFY's buffer area. The buffer area available may be increased by installing VFY in a larger partition. Four blocks are available when VFY is installed in an 8K partition, and four blocks are added for each 1K increment.

For the fastest read check, the maximum block factor should be used. Whenever an error is encountered, each block of the portion in error is reread to determine which data block(s) cannot be read.

When an error is detected, a file identification line is displayed in the following format:

FILE ID nn,nn filename.typ;ver. n blocks used/n blocks allocated

Following this line, an error message is displayed. If a blocking factor other than 1 is in use, an error message in the following form will be issued:

ERROR STARTING AT VBN n1,n2 LBN n1,n2 - ERROR CODE -n

Following the first error message, there should be one or more error messages indicating the exact block(s) in error. The second error message line(s) will be in the following form:

ERROR AT VBN n1,n2 LBN n1,n2 - ERROR CODE -n

If an ERROR STARTING AT line is displayed without one or more ERROR AT lines, a multiblock read operation on the selected device has failed, but the data blocks appear to be individually readable.

If the VBN of the unreadable block listed in the ERROR AT line is beyond the block-used-count, the data portion of the file is readable.

The negative number printed after the ERROR CODE message is usually -4 to indicate a device parity error.

14.6 FILE ERROR REPORTING

As VFY verifies a volume, error conditions are reported. Errors for a given file are preceded by a line that identifies the file in error. This line is formatted as follows:

FILE ID nn,mm filename.filetype;version OWNER [uic]

nn,mm

Represents the unique file identification number assigned to the file by the system at file-creation time.

filename

Represents the filename.

.filetype

Represents the file type (for example, .OBJ for object file).

;version

Represents the version number of the file.

[uic]

Represents the UIC for the file.

This file identification line is followed by one or more of the following messages:

I/O ERROR READING FILE HEADER-ERROR CODE -32

Explanation: VFY failed to read the file header for the specified file ID. The device is not mounted or is off-line, or the hardware has failed.

BAD FILE HEADER

Explanation: VFY checks on the validity of the file header indicate that the header has been corrupted.

MULTIPLE ALLOCATION n,m

Explanation: The specified (double-precision) logical block number is allocated to more than one file. If this error occurs, a second pass automatically occurs that indicates all files that share each multiple-allocated block. The second pass is taken after all file headers are checked.

BLOCK IS MARKED FREE n,m

Explanation: The specified logical block number is allocated to the indicated file but is not marked as allocated in the storage allocation map.

BAD BLOCK NUMBER n,m

Explanation: The specified block number was found in the header for this file but is illegal for the device (out of range). This indicates a corrupted file header.

FILE IS MARKED FOR DELETE

Explanation: A system failure occurred while the specified file was being deleted. The deletion was not completed and the file header still exists.

HEADER MAP ERROR

Explanation: VFY detected an error in the header map area that also indicates a corrupted file header.

The last error message for the file is followed by a summary line for that file, as follows:

SUMMARY: MULT=nn, FREE=nn, BAD=nn.

MULT

Specifies the number of multiple block allocations.

FREE

Specifies the number of blocks marked free that should have been allocated.

BAD

Specifies the number of errors encountered in the the map area of the file header.

If the output for VFY is directed to a terminal and you do not wish to see the error messages for a given file, enter CTRL/O. This terminates the listing of error messages for that file that is, all messages but the summary line.

14.6.1 Files Marked for Delete

If a file has been marked for delete but the deletion process was not completed, you can either restore the file, if you still need it, or you can delete the file to recover the space it was occupying. This situation only occurs when the system crashes during file processing.

14.6.1.1 Deleting a File Marked for Delete - Files that are marked for delete can be deleted directly with PIP, once their unique file ID has been obtained by doing a validity check. The file ID appears as the first entry in the file identification line that precedes each list of file errors. The following example shows how the file ID is used with PIP to delete a file:

```
$ RUN $PIP
PIP>/FI:12:20/DE
```

In this example, the file with file ID 12,20 is deleted from the system device. PIP issues the error message:

```
PIP -- FAILED TO MARK FILE FOR DELETE-NO SUCH FILE
```

since the file system denies the existence of files already marked for delete; however, the file is deleted.

14.6.2 Deletion of Multiple-Allocated Blocks

If the file structure contains multiple-allocated blocks, it is necessary to delete files until there are no such blocks. An automatic rescan of the volume identifies which files share which blocks. This rescan lists the files which contain the multiple-allocated blocks. Use this information to determine which, if any, of the files can be saved and then delete the rest with the PIP delete function.

After the files have been deleted, VFY should be run once again to ensure that all of the files containing multiple-allocated blocks have been deleted.

14.6.3 Recovering Lost Blocks

To determine whether any blocks have been lost on a file-structured volume, examine the last two lines of output from the Validity Check. The last two lines of output give the free space on the volume. The first line reports the amount of available space according to the index file (that is, the number of blocks that are not in use by any file in the index file). The second line reports the amount of available space according to the storage allocation bitmap.

If there are no errors, the two figures should agree. If the index file indicates that more blocks are free than the storage allocation bitmap indicate, then those blocks are "lost" (they appear to be allocated, but no file contains them).

14.7 VFY ERROR MESSAGES

The VFY error messages, their explanations, and suggested user actions are described below.

VFY -- COMMAND SYNTAX ERROR

Explanation: The command as entered does not conform to command syntax rules.

User Action: Reenter the command line with the correct syntax specified.

VFY -- CLOSE FAILURE ON BIT MAP

or

VFY -- CLOSE FAILURE ON INDEX FILE

or

VFY -- CLOSE FAILURE ON TEMPORARY FILE

or

VFY -- CLOSE FAILURE ON LISTING FILE

or

VFY -- I/O ERROR ON INPUT FILE

or

VFY -- I/O ERROR ON OUTPUT FILE

or

VFY -- I/O ERROR READING DIRECTORY FILE

or

VFY -- I/O ERROR WRITING FILE HEADER

or

VFY -- FAILED TO CLOSE DIRECTORY FILE

Explanation: One of the following conditions may exist:

- The device is not on-line
- The device is not mounted
- The hardware has failed

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

VFY -- FAILED TO ALLOCATE SPACE FOR TEMP FILE

Explanation: The volume specified for the temporary scratch file is full.

User Action: Use PIP to delete unnecessary files and rerun VFY, or specify another volume as the scratch device when you reenter the command line.

VFY -- FAILED TO ATTACH DEVICE

or

VFY -- FAILED TO DETACH DEVICE

Explanation: The list file specified a terminal device. VFY was not able to attach or detach the device.

User Action: Reenter the command line with a list file device that can be attached or detached.

VFY -- FAILED TO ENTER FILE

Explanation: One of the following conditions may exist:

- VFY is not running under a system UIC
- The device is not on-line.
- The device is not mounted.
- The hardware has failed.

User Action: Determine which of the conditions caused the message and correct that condition. Reenter the command line.

VFY -- FAILED TO FIND INDEXF.SYS;1 IN MFD-WILL OPEN INDEX BY FILE ID 1,1

or

VFY -- FAILED TO FIND BITMAP.SYS;1 IN MFD-WILL OPEN BITMAP BY FILE ID 2.2

Explanation: The Master File Directory has been corrupted.

User Action: Copy the disk using the BRU Utility.

VFY -- FAILED TO OPEN DIRECTORY FILE (See OPEN FAILURE error messages)

VFY -- ILLEGAL DEVICE

Explanation: The input device specified is something other than a disk or DECtape.

User Action: Reenter the command line with a mounted FILES-11 device specified.

VFY -- ILLEGAL SWITCH

Explanation: The switch specified is not a valid VFY switch or a valid switch is used illegally.

User Action: Reenter the command line with the correct switch specified.

VFY -- NO DYNAMIC MEMORY AVAILABLE - PARTITION TOO SMALL

Explanation: VFY does not have enough buffer space to run.

User Action: Run VFY in a larger partition (8K minimum).

VFY -- OPEN FAILURE ON BIT MAP

or

VFY -- OPEN FAILURE ON INDEX FILE

or

VFY -- OPEN FAILURE ON LISTING FILE

or

VFY -- OPEN FAILURE ON TEMPORARY FILE

or

VFY -- FAILED TO OPEN DIRECTORY FILE

or

VFY -- FAILED TO OPEN FILE FOR READ CHECK

Explanation: One of the following conditions may exist:

- VFY is not running under a system UIC but should be.
- The named file does not exist in the specified directory.
- The volume is not mounted.
- The specified file is read protected.
- The specified file does not exist.

User Action: Determine which of the above conditions caused the message and correct that condition. Reenter the command line.

VFY -- STORAGE CONTROL BLOCK (VBN1 of BITMAP.SYS) IS CORRUPTED

Explanation: The Storage Control Block is corrupt. This is harmless, because only VFY and PIP /FR can examine the block.

User Action: Copy the disk using the Backup and Restore Utility.

VFY -- THEY ARE STILL LOST, COULD NOT FIND DIRECTORY

Explanation: UFD [1,3] did not exist on the volume. UFD [1,3] is the "lost files" directory. VFY enters all files found by the /LO switch into this directory.

User Action: Use the DCL CREATE/DIRECTORY command to enter UFD [1,3] on the volume.

APPENDIX A

FUNCTIONS INITIATED BY DCL COMMANDS

Most DCL commands initiate functions that are actually performed by some other system task or utility. Table A-1 lists these relationships.

Table A-1: Functions Initiated by DCL Commands

DCL Command	Task/Utility Filename	Installed Task Name
ABORT	LB:[ZZPRODCL]CATCH.TSK	...CA3
APPEND	LB:[ZZPRODCL]PIP.TSK	...PIP
ASSIGN	LB:[ZZPRODCL]LCT.TSK	...MMV
ASSIGN/TASK	LB:[ZZPRODCL]CA2.TSK	...CA2
BASIC	(Language dependent)	...BP2
CANCEL	LB:[ZZPRODCL]LCT.TSK	...MMV
COBOL	(Language dependent)	...C81
CONTINUE	LB:[ZZPRODCL]LCT.TSK	...MMV
CONVERT	LB:[ZZPRODCL]RMSCNV.TSK	...CNV
COPY	LB:[ZZPRODCL]PIP.TSK	...PIP
CREATE	LB:[ZZPRODCL]PIP.TSK	...PIP
CREATE/DIR	LB:[ZZPRODCL]LCT.TSK	...MMV

FUNCTIONS INITIATED BY DCL COMMANDS

DCL Command	Task/Utility Filename	Installed Task Name
DEASSIGN	LB:[ZZPRODCL]LCT.TSK	...MMV
DEFINE	LB:[ZZPRODCL]LCT.TSK	...MMV
DELETE	LB:[ZZPRODCL]PIP.TSK	...PIP
DIBOL	(Language dependent)	...DIB
DIFFERENCES	LB:[ZZPRODCL]CMP.TSK	...CMP
DIRECTORY (Local files)	LB:[ZZPRODCL]PIP.TSK	...PIP
DIRECTORY (Remote files)	LB:[ZZDECNET]NFT.TSK	...NFT
DIRECTORY/ATTRIBUTES	LB:[ZZPRODCL]RMSDSP.TSK	...DSP
DISMOUNT	LB:[ZZPRODCL]LCT.TSK	...MMV
DUMP	LB:[ZZPRODCL]DMP.TSK	...DMP
EDIT/EDT	LB:[ZZPRODCL]EDT.TSK	...EDT
EDIT/SLP	LB:[ZZPRODCL]SLP.TSK	...SLP
EDIT/PROSE (Starts the editor)	LB:[ZZPRODCL]DCLPROSE.TSK	...PRO
EDIT/PROSE (PROSE editor)	LB:[ZZSYS]CET.TSK	CET
FORMAT	LB:[ZZPRODCL]LCT.TSK	...MMV
FORTRAN	(Language dependent)	...F77
HELP	LB:[ZZPRODCL]HELP.TSK	...HLP
INITIALIZE	LB:[ZZPRODCL]LCT.TSK	...MMV
INSTALL	LB:[ZZPRODCL]LCT.TSK	...MMV
LIBRARY	LB:[ZZPRODCL]LBR.TSK	...LBR
LINK	LB:[ZZPRODCL]PAB.TSK	...PAB

FUNCTIONS INITIATED BY DCL COMMANDS

DCL Command	Task/Utility Filename	Installed Task Name
LINK/C81 (Invokes ...BLD, ...PAB, and ...PIP)	(Language dependent)	...BLD
LOAD	LB:[ZZPRODCL]LCT.TSK	...MMV
MACRO	LB:[ZZPRODCL]PMA.TSK	...PMA
MOUNT	LB:[ZZPRODCL]LCT.TSK	...MMV
PASCAL	(Language dependent)	...PAS
PURGE	LB:[ZZPRODCL]PIP.TSK	...PIP
REMOVE	LB:[ZZPRODCL]LCT.TSK	...MMV
RENAME	LB:[ZZPRODCL]PIP.TSK	...PIP
RUN (If the task is installed, DCL runs the task.)	LB:[ZZPRODCL]LCT.TSK	...MMV
SET [DAY]TIME	LB:[ZZPRODCL]CA2.TSK	...CA2
SET DEFAULT	LB:[ZZPRODCL]LCT.TSK	...MMV
SET DEVICE	LB:[ZZPRODCL]CA2.TSK	...CA2
SET PRIORITY	LB:[ZZPRODCL]CA2.TSK	...CA2
SET PROTECTION	LB:[ZZPRODCL]PIP.TSK	...PIP
SET TERMINAL	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW ASSIGNMENTS (Show a specific logical)	LB:[ZZPRODCL]LCT.TSK	...MMV
SHOW ASSIGNMENTS (Show all logicals)	LB:[ZZPRODCL]LNBDMP.TSK	...SLG
SHOW CLOCK_QUEUE	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW COMMON	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW [DAY]TIME	LB:[ZZPRODCL]CA2.TSK	...CA2

FUNCTIONS INITIATED BY DCL COMMANDS

DCL Command	Task/Utility Filename	Installed Task Name
SHOW DEFAULT	LB:[ZZPRODCL]LCT.TSK	...MMV
SHOW LOGICALS (Show a specific logical)	LB:[ZZPRODCL]LCT.TSK	...MMV
SHOW LOGICALS (Show all logicals)	LB:[ZZPRODCL]LNBDMP.TSK	...SLG
SHOW DEVICES	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW MEMORY	LB:[ZZPRODCL]RMD.TSK	...RMD
SHOW TASKS	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW TASKS/ACTIVE	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW TASKS/DYNAMIC	LB:[ZZPRODCL]RMD.TSK	...RMD
SHOW TASKS/INSTALLED	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW TASKS/LOGICAL_UNIT	LB:[ZZPRODCL]CA2.TSK	...CA2
SHOW TERMINAL	LB:[ZZPRODCL]CA2.TSK	...CA2
START	LB:[ZZPRODCL]LCT.TSK	...MMV
START/UNBLOCK	LB:[ZZPRODCL]CATCH.TSK	...CA3
STOP/BLOCK	LB:[ZZPRODCL]CATCH.TSK	...CA3
TYPE	LB:[ZZPRODCL]PIP.TSK	...PIP
UNLOAD	LB:[ZZPRODCL]LCT.TSK	...MMV
UNLOCK	LB:[ZZPRODCL]PIP.TSK	...PIP

APPENDIX B

ERROR MESSAGES

This appendix lists two categories of error messages: general error messages and I/O error messages.

NOTE

> Many system utilities issue error messages; these messages are explained elsewhere in the documentation set.

B.1 GENERAL ERROR MESSAGES

General error messages are common to many DCL commands. They can appear on your terminal screen preceded by a 3-letter code identifying the system component that detected the error. This can be the name of the utility (task) performing the command, or the command itself. This code appears here as yyy.

Most of the general error messages with the word "expected" in them reprint the command on your terminal with a circumflex (^) pointing to the error. Sometimes the circumflex points to the character just past the last successfully parsed command element.

Many of the explanations refer to the Radix-50 character set. The Radix-50 characters are the uppercase alphabet, the numerals 0 through 9, the dollar sign ($), and the period (.).

yyy -- Allocation failure - no contiguous space

> **Explanation:** Not enough contiguous space is available on the output volume for the file being copied.
>
> **User Action:** Delete any files no longer required on the output volume and retry the command.

yyy -- Allocation failure on output file

or

yyy -- Allocation failure -- no space available

Explanation: Not enough space is available on the output volume for the file being copied.

User Action: Delete any files no longer required on the output volume and retry the command.

yyy -- A-Z expected

Explanation: The command as typed included a nonalphabetic character.

User Action: Check command for proper syntax and reenter.

yyy -- A-Z and/or 0-9 expected

Explanation: The command as typed included a nonalphanumeric character.

User Action: Check command for proper syntax and reenter.

yyy -- Bad error message

Explanation: Some unusual condition has caused an error.

User Action: Record the command that caused the error and other information on activity at your terminal at the time. Then call your DIGITAL Customer Support Center.

yyy -- Bad use of wild cards in destination file name

Explanation: A wildcard (*) was specified for an output file where it is not permitted.

User Action: Reenter the command with a complete and explicit file specification for the output file.

yyy -- Cannot find directory file

Explanation: The command specified a directory not found on the current volume.

User Action: Reenter the command after checking for the correct directory and correct volume.

yyy -- Cannot find files

Explanation: The file or files specified in the command are not in the designated directory.

User Action: Check the file specification and reenter the command line.

yyy -- Cannot rename from one device to another

Explanation: The command attempted to rename a file across devices.

User Action: Use the COPY command to move the file from one device to another and rename it.

yyy -- Cannot truncate this filetype

Explanation: The command attempted to truncate a file that cannot be truncated. Only files containing fixed-length, variable-length, or sequenced records can be truncated.

User Action: Check to see if you have named the proper file and retry the command.

yyy -- Close failure on input file

or

yyy -- Close failure on output file

Explanation: A file name in the command could not be properly closed. The file is locked by PIP.

User Action: Use UNLOCK to unlock the file. Determine the cause of the error and correct it if you can.

yyy -- Command function not unique

Explanation: The command as typed did not include sufficient characters to identify some command function.

User Action: Retype command after checking proper syntax.

yyy -- Command line incomplete

Explanation: The command as typed is not a complete command.

User Action: Retype command after checking proper syntax.

yyy -- Command syntax error

Explanation: The command did not conform to the syntax rules.

User Action: Check command for proper syntax and reenter it.

yyy -- Conflicting qualifier

Explanation: The command as typed included qualifiers that conflict with each other in their effect.

User Action: Retype command after checking proper syntax.

yyy -- Contradictory qualifier

Explanation: The command as typed included contradictory qualifiers, such as /DELETE and /NODELETE.

User Action: Check for proper syntax and reenter command.

yyy -- Contradictory qualifier in key specification

Explanation: The command included a contradictory qualifier in the key definition argument to the /KEY qualifier.

User Action: Check command for proper syntax and reenter.

yyy -- Decimal number expected

Explanation: The command included a number not in proper format.

User Action: Check command for proper syntax and reenter with a decimal point (.) terminating the number.

yyy -- Device invalid or not specified

Explanation: The command specified an invalid device or no device at all when a device name is required.

User Action: Check the devices on the system with SHOW DEVICES. Reenter command after checking for proper syntax. Determine the cause of the error and correct it if you can.

yyy -- Device not in system

Explanation: The command specified a device that is not in the current system.

User Action: Check the devices on the system with SHOW DEVICES. Reenter command after checking for proper syntax.

yyy -- Device not mounted/allocated

Explanation: The command specified a device that is not properly mounted or allocated for the command to execute.

User Action: Check the status of the device with SHOW DEVICES. Find the cause of the error and correct it if you can.

yyy - Device not terminal

Explanation: The command specified a device other than a terminal where a terminal device name is required. TI: is not an acceptable terminal device name in all contexts.

User Action: Reenter the command after checking for proper syntax.

yyy -- Directory write protected

Explanation: The command attempted to remove an entry from a directory that is privileged or from a directory on a device that is write-protected.

User Action: Determine the cause of the error and correct it if you can. You may need to enable write access through the device hardware or change the protection for the directory.

yyy -- Error dispatching command. DSW = 'n'

Explanation: An error occurred that was not explicitly handled by DCL or some invoked task.

User Action: Look up the DSW error code in the *P/OS System Reference Manual*. Determine the cause of the error and correct it if you can.

yyy -- Explicit output file name required

Explanation: The command requires an explicit output filename.

User Action: Reenter the command line in proper syntax and without wildcards.

yyy -- Extraneous input

Explanation: The command as typed included extraneous input. The circumflex (^) points to the error or just past the last successfully parsed command element.

User Action: Reenter the command line in proper syntax.

yyy -- Failed to attach output device

or

yyy -- Failed to detach output device

Explanation: An attempt to attach or detach a record-oriented output device, such as a terminal or line printer, failed. This error usually means the device is off-line or nonresident.

User Action: Determine the cause of the error and correct it if you can.

yyy -- Failed to create output UFD

Explanation: The command failed to create an entry in a directory because the device was write-protected or because of a privilege violation.

User Action: Determine the cause of the error and correct it if you can. You may need to enable write access through the device hardware or change the protection for the directory.

yyy -- Failed to delete file

or

yyy -- Failed to mark file for delete

Explanation: The command attempted to delete a protected file.

User Action: Check for the proper file specification and default directory and reenter the command.

yyy -- Failed to enter new filename

Explanation: The command specified a file that already exists in the directory, or the directory is protected.

User Action: Check for proper syntax; reenter command.

yyy -- Failed to find files

Explanation: The command specified a file or files that could not be found as specified.

User Action: Check for proper file specifications and reenter the command.

yyy -- Failed to get time parameters

Explanation: An internal system problem has occurred.

User Action: Retry the command. If the error recurs, record the command that caused the error and other information on activity at your terminal at the time. Then call your DIGITAL Customer Support Center.

yyy -- Failed to read attributes

Explanation: The command specified a volume that is either corrupted or protected against access.

User Action: You may be able to correct the error by making your defaults the same as the device and directory of the file you wish to affect.

yyy -- Failed to remove directory entry

Explanation: The command attempted to remove an entry from a directory that was either protected against access or on a write-protected device.

User Action: Determine the cause of the error and correct it if you can. You may need to enable write access through the device hardware or change the protection for the directory.

yyy -- Failed to truncate file

Explanation: The command specified a volume that is corrupted or is protected against access.

User Action: You may be able to correct the error by making your defaults the same as the device and directory of the file you wish to affect.

yyy -- Failed to write attributes

Explanation: The command specified a volume that is corrupted or is protected against access.

User Action: You may be able to correct the error by making your defaults the same as the device and directory of the file you wish to affect.

yyy -- Fatal I/O error

Explanation: The command failed to execute because of some I/O error. This error can be caused by the unavailability of a device or of pool space, or by a device error. The device may be write-locked.

User Action: Determine the cause of the error and correct it if you can.

yyy -- File is lost

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

User Action: Type @VERIFY to run the lost-file check to recover the filename.

yyy -- Filename or filetype not specified

Explanation: The command as typed did not clearly specify a filename and file type where one or the other or both is required. This error can be caused if you do not leave a space in front of a file specification that is a parameter.

User Action: Retype command after checking for proper syntax.

yyy -- File not locked

Explanation: An UNLOCK command specified a file that is not locked.

User Action: Check for the proper file specification and reenter the command.

yyy -- File specification either invalid or not specified

Explanation: The system could not read a file specification included in the command. This error often results from a typing mistake or typing the command in the wrong format.

User Action: Check for proper syntax and reenter the command.

yyy -- File specification list not available for RMS-11

Explanation: A command to an RMS-11 utility included more than one input file specification.

User Action: Check for proper syntax and reenter command.

yyy -- File version number not specified

Explanation: The command requires a file version number to be specified.

User Action: Reenter command after checking for proper syntax.

yyy -- Function not unique

Explanation: The command as typed did not include sufficient characters to identify some function.

User Action: Retype the command but include more characters.

yyy -- Get command line - Bad @ filename

Explanation: The command specified an illegal indirect command file.

User Action: Check for proper file specification and reenter the command.

yyy -- Illegal command

Explanation: The command, which is the first word on the command line, is not part of DCL.

User Action: Check command for proper syntax and reenter.

yyy -- Illegal device

Explanation: The command named a device in an illegal format or contained some other syntax error.

User Action: Check command for proper syntax and reenter. Device names are two alphabetical characters followed by an octal number and a colon.

yyy -- Illegal filespec

Explanation: The command required a file specification that was not present.

User Action: Check command for proper syntax and reenter. See Chapter 2 for a complete description of a file specification. Perhaps some other command element is being parsed as a filespec. Use the prompting version.

yyy -- Illegal or contradictory qualifier

Explanation: One or more qualifiers to the command are in conflict, or are in error.

User Action: Check command for proper syntax and reenter.

yyy -- Illegal protection code

Explanation: The command specified a protection code in an improper format.

User Action: Check command for proper syntax and reenter.

yyy -- Illegal qualifier value

Explanation: The command as typed included an improper argument to a qualifier.

User Action: Check command for proper syntax and reenter. In DCL, an argument is preceded by a colon (:).

yyy -- Illegal use of wildcard character

Explanation: The command included a wildcard (* or %) in a file specification in a way that would result in unpredictable or inconsistent output.

User Action: Check command for proper syntax and reenter. You may not be able to use the wildcard.

yyy -- Illegal task name

Explanation: The command named a task using a task name in an illegal format.

User Action: Check command for proper syntax and reenter. Task names include as many as six Radix-50 characters.

yyy -- Input device must be a directory device

Explanation: The command as typed specified a device that is not a directory device, such as a printer. Directory devices are those on which FILES-11 volumes with directories can be mounted.

User Action: Correct syntax and reenter command.

yyy -- Input files have conflicting attributes

Explanation: Warning message. The command operation completed, but the files named had conflicting attributes.

User Action: Use DIRECTORY/ATTRIBUTES to find the attributes of all input and output files involved. Determine if the conflict causes any difficulty.

yyy -- Invalid command function

Explanation: The command as typed requested a function that is not valid for that command.

User Action: Check command for proper syntax and reenter it.

yyy -- Invalid command parameter

Explanation: The command as typed included a parameter that is not valid. In DCL, a parameter is either entered in response to a prompt or preceded by a space.

User Action: Check command for proper syntax and reenter.

yyy -- Invalid file specification list

Explanation: The command included a list of file specifications in an invalid format. In general, file specifications in lists should be separated by commas and, optionally, blanks.

User Action: Retype command after checking proper syntax.

yyy -- Invalid file specification qualifier

Explanation: The command included a qualifier to a file specification that was not valid. In DCL, a qualifier is preceded by a slash (/).

User Action: Check for proper syntax and retype the command.

yyy -- Invalid terminal specified

Explanation: A command directed to a specific terminal named the terminal in an improper format.

User Action: Check the name of the terminal using SHOW DEVICE and reenter the command with the proper format.

yyy -- Invalid time or date

Explanation: The command specified a clock or calendar field, or both, incorrectly.

User Action: Check for proper syntax and retype the command.

yyy -- Invalid UIC specified

or

yyy -- Invalid directory specified

Explanation: The command specified:

- A named character containing a nonalphanumeric string or more than nine characters, or
- A numbered directory containing 0 or a number that includes 8 or 9 or a number greater than 377 (octal).

User Action: Retype command after checking for proper syntax.

yyy -- I/O error on input file

or

yyy -- I/O error on output file

Explanation: One of the following conditions exists:

- The device is not on line.
- The device is not mounted.
- The hardware has failed.
- The output volume is full.
- The input file is corrupted.

User Action: Determine the cause of the error and correct it if you can.

yyy -- Key position size or number not specified

Explanation: The command failed to include the size or number of a key position in the key definition argument to the /KEY qualifier.

User Action: Check command for proper syntax and retype it.

yyy -- Key specification out of sequence

Explanation: The command included improper syntax in the key definition argument to the /KEY qualifier.

User Action: Check command for proper syntax and reenter.

yyy -- More command parameters than permitted

Explanation: The command as typed included too many parameters.

User Action: Check command for proper syntax and reenter.

yyy -- No such file

Explanation: The command requested operations on a file that does not exist.

User Action: Make sure you have named the file properly. Check your defaults to be sure you are looking in the right directory on the right device. You may have made a typing error. If the desired file is in fact not present, find out why it is not present and proceed accordingly.

yyy -- Not a directory device

Explanation: A directory-oriented command named a device that does not have directories, such a line printer.

User Action: If you can, reenter the command without specifying a directory.

yyy -- Numeral expected

or

yyy -- Numeral required

Explanation: Command included nonnumeric characters in a position where numerals are required or expected.

User Action: Check command for proper syntax and reenter.

yyy -- Octal number expected

Explanation: The command included a number with an 8 or 9 where an octal number was expected.

User Action: Check command for proper syntax and reenter.

yyy -- Open failure on file

Explanation: The system could not open a required file for some reason.

User Action: Check the directory to be sure that the file is present, not locked, and in the proper format. If the file is locked, it may be corrupted or contain bad data. Determine the cause of the error and correct it if you can.

yyy -- Output device must be a directory device

Explanation: The command as typed specified a device that is not a directory device, such as a printer. Directory devices are those on which FILES-11 volumes with directories can be mounted.

User Action: Correct syntax and reenter command.

yyy -- Primary key not specified

Explanation: A command affecting an indexed file failed to include a primary key. In many such commands, you cannot specify any action on an alternate key without first identifying the primary key.

User Action: Check for proper syntax and reenter command.

yyy -- Qualifier inconsistent with compiler

Explanation: A compiler command included a qualifier not acceptable by that compiler.

User Action: Check the command for proper syntax and try it again.

yyy -- Qualifier not available for this command format

Explanation: The command included a qualifier that is invalid in the current context of the command. The context is usually determined by some other qualifier in the command.

User Action: Reenter command after checking for proper syntax.

yyy -- Qualifier not unique

Explanation: The command as typed did not include sufficient characters to identify some qualifier. In DCL, qualifiers are preceded by a slash (/).

User Action: Retype the command, but include more characters.

yyy -- Qualifier value invalid here

Explanation: The commands as typed included an inappropriate argument to a qualifier. In DCL, an argument is usually preceded by a colon (:).

User Action: Check command for proper syntax and reenter.

yyy -- Radix-50 expected

Explanation: The command included a non-Radix-50 character where Radix-50 required.

User Action: Reenter the command using proper syntax. The Radix-50 characters are the uppercase alphabet, the numbers 0 through 9, the dollar sign ($), and the period (.).

yyy -- Repeated command parameter

Explanation: The command as typed included one parameter more than once. In DCL, a parameter is preceded by a blank or prompt.

User Action: Retype command after checking for proper syntax.

yyy -- Repeated key specification

Explanation: The command as typed included the same key specification more than once in the key definition argument to the /KEY qualifier.

User Action: Retype command after checking for proper syntax.

yyy -- Repeated keyword in key specification

Explanation: The command as typed included the same keyword more than once in the key definition argument to the /KEY qualifier.

User Action: Retype command after checking for proper syntax.

yyy -- Repeated qualifier

Explanation: The command specified the same qualifier more than once. In DCL, a qualifier is preceded by a slash (/).

User Action: Reenter command after checking the syntax to see if it is right.

yyy -- Required parameter not specified

Explanation: The command cannot execute without required parameters, such as a file specification or attribute. In DCL, a parameter is preceded by a blank or prompt.

User Action: Retry the command, using the prompts. DCL prompts for all required parameters.

yyy -- Required qualifier not specified

Explanation: The command requires a qualifier that it does not include. In DCL, a qualifier is preceded by a slash (/).

User Action: Check for proper syntax and reenter the command.

yyy -- Required qualifier value not specified

Explanation: A qualifier to the command requires that you state a numerical argument. In DCL, an argument is usually preceded by a colon (:).

User Action: Retype command after checking for proper syntax.

yyy -- Required value not specified for position size or number

Explanation: The command failed to include a required value in the key definition argument to the /INDEXED qualifier.

User Action: Check command for proper syntax and reenter.

yyy -- Sorry, line too long

Explanation: DCL commands are translated for execution by a system task or utility. This error is caused by a translated command line that the destination task cannot handle.

User Action: Check command syntax to see if you are specifying elements that can be defaulted. Check to see if you can enter the command twice with different qualifiers instead of entering one command.

yyy -- Sorry, low pool

Explanation: The command could not execute because of insufficient space in the system pool (dynamic storage region).

User Action: The pool is the Executive's data base. In general, each task, including commands, uses a certain amount of pool. If the pool is full or badly fragmented, there may not be sufficient space for the command to execute. Usually, pool problems clear up spontaneously if you wait.

Retry the command after an interval. You should not attempt to execute any other task, not even an ABORT, when the pool is low. The task also absorbs pool.

yyy -- Sorry, task active

Explanation: The command required some action that cannot be taken on an active task.

User Action: Determine the cause of the error and correct it if you can. Check the task with the various SHOW TASK commands, or SHOW MEMORY.

yyy -- Sorry, task not installed

Explanation: The command attempted to invoke a task that was not installed.

User Action: Try running the task with a command in the RUN $ form. Determine the cause of the error and correct it if you can.

yyy -- Syntax error

Explanation: The command included some error in typing or specification, such as a letter where a number should appear.

User Action: Check for proper syntax and reenter command.

yyy -- Version must be explicit or "*"

Explanation: The command syntax requires that the version number of the file must be specified explicitly or as a wildcard (*).

User Action: Reenter the command with the version number correctly expressed.

yyy -- Wildcards not permitted

Explanation: Command included a wildcard (* or %) in a context where it is not permitted.

User Action: Check for proper syntax and reenter command.

yyy -- Zero value not valid for key size or number

Explanation: The command included a zero value in the key definition argument to the /KEY qualifier.

User Action: Check command for proper syntax and reenter.

B.2 I/O ERROR MESSAGES

I/O error messages are returned as codes. Table B-1 lists P/OS I/O error codes. Only partial abbreviations (xxx) are listed; the complete abbreviation is IE.xxx. The octal number listed is the low-order byte of the complete value (2's complement of the decimal number).

Table B-1: I/O Error Messages

Abbre-viation	--Error Decimal	No.-- Octal	Meaning
.BAD	-1	377	Bad parameters
.IFC	-2	376	Invalid function code
.DNR	-3	375	Device not ready
.VER	-4	374	Parity error on device
.ONP	-5	373	Hardware option not present
.SPC	-6	372	Invalid user buffer
.DNA	-7	371	Device not attached
.DAA	-8	370	Device already attached
.DUN	-9	367	Device not attachable
.EOF	-10	366	End-of-file detected
.EOV	-11	365	End-of-volume detected
.WLK	-12	364	Write attempted to locked unit
.DAO	-13	363	Data overrun
.SRE	-14	362	Send/receive failure
.ABO	-15	361	Request terminated
.PRI	-16	360	Privilege violation
.RSU	-17	357	Shareable resource in use
.OVR	-18	356	Invalid overlay request
.BYT	-19	355	Odd byte count or virtual address
.BLK	-20	354	Logical Block Number too large
.MOD	-21	353	Invalid UDC module number
.CON	-22	352	UDC connect error
.NOD	-23	351	System dynamic memory
.DFU	-24	350	Device full
.IFU	-25	347	Index file full
.NSF	-26	346	No such file
.LCK	-27	345	Locked from read-write access
.HFU	-28	344	File header full
.WAC	-29	343	Accessed for write
.CKS	-30	342	File header checksum failure
.WAT	-31	341	Attribute control list format error
.RER	-32	340	File processor device read error
.WER	-33	337	File processor device write error
.ALN	-34	336	File already accessed on LUN
.SNC	-35	335	File ID, file number check
.SOC	-36	334	File ID, sequence number check
.NLN	-37	333	No file accessed on LUN

Abbre-viation	--Error Decimal	No.-- Octal	Meaning
.CLO	-38	332	File was not properly closed
.NBF	-39	331	File buffer space unavailable
.RBG	-40	330	Invalid record size
.NBK	-41	327	File exceeds space allocated, no blocks
.ILL	-42	326	Invalid operation on file descriptor block
.BTP	-43	325	Bad record type
.RAC	-44	324	Invalid record access bits set
.RAT	-45	323	Invalid record attribute bits set
.RCN	-46	322	Invalid record number-too large
.ICE	-47	321	Internal consistency error
.2DV	-48	320	Rename--two different devices
.FEX	-49	317	Rename--a new filename already in use
.BDR	-50	316	Bad directory syntax
.RNM	-51	315	Cannot rename old file system
.BDI	-52	314	Bad directory syntax
.FOP	-53	313	File already open
.BNM	-54	312	Bad filename
.BDV	-55	311	Bad device name
.BBE	-56	310	Bad block on device
.DUP	-57	307	Enter-duplicate entry in directory
.STK	-58	306	Not enough stack space (FCS or FCP)
.FHE	-59	305	Fatal hardware error on device
.NFI	-60	304	File ID was not specified
.ISQ	-61	303	Invalid sequential operation
.EOT	-62	302	End-of-tape detected
.BVR	-63	301	Bad version number
.BHD	-64	300	Bad file header
.OFL	-65	277	Device off line
.BCC	-66	276	Block check, CRC, or framing error
.ONL	-67	275	Device on line
.NNN	-68	274	No such node
.NFW	-69	273	Path lost to partner
.BLB	-70	272	Bad logical buffer
.TMM	-71	271	Too many outstanding messages
.NDR	-72	270	No dynamic space available
.URJ	-73	267	Connection rejected by user

Abbre-viation	--Error Decimal	No.-- Octal	Meaning
.BTF	-76	264	Bad tape format
.NNC	-77	263	Not ANSI 'D' format byte count
.NNL	-78	262	Not a network LUN
.NLK	-79	261	Task not linked to specified ICS/ICR interrupts
.NST	-80	260	Specified task not installed
.AST	-80	260	No AST specified in connect
.FLN	-81	257	Device off line when off-line request issued
.IES	-82	256	Invalid escape sequence
.PES	-83	255	Partial escape sequence
.ALC	-84	254	Allocation failure
.ULK	-85	253	Unlock error
.WCK	-86	252	Write check failure
.CNR	-96	240	Connection rejected

APPENDIX C

SAMPLE EDT INITIALIZATION FILE

The Edit command initializes EDT using a default EDT initialization file. The complete file specification for this initialization file is:

```
LB000:[1,2]EDTSYS.EDT
```

You can edit this initialization file to include commands that are similar to those found in the example in this appendix (refer to the *EDT Quick Reference Guide* for further information on creating EDT initialization files).

You can maintain one or more EDT initialization files in directories other than [1,2]. However, you must use the /COM qualifier in the EDT command line to use any command file not found in the default directory.

A sample initialization file follows.

```
SET SCREEN 79    ! Sets screen width to 79 characters.

SET CURSOR 2:15 ! EDT will scroll file, if cursor is moved before line
                ! 2 or after line 15.

SET WRAP 70      ! Sets the maximum line length of n character
                 ! positions (70 in this example). As you enter
                 ! text, EDT will attempt to wrap a full word that
                 ! extends beyond character position 70.

SET NOTRUNCATE   ! Displays lines that extend beyond the line width set
                 ! by SET SCREEN 79.

SET TAB 3        ! Sets the first tab stop at character position 3; all
                 ! other tab stops are set at multiples of 8 after the
                 ! first tab stop.
```

```
DEFINE KEY GOLD         E AS "EXT EXIT."

                        ! Defines the keystroke sequence GOLD + E
                        ! as the EXIT command in Change mode.

DEFINE KEY GOLD         Q AS "EXT QUIT."

                        ! Defines the keystroke sequence GOLD + Q as
                        ! the QUIT command (used only in Change mode).

DEFINE KEY CONTROL      R AS "+DC ADV C UNDC C."

                        ! This keystroke sequence
                        ! reverses the position of two letters.

SET MODE CHANGE         ! Causes EDT to begin the editing session in
                        ! Change mode.
```

INDEX

Abbreviated form of DCL commands, 2-1
Abbreviations, 2-7
ABORT, 3-2
  background processing, 2-15
  definition, 2-16
Active task display, 8-8
Adding lines to file, 10-12
Adding subroutine to module, 13-7
Addressing locations in files, 9-5
Addressing modes
  ZAP, 9-5
Alphabetical PRO/Tool Kit DCL command descriptions, 3-1
Altering
  active task display, 8-8
  RMD memory display, 8-7
  task header display, 8-10
APPEND, 3-4
Application
  refer to program development cycle
Applying updates, 13-3
Arguments
  in DCL command line, 2-2
.ASK, 4-28
.ASKN, 4-30
.ASKS, 4-33
ASSIGN, 3-8
ASSIGN/TASK, 3-12
Associated documents, iv
Attaching terminal in background mode
  caution against, 2-16
Audit trail
  controlling, 10-19
  deleting, 10-23
  length, 10-20
  position, 10-20
  suppressing, 10-22
  value, 10-21

Background processing, 2-15
BASIC, 3-13
.BEGIN, 4-35
CANCEL, 3-14
Cancellation
  of DCL command, 2-9
.CHAIN, 4-36
Change bar format, 5-8
CLEAR, 3-15
.CLOSE, 4-37
.CMD extension, 4-1
CMP, 5-1
  change bar format, 5-8
  command format, 5-2
  differences format, 5-7
  invoking, 5-2
  messages, 5-9
  output files, 5-6
  switches, 5-3
COBOL, 3-16
Command
  ABORT, 3-2
  APPEND, 3-4
  ASSIGN, 3-8
  ASSIGN/TASK, 3-12
  BASIC, 3-13
  CANCEL, 3-14
  CLEAR, 3-15
  COBOL, 3-16
  CONTINUE, 3-23
  CONVERT, 3-24
  COPY, 3-30
  CREATE, 3-37
  CREATE/DIRECTORY, 3-40
  DEASSIGN, 3-42
  DEBUG, 3-44
  DEFINE, 3-45
  DELETE, 3-49
  DELETE/DIRECTORY, 3-52
  DIBOL, 3-53
  DIFFERENCES, 3-57
  DIRECTORY, 3-61
  DISMOUNT, 3-65
  DUMP, 3-66
  EDIT, 3-71
  EDIT/EDT, 3-71
  EDIT/PROSE, 3-74
  EDIT/SLP, 3-75
  EXIT, 3-78
  FIX, 3-79

FORMAT, 3-81
FORTRAN, 3-82
HELP, 3-88
INITIALIZE, 3-89
INSTALL, 3-91
LIBRARY, 3-93
LIBRARY/COMPRESS, 3-94
LIBRARY/CREATE, 3-96
LIBRARY/DELETE, 3-99
LIBRARY/EXTRACT, 3-100
LIBRARY/INSERT, 3-101
LIBRARY/LIST, 3-103
LIBRARY/REMOVE, 3-105
LIBRARY/REPLACE, 3-106
LINK, 3-108
LINK/C81, 3-123
LOAD, 3-126
MACRO, 3-127
MAIL, 3-135
MOUNT, 3-136
PASCAL, 3-137
PHONE, 3-140
PRINT, 3-141
PRINT/REMOTE, 3-142
PURGE, 3-143
REMOVE, 3-146
RENAME, 3-147
RUN installed task, 3-152
RUN uninstalled task, 3-150
SET DAYTIME, 3-155
SET DEFAULT, 3-156
SET DEVICE, 3-157
SET HOST, 3-159
SET PRIORITY, 3-160
SET PROTECTION, 3-161
SET PROTECTION/DEFAULT, 3-164
SET PROTECTION/NODEFAULT, 3-166
SET TERMINAL, 3-167
SET TIME, 3-155
SHOW ASSIGNMENTS, 3-172
SHOW CLOCK_QUEUE, 3-173
SHOW COMMON, 3-174
SHOW DAYTIME, 3-176
SHOW DEFAULT, 3-177
SHOW DEVICES, 3-178
SHOW LOGICALS, 3-179
SHOW MEMORY, 3-180
SHOW PROTECTION, 3-181
SHOW TASK/DYNAMIC, 3-188
SHOW TASK/LOGICAL_UNITS, 3-191
SHOW TASKS/ACTIVE, 3-182
SHOW TASKS/ACTIVE/DYNAMIC, 3-190
SHOW TASKS/INSTALLED, 3-186
SHOW TERMINAL, 3-192
SHOW TIME, 3-176
SPAWN, 3-193
START, 3-194
START/UNBLOCK, 3-195
STOP/BLOCK, 3-196
SUBMIT/REMOTE, 3-197
TYPE, 3-198
UNLOAD, 3-201
UNLOCK, 3-202
Command files, 4-1
Command line
  DCL, 2-1
Command line continuation, 2-8
Commands
  file manipulation, 1-2
  manipulating Tasks, 1-2
  miscellaneous, 1-3
  program development, 1-2
  set and show, 1-3
Comments
  comment character, 2-8
Compare utility
  see CMP
Compatibility
  with RSX command files, 4-93
Compound tests, 4-25
CONTINUE, 3-23
Controlling audit trail, 10-19
CONVERT, 3-24
Convert Utility, 11-1
COPY, 3-30
Correction file, 13-4
CREATE, 3-37
CREATE/DIRECTORY, 3-40
Creating source files with SLP, 10-17
CTRL/C
  definition, 2-15
  foreground processing, 2-15
CTRL/Z, 2-9
  to cancel DCL command, 2-9
CVT, 11-1

.DATA, 4-38
DCL
  abbreviations, 2-7
  ABORT command use, 2-16

  background processing, 2-15
  colon use, 2-7
  command abbreviations, 2-1
  command format, 2-2
  command line comments, 2-8
  command line continuation, 2-8
  command line errors, 2-9
  command line format, 2-1
  command line terminators, 2-2
  CTRL/C use, 2-15
  equal sign use, 2-7
  errors, 2-9
  file specifications, 2-13
  foreground processing, 2-15
  functional groups, 1-1
  HELP, 2-5
  initialization, 2-10
  logical name translation, 2-15
  prompting, 2-2
  qualifiers, 2-3
  Single Line Editor, 2-11
  termination, 2-10
  wildcard features in, 2-13
)EASSIGN, 3-42
)EBUG, 3-44
.DEC, 4-39
)efault file type, 4-1
)EFINE, 3-45
)efine logical EOF, 4-27
.DELAY, 4-40
)ELETE, 3-49
)ELETE/DIRECTORY, 3-52
)eleting file marked for delete, 14-10
)eleting lines from file, 10-14
)eletion of multiple-allocated blocks, 14-10
)escription
  of PRO/Tool Kit Utilities, 1-4
)evelopment of the application, iii
)evice and file manipulation, 1-1
  also see Chapter 3
)IBOL, 3-53
)IFFERENCES, 3-57
)ifferences format, 5-7
)IRECTORY, 3-61
)irectory validation, 14-4
.DISABLE, 4-41
)ISMOUNT, 3-65
)MP, 6-1
  command format, 6-2
  error messages, 6-11
  examples, 6-8
  header dump, 6-10
  invoking, 6-2
  multiple format dump, 6-8
  record dump, 6-9
  switches, 6-3
DO
  command line terminator, 2-2
Document conventions, v
Document structure, iii
DUMP, 3-66

EDIT, 3-71
EDIT/EDT, 3-71
  see EDT
EDIT/PROSE, 3-74
EDIT/SLP, 3-75
Editors
  text, 1-4
EDT command file
  directory location, 3-71
  use with /COMMAND qualifier, 3-71
Effect of qualifier position in command line, 2-4
.ENABLE, 4-42
.END, 4-46
Entry point table, 7-3
.ERASE, 4-47
Error
  detection by system, 2-9
Error messages
  indirect, 4-95
Errors
  DCL Command Line, 2-9
.EXIT, 4-49
EXIT, 3-78
Exit
  from PRO/Tool Kit, 2-10
Expressions, 4-16, 4-18

File Compare Utility, 5-1
File contents determination, 13-8
File contents validation, 13-8
File Dump Utility, 6-1
File manipulation commands, 1-2
File specifications, 2-13
File Structure Verification Utility, 14-1

Files marked for delete, 14-9
/FINAL qualifier
  with ASSIGN, 3-9
  with DEFINE, 3-46
FIX, 3-79
Foreground processing, 2-15
.FORM, 4-50
FORMAT, 3-81
Format
  PIP commands, 12-2
Format of library files, 7-2
Formats of CMP output files, 5-6
FORTRAN, 3-82
/FR, 14-5
FREE command, 8-7
Functional groups
  DCL commands, 1-1
Functions initiated by DCL
    commands, A-1

general error messages, B-1
.GOSUB, 4-56
.GOTO, 4-57

Header dump, 6-10
HELP, 2-5, 3-88
  as a separate command, 2-5
  while using a DCL command, 2-5
Hyphen
  in command line, 2-8

I/O error messages, B-19
.IF, 4-25, 4-58
IFACT, 4-60
IFDF, 4-61
.IFDISABLED, 4-63
.IFENABLED, 4-63
.IFF, 4-65
.IFINS, 4-62
.IFLOA, 4-64
.IFNACT, 4-60
.IFNDF, 4-61
.IFNINS, 4-62
.IFNLOA, 4-64
.IFT, 4-65
.INC, 4-66
Indirect
  .ASK, 4-28
  .ASKN, 4-30
  .ASKS, 4-33
  .BEGIN, 4-35
  .CHAIN, 4-36
  .CLOSE, 4-37
  command files, 4-1
  command processor, 4-2
  .DATA, 4-38
  .DEC, 4-39
  define logical EOF, 4-27
  .DELAY, 4-40
  directive summary, 4-4
  .DISABLE, 4-41
  .ENABLE, 4-42
  .END, 4-46
  .ERASE, 4-47
  error messages, 4-95
  .EXIT, 4-49
  expressions, 4-16, 4-18
  .FORM, 4-50
  .GOSUB, 4-56
  .GOTO, 4-57
  .IF, 4-25, 4-58
  .IFACT, 4-60
  .IFDF, 4-61
  .IFDISABLED, 4-63
  .IFENABLED, 4-63
  .IFF, 4-65
  .IFINS, 4-62
  .IFLOA, 4-64
  .IFNDF, 4-61
  .IFNINS, 4-62
  .IFNLOA, 4-64
  .IFT, 4-65
  .INC, 4-66
  information only messages, 4-94
  .LABEL, 4-67
  messages, 4-94
  numeric symbols, 4-16
  .ONERR, 4-68
  .OPEN, 4-69
  .OPENA, 4-70
  .OPENR, 4-71
  .PARSE, 4-72
  .PAUSE, 4-73
  .READ, 4-74
  reserved symbols, 4-19
  .RETURN, 4-76
  RSX compatibilty, 4-93
  .SETD, 4-79
  .SETF, 4-77
  .SETL, 4-77
  .SETN, 4-78
  .SETO, 4-79

.SETS, 4-80
.SETT, 4-77
special symbols, 4-9
logical, 4-9
numeric, 4-11
string, 4-14
.STOP, 4-82
string symbols, 4-18
substitution format, 4-20
substrings, 4-18
switches, 4-21
symbol substition, 4-19
symbols, 4-8
.TEST, 4-83
.TESTDEVICE, 4-85
.TESTFILE, 4-86
.TESTPARTITION, 4-87
.TESTSYSTEM, 4-88
.TRANSLATE, 4-90
using directives, 4-24
.WAIT, 4-91
.XQT, 4-92
Indirect Command Processor
refer to Chapter 4
use with PRO/Tool Kit DCL commands, 2-1
Indirect Command Processor, 4-1
INITIALIZE, 3-89
Input file wildcards, 12-11
INSTALL, 3-91
Installation of application, iii
Intended audience, iii
Interactive
program development, 1-1
program execution and control, 1-1
Interactive use of PRO/Tool Kit DCL, 2-1
Invoking
CMP, 5-2
CVT, 11-1
DMP, 6-2
LBR, 7-11
PAT, 13-2
PIP, 12-1
RMD, 8-2
SLP, 10-2
VFY, 14-1
ZAP, 9-7

.LABEL, 4-67
LB:[1,2]EDTSYS.EDT
see EDT use with /COMMAND qualifier
LBR, 7-1
/CO, 7-15
combining library functions, 7-42
compress switch, 7-15
/CR, 7-16
create switch, 7-16
/DE, 7-18
default switch, 7-19
delete global switch, 7-21
delete switch, 7-18
/DF, 7-19
/DG, 7-21
entry point switch, 7-23
entry point table, 7-3
/EP, 7-23
error messages, 7-43, 7-44
/EX, 7-25
extract switch, 7-25
fatal errors, 7-44
file specifier defaults, 7-11
/FU, 7-29
/IN, 7-26, 7-27
insert switch, 7-26, 7-27
invoking, 7-11
/LE, 7-29
/LI, 7-29
library file format, 7-2
library header, 7-2
list switches, 7-29
/MH, 7-30
modify header switch, 7-30
module header, 7-3
module name table, 7-3
replace switch, 7-31, 7-37
restrictions, 7-10
/RP, 7-31, 7-37
selective search switch, 7-39
squeeze switch, 7-40
/SS, 7-39
switches, 7-13
/SZ, 7-40
/LI, 14-6
Librarian Utility Program, 7-1
LIBRARY, 3-93
Library header, 7-2
LIBRARY/COMPRESS, 3-94
LIBRARY/CREATE, 3-96

LIBRARY/DELETE, 3-99
LIBRARY/EXTRACT, 3-100
LIBRARY/INSERT, 3-101
LIBRARY/LIST, 3-103
LIBRARY/REMOVE, 3-105
LIBRARY/REPLACE, 3-106
LINK, 3-108
LINK/C81, 3-123
/LO, 14-6
LOAD, 3-126
Locations
  addressing
    in files, 9-5
/LOG qualifier
  with PRINT/REMOTE, 3-142
Logical name translation, 2-15
Lost block recovery, 14-10

MACRO, 3-127
MAIL, 3-135
Manual objectives, iii
Miscellaneous commands, 1-3
Module name table, 7-3
MOUNT, 3-136
Multiple format dump, 6-8

Nesting command files, 4-1
/NOWARNINGS qualifier
  with PRINT/REMOTE, 3-142
Numeric symbols, 4-16

Object Module Patch Utility, 13-1
Object modules, 13-5
.ONERR, 4-68
.OPEN, 4-69
.OPENA, 4-70
.OPENR, 4-71
Output file wildcards, 12-10
Overlaying lines, 13-5
OWNER command, 8-9

Parameters
  in DCL command line, 2-2
.PARSE, 4-72
PASCAL, 3-137
PAT, 13-1
  adding subroutine, 13-7
  applying updates, 13-3
  command line errors, 13-10
  command line format, 13-2
  correction file, 13-4
  errors in file contents, 13-15
  errors in file format, 13-15
  file contents determination, 13-8
  file contents validation, 13-8
  file specification errors, 13-11
  I/O errors, 13-13
  information messages, 13-10
  input file, 13-4
  internal software error, 13-16
  invoking, 13-2
  messages, 13-9
  module, 13-5, 13-7
  object modules, 13-5
  overlaying lines, 13-5
  storage allocation error, 13-17
  updates, 13-3
  updating object modules, 13-5
  validation, 13-8
.PAUSE, 4-73
Peripheral Interchange Program, 12-1
PHONE, 3-140
PIP, 12-1
  command format, 12-2
  error codes, 12-23
  error messages, 12-13
  file specification fields, 12-3
  input file wildcards, 12-11
  invoking, 12-1
  output file wildcards, 12-10
  specifying wildcards, 12-10
  subswitches, 12-5, 12-9
  switches, 12-5, 12-8
PRINT, 3-141
PRINT/REMOTE, 3-142
PRIORITY command, 8-9
PRO/TK
  Refer to PRO/Tool Kit
PRO/Tool Kit, 1-1
  environment, 1-1
  introduction, 1-1
  origins, 1-1
  requirements to run, 1-1
  utilities description, 1-4
PRO/Tool Kit DCL
  Refer to PRO/Tool Kit Command Language
  use with Indirect, 2-1
PRO/Tool Kit DCL commands

  alphabetical
    description of, 3-1
Processing
  background
    use of ABORT command, 2-15
  foreground
    use of CTRL/C, 2-15
Program development commmands, 1-2
Prompting, 2-2
  examples of, 2-2
  when invoking DCL, 2-2
PURGE, 3-143

Qualifier
  /SHOW, 3-55
Qualifiers
  definition of, 2-3

Radix, 11-2
RATE command, 8-7, 8-9, 8-11
/RC, 14-6
.READ, 4-74
Record dump, 6-9
REMOVE, 3-146
RENAME, 3-147
Replacing lines in file, 10-16
Reserved symbols, 4-19
Resource Monitoring Display, 8-1
.RETURN, 4-76
RETURN
  command line terminator, 2-2
RMD, 8-1
  active task display, 8-8
  altering active task display, 8-8
  altering memory display, 8-7
  altering task header display, 8-10
  display pages, 8-1
  error messages, 8-11
  FREE command, 8-7
  help display, 8-3
  invoking, 8-2
  memory display, 8-4
  OWNER command, 8-9
  PRIORITY command, 8-9
  RATE command, 8-7, 8-9, 8-11
  running on second terminal, 8-3
  setup pages, 8-1
  TASK command, 8-9, 8-11
  task header display, 8-9
RSX
  compatibilty, 4-93
RSX-11M-PLUS, 1-1
RUN installed task, 3-152
RUN uninstalled task, 3-150
Running RMD on second terminal, 8-3

/SESSION qualifier
  with ASSIGN, 3-9
  with DEFINE, 3-46
Set and Show commands, 1-3
SET DAYTIME, 3-155
SET DEFAULT, 3-156
SET DEVICE, 3-157
SET HOST, 3-159
SET PRIORITY, 3-160
SET PROTECTION, 3-161
SET PROTECTION/DEFAULT, 3-164
SET PROTECTION/NODEFAULT, 3-166
SET TERMINAL, 3-167
SET TIME, 3-155
.SETD, 4-79
.SETF, 4-77
.SETL, 4-77
.SETN, 4-78
.SETO, 4-79
.SETS, 4-80
.SETT, 4-77
SHOW ASSIGNMENTS, 3-172
SHOW CLOCK_QUEUE, 3-173
SHOW COMMON, 3-174
SHOW DAYTIME, 3-176
SHOW DEFAULT, 3-177
SHOW DEVICES, 3-178
SHOW LOGICALS, 3-179
SHOW MEMORY, 3-180
SHOW PROTECTION, 3-181
/SHOW qualifier
  arguments to, 3-55
SHOW TASK/DYNAMIC, 3-188
SHOW TASK/LOGICAL_UNITS, 3-191
SHOW TASKS/ACTIVE, 3-182
SHOW TASKS/ACTIVE/DYNAMIC, 3-190
SHOW TASKS/INSTALLED, 3-186
SHOW TERMINAL, 3-192
SHOW TIME, 3-176
SLE
  see DCL Single Line Editor
SLP, 10-1

adding lines to file, 10-12
audit trail
controlling, 10-19
deleting, 10-23
length, 10-20
position, 10-20
suppressing, 10-22
value, 10-21
command input, 10-3
command input format, 5-8
controlling, 10-18
controlling audit trail, 10-19
creating source files with, 10-17
deleting lines from file, 10-14
edit commands, 10-6, 10-8
error messages, 10-24
file processing, 10-5
information message, 10-24
input files, 10-2
interactive commands, 10-9
invoking, 10-2
listing, 10-11
listing file, 10-4
messages, 10-24
numbered listing, 10-11
operators, 10-10
output files, 10-2, 10-4
replacing lines in file, 10-16
source files, 10-11
switches, 10-18
through Indirect, 10-10
updating source files, 10-11
using, 10-6
Source files, 10-11
Source Language Input Program, 10-1
SPAWN, 3-193
Special logical symbols, 4-9
Special numeric symbols, 4-11
Special string symbols, 4-14
Special symbols, 4-9
Specifying wildcards, 12-10
START, 3-194
START/UNBLOCK, 3-195
Status bits
task, 3-174
.STOP, 4-82
STOP/BLOCK, 3-196
String symbols, 4-18
Structure of DCL command line, 2-1
SUBMIT/REMOTE, 3-197
Substitution format control, 4-20
Substrings, 4-18
Summary of indirect directives, 4-4
Symbol value substitution, 4-19
Symbols, 4-8
/SYSTEM qualifier
with ASSIGN, 3-9
with DEFINE, 3-46

TASK command, 8-9, 8-11
Task header display, 8-9
Task Manipulation Commands, 1-2
Task status bits, 3-174
Task/File Patch Program, 9-1
Terminal
attaching in background mode, 2-16
.TEST, 4-83
.TESTDEVICE, 4-85
.TESTFILE, 4-86
.TESTPARTITION, 4-87
.TESTSYSTEM, 4-88
.TRANSLATE, 4-90
Translation of logical names, 2-15
/TRANSLATION_ATTRIBUTES
with ASSIGN, 3-9
with DEFINE, 3-46
TYPE, 3-198

UNLOAD, 3-201
UNLOCK, 3-202
Updating object modules, 13-5
Updating source files, 10-11
/USER qualifier
with ASSIGN, 3-9
with DEFINE, 3-46
Using indirect directives, 4-24
Utilities
form, 1-4
frame, 1-4
program development, 1-4
Utility
invocation of
CMP, 5-2
CVT, 11-1
DMP, 6-2

LBR, 7-11
PAT, 13-2
PIP, 12-1
RMD, 8-2
SLP, 10-2
VFY, 14-1
ZAP, 9-7

Validation, 13-8
Validity check, 14-3
VAX/VMS, 1-1
Verify utility
see VFY
VFY, 14-1
command format, 14-2
deleting file marked for delete, 14-10
deletion of multiple-allocated blocks, 14-10
directory validation, 14-4
/DV, 14-4
error messages, 14-11
file error reporting, 14-7
files marked for delete, 14-9
/FR, 14-5
free switch, 14-5
invoking, 14-1
/LI, 14-6
list switch, 14-6
/LO, 14-6
lost block recovery, 14-10
lost switch, 14-6
mode of operation, 14-3
/RC, 14-6
read check switch, 14-6
switches, 14-4
validity check, 14-3

.WAIT, 4-91
Wildcard, 2-13
Wildcards, 12-10

.XQT, 4-92

ZAP, 9-1
absolute addressing, 9-7
absolute location, 9-18
addressing locations, 9-5
addressing modes, 9-5
arithmetic operators, 9-11
block number, 9-13
branch target location, 9-19
byte offset, 9-13
changing location contents, 9-16
close command, 9-14
close location, 9-9
closing file locations, 9-17
command line, 9-8
commands, 9-9
commands, general purpose, 9-20
current location symbol, 9-13
equal sign command, 9-22
error messages, 9-31
examples, 9-24
generap purpose commands, 9-10
I- and D-space, 9-5
internal registers, 9-10
K command, 9-21
/LI, 9-3
list switch, 9-3
location specifier, 9-12
multiuser task image, 9-4
O command, 9-22
offset location, 9-18
open command, 9-14
open location, 9-9
opening file locations, 9-16
opening preceding location, 9-17
operating modes, 9-2
previous location, 9-19
R command, 9-23
regular task image, 9-3
relocation bias, 9-5
relocation register, 9-14
resident libraries, 9-4
RETURN key, 9-10
separators in command line, 9-12
switches, 9-2
task image addressing, 9-7
termination of, 9-7
through Indirect, 9-8
V command, 9-23
X command, 9-20

PRO/Tool Kit
Command Language
and Utilities Manual
AA-X912B-TH

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized?
Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country

Please cut along this line.

-- Do Not Tear - Fold Here and Tape ----------

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Professional Workstations Publications
DIGITAL EQUIPMENT CORPORATION
146 Main Street, MLO21-2/T76
Maynard, Massachusetts 01754-2571

--- Do Not Tear - Fold Here ----------

Cut Along Dotted Line