

VMS User's Manual

Order Number: AA-LA98B-TE

June 1989

This manual describes tasks you can perform using the VMS operating system. The information contained in this manual is intended for all users and is applicable to all members of the VAX and MicroVAX families of computers running the VMS operating system.

Revision/Update Information: This manual supersedes the *VMS General User's Manual, Version 5.0*.

Software Version: VMS Version 5.2

**digital equipment corporation
maynard, massachusetts**

June 1989

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

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


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

© Digital Equipment Corporation 1989.

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

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

CDA	MASSBUS	VAX RMS
DDIF	PrintServer 40	VAXstation
DEC	Q-bus	VMS
DECnet	ReGIS	VT
DECUS	ULTRIX	XUI
DECwindows	UNIBUS	
DIGITAL	VAX	
LN03	VAXcluster	

The following is a third-party trademark:

PostScript is a registered trademark of Adobe Systems, Inc.

ZK4323

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by Digital. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use Digital-supported devices, such as the LN03 laser printer and PostScript printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

Contents

Preface

xxix

Chapter 1 Introduction: VMS Concepts and Definitions

1.1	How to Use This Manual	1-1
1.2	Logging In to the System	1-2
1.3	Using a Network	1-2
1.4	The DIGITAL Command Language (DCL)	1-2
1.4.1	The DCL Command Line	1-3
1.5	Files and Directories	1-4
1.5.1	File and Directory Specifications	1-4
1.5.2	Directory Structures	1-5
1.6	Devices	1-6
1.6.1	Physical Device Names	1-6
1.6.2	Logical Device Names	1-7
1.6.3	Generic Device Names	1-7
1.7	Processes	1-7
1.8	Programs	1-8
1.9	Utilities	1-8
1.9.1	MAIL	1-9
1.9.2	VMS SORT/MERGE	1-9
1.10	Text Editors	1-9
1.11	DIGITAL Standard Runoff (DSR)	1-9
1.12	Logical Names	1-10
1.13	Symbols	1-10
1.14	Command Procedures	1-11

1.15	Account and System Security	1-11
------	-----------------------------------	------

Chapter 2 Getting Started: Interacting with VMS

2.1	Logging In to the System	2-1
2.2	Logging In to a Remote Node	2-2
2.3	Changing Your Password	2-3
2.4	Recognizing System Responses	2-4
2.5	Getting Help	2-5
2.6	Ending a Remote Session	2-7
2.7	Logging Out of the System	2-7

Chapter 3 The DIGITAL Command Language: Communicating with VMS

3.1	Using DCL Commands	3-1
3.2	Constructing a DCL Command	3-3
3.2.1	Vocabulary of a DCL Command	3-3
3.2.2	Putting the Parts in Order: Syntax	3-4
3.3	Entering a DCL Command	3-5
3.3.1	Rules for Entering a DCL Command	3-5
3.3.2	Entering an Incomplete Command Line	3-6
3.3.3	Entering a Command Longer Than One Line ..	3-6
3.3.4	Entering Parameters	3-7
3.3.5	Entering Qualifiers	3-8
3.4	Recalling Commands	3-10
3.5	Entering Dates and Times as Values	3-12
3.5.1	Absolute Time	3-12
3.5.2	Delta Time	3-13
3.5.3	Combination Time	3-14
3.6	Defining Terminal Keys	3-15
3.7	Summary of Key Combinations	3-15

Chapter 4 Files: Storing Information

4.1	Understanding File Names and Specifications	4-2
4.2	Using Wildcards with Files	4-4
4.2.1	The Asterisk (*) Wildcard Character	4-5
4.2.2	The Percent (%) Wildcard Character	4-5
4.3	Creating and Modifying Files	4-5
4.3.1	Creating Files	4-5
4.3.2	Copying Files	4-6
4.4	Renaming Files	4-7
4.5	Displaying the Contents of Files	4-7
4.6	Deleting Files	4-8
4.7	Protecting a File from Other Users	4-9
4.7.1	Default File Protection	4-9
4.7.2	Explicit File Protection	4-10
4.8	Printing Files	4-11
4.8.1	Displaying Queue Information	4-11
4.8.2	Stopping and Deleting a Print Job	4-12
4.8.3	Printing a File on Another Node	4-12
4.8.4	DCL Commands That Control Print Jobs	4-12

Chapter 5 Directories: Organizing and Managing Files

5.1	Understanding Directory Structures	5-1
5.2	Understanding Directory Names and Specifications	5-3
5.3	Creating Directories	5-3
5.4	Displaying Directories	5-3
5.5	Setting a Default Directory	5-4
5.6	Deleting Directories	5-5
5.7	Protecting a Directory from Other Users	5-6
5.8	Using Wildcards to Search the Directory Structure	5-6
5.8.1	The Ellipsis (. . .) Wildcard Character	5-7
5.8.2	The Hyphen (-) Wildcard Character	5-8

Chapter 6 Editing Text Files: Using EVE

6.1	Beginning an Editing Session	6-1
6.2	Using Online Help	6-2
6.3	Ending an Editing Session	6-2
6.3.1	Saving Your Edits	6-3
6.3.2	Ending the Session Without Saving Your Edits	6-3
6.4	Entering EVE Commands	6-3
6.4.1	Using Defined Keys to Enter EVE Commands	6-4
6.4.2	Typing EVE Commands	6-7
6.4.3	EVE Key Names	6-7
6.5	Editing Text	6-8
6.5.1	Locating Text	6-9
6.5.2	Replacing Text	6-11
6.5.3	Recovering from System Interruptions	6-12
6.5.4	Refreshing the Screen	6-12
6.5.5	Using the Journal File	6-12
6.5.6	Listing Buffers	6-14
6.5.7	Editing Two Buffers	6-14
6.5.8	Reading and Writing Files	6-14
6.5.9	EVE Default Settings	6-15
6.6	Saving Time and Keystrokes—Defining Keys in EVE	6-18
6.6.1	Using EVE to Emulate EDT	6-18
6.6.2	Using EVE to Emulate WPS	6-21
6.6.3	Defining a Key While Using EVE	6-22
6.6.4	Using Startup Files to Define Keys	6-25
6.7	Using DCL Within EVE	6-32
6.7.1	Executing a DCL Command	6-33
6.7.2	Creating a Subprocess	6-33
6.8	Converting from EDT to EVE	6-33
6.9	EVE Command Summary	6-37

Chapter 7 Editing Text Files: Using EDT

7.1	Invoking and Ending an EDT Session	7-1
7.1.1	Invoking EDT	7-1
7.1.2	Ending an EDT Session	7-2
7.2	Entering EDT Commands	7-3
7.2.1	Entering EDT Line Commands	7-3
7.2.2	Entering Keypad Commands	7-3
7.2.3	Canceling EDT Commands	7-5
7.3	Getting HELP in EDT	7-5
7.3.1	Getting HELP with Keypad-Editing Commands	7-5
7.3.2	Getting HELP with Line-Editing Commands	7-5
7.4	Changing Editing Modes	7-5
7.4.1	Changing from Keypad to Line Editing	7-6
7.4.2	Changing from Line to Keypad Editing	7-6
7.4.3	Entering Line-Editing Commands from Keypad Mode	7-6
7.5	Recovering from Interruptions	7-7
7.6	EDT Keypad Editing	7-8
7.6.1	Manipulating the Cursor	7-8
7.6.2	Inserting Text	7-13
7.6.3	Deleting and Restoring Text	7-14
7.6.4	Locating Text	7-17
7.6.5	Substituting Text	7-19
7.6.6	Moving Text	7-21
7.6.7	Moving Text Within the File	7-21
7.6.8	Using Multiple Buffers	7-25
7.7	Saving Time and Keystrokes—Defining Keys in EDT	7-27
7.7.1	Defining Keys While in EDT	7-27
7.7.2	Advanced Key Definitions	7-31
7.7.3	Permanent Key Definitions	7-34
7.7.4	Summary	7-41
7.8	Controlling EDT Sessions	7-42
7.8.1	Controlling Screen Format with SET Commands	7-42
7.8.2	Controlling Editing Functions with SET Commands	7-43
7.8.3	Defining EDT Macros	7-44

Chapter 8 MAIL: Communicating with Other Users

8.1	Invoking and Exiting MAIL	8-2
8.2	Reading Messages	8-2
8.2.1	Reading a New Message	8-3
8.2.2	Reading Old Messages	8-3
8.3	Sending a Message	8-4
8.3.1	Sending MAIL over the Network	8-5
8.3.2	Sending a Message to More Than One User ...	8-5
8.3.3	Sending a File	8-7
8.3.4	Creating a File from a Message	8-8
8.4	Replying to a Message	8-9
8.5	Forwarding a Message	8-9
8.6	Organizing Your Messages	8-9
8.6.1	Creating and Modifying Folders	8-9
8.7	Selecting Folders	8-10
8.8	Deleting Messages	8-11
8.9	Customizing Your MAIL Environment	8-12
8.9.1	Creating a Mail Subdirectory	8-12
8.9.2	Using the Mail Keypad	8-12
8.9.3	Using a Text Editor in MAIL	8-13

Chapter 9 VMS SORT/MERGE: Sorting and Merging Files

9.1	Sorting Records	9-1
9.2	Sorting Character Data Files	9-3
9.3	Sorting Noncharacter Data Files	9-4
9.4	Entering Records from a Terminal	9-4
9.5	Submitting Batch Jobs	9-5
9.6	Merging Files	9-5

Chapter 10 Processes: Using the VMS Environment

10.1	Interpreting Your Process Context	10-1
10.2	Using Subprocesses	10-3
10.2.1	Creating a Subprocess	10-4
10.2.2	Exiting from a Subprocess	10-5
10.2.3	Looking at a Subprocess Context	10-6
10.3	Executing Programs Across the Network	10-7
10.4	Using Batch Jobs	10-7
10.4.1	Submitting a Batch Job	10-7
10.4.2	Batch Job Output	10-8
10.4.3	Restarting a Batch Job	10-9

Chapter 11 Logical Names: Defining Names for Devices and Files

11.1	Creating Logical Names	11-1
11.1.1	Rules for Creating Logical Names	11-2
11.1.2	Equating More Than One Equivalence Name ..	11-3
11.2	Displaying Logical Names	11-3
11.3	Deleting Logical Names	11-4
11.4	Understanding Logical Name Tables	11-4
11.4.1	The Process Table	11-5
11.4.2	The Job Table	11-6
11.4.3	The Group Table	11-6
11.4.4	The System Table	11-7
11.5	Directory Logical Name Tables	11-8
11.5.1	The Process Directory Table	11-8
11.5.2	The System Directory Table	11-9
11.6	Logical Name Access Modes	11-11
11.7	Creating a Logical Name Table	11-12
11.8	Using Search Lists	11-13
11.9	Using Logical Node Names	11-14
11.10	System-Created Logical Names	11-15
11.10.1	Process-Permanent Logical Names	11-15
11.10.2	System-Permanent Logical Names	11-18

Chapter 12 Symbols: Defining Commands and Expressions

12.1	Using Symbols to Represent DCL Commands	12-1
12.2	Using Symbols to Collect, Store, and Manipulate Data	12-3
12.2.1	Defining Symbols as Character Strings	12-3
12.2.2	Creating Symbols	12-4
12.2.3	Understanding Symbol Tables	12-5
12.2.4	Understanding Symbol Substitution	12-6
12.2.5	Using Symbol Values	12-7
12.2.6	Using Symbols in Expressions	12-11

Chapter 13 Command Procedures: Programming with DCL

13.1	Formatting a Command Procedure	13-2
13.2	Executing a Command Procedure	13-2
13.2.1	Changing Command Procedure Levels	13-4
13.2.2	Exiting from a Command Procedure	13-5
13.3	Designing a Login Command Procedure	13-5
13.4	Using Loops	13-7
13.5	Passing Data	13-9
13.5.1	Using Parameters to Pass Data	13-10
13.5.2	Using the INQUIRE Command	13-13
13.5.3	Using the READ Command	13-14
13.5.4	Obtaining Data from SYS\$INPUT	13-14
13.6	Returning Data	13-15
13.7	Displaying Data	13-16
13.7.1	Displaying Character Strings and Symbols	13-16
13.7.2	Displaying Text	13-17
13.7.3	Displaying Files	13-17
13.8	Reading and Writing Files (File I/O)	13-17
13.8.1	Specifying Files in Batch Job Command Procedures	13-17
13.8.2	Writing to a File	13-18
13.8.3	Reading from a File	13-20
13.8.4	Modifying a File	13-21
13.8.5	Handling Input/Output (I/O) Errors	13-24

13.9 Restarting Batch Jobs	13-24
13.10 Cleanup Operations	13-25

Reference Section

DCL Commands

DCL Commands	DCL-1
= (Assignment Statement)	DCL-1
:= (String Assignment)	DCL-2
@ (Execute Procedure)	DCL-2
ACCOUNTING	DCL-4
ALLOCATE	DCL-4
ANALYZE/AUDIT	DCL-5
ANALYZE/CRASH_DUMP	DCL-5
ANALYZE/DISK_STRUCTURE	DCL-5
ANALYZE/ERROR_LOG	DCL-6
ANALYZE/IMAGE	DCL-6
ANALYZE/MEDIA	DCL-7
ANALYZE/OBJECT	DCL-7
ANALYZE/PROCESS_DUMP	DCL-9
ANALYZE/RMS_FILE	DCL-11
ANALYZE/SYSTEM	DCL-12
APPEND	DCL-12
ASSIGN	DCL-16
ASSIGN/MERGE	DCL-18
ASSIGN/QUEUE	DCL-19
ATTACH	DCL-20
BACKUP	DCL-20
CALL	DCL-21
CANCEL	DCL-23
CLOSE	DCL-25
CONNECT	DCL-26
CONTINUE	DCL-27
CONVERT	DCL-28
CONVERT/DOCUMENT	DCL-28
CONVERT/RECLAIM	DCL-28
COPY	DCL-29
CREATE	DCL-35
CREATE/DIRECTORY	DCL-36
CREATE/FDL	DCL-37
CREATE/NAME_TABLE	DCL-38
DEALLOCATE	DCL-39
DEASSIGN	DCL-40
DEASSIGN/QUEUE	DCL-42

DEBUG	DCL-43
DECK	DCL-43
DEFINE	DCL-45
DEFINE/CHARACTERISTIC	DCL-47
DEFINE/FORM	DCL-48
DEFINE/KEY	DCL-50
DELETE	DCL-53
DELETE/CHARACTERISTIC	DCL-56
DELETE/ENTRY	DCL-57
DELETE/FORM	DCL-58
DELETE/INTRUSION_RECORD	DCL-58
DELETE/KEY	DCL-59
DELETE/QUEUE	DCL-60
DELETE/SYMBOL	DCL-61
DEPOSIT	DCL-62
DIFFERENCES	DCL-63
DIRECTORY	DCL-68
DISCONNECT	DCL-74
DISMOUNT	DCL-75
DUMP	DCL-78
EDIT/ACL	DCL-81
EDIT/EDT	DCL-81
EDIT/FDL	DCL-83
EDIT/SUM	DCL-83
EDIT/TECO	DCL-84
EDIT/TPU	DCL-85
EOD	DCL-86
EOJ	DCL-86
ENDSUBROUTINE	DCL-87
EXAMINE	DCL-87
EXCHANGE	DCL-89
EXCHANGE/NETWORK	DCL-90
EXIT	DCL-94
GOSUB	DCL-95
GOTO	DCL-96
HELP	DCL-97
IF	DCL-98
INITIALIZE	DCL-100
INITIALIZE/QUEUE	DCL-106
INQUIRE	DCL-115
INSTALL	DCL-116
JOB	DCL-116

Lexical Functions	DCL-121
F\$CONTEXT	DCL-123
F\$CVSI	DCL-129
F\$CVTIME	DCL-130
F\$CVUI	DCL-131
F\$DIRECTORY	DCL-132
F\$EDIT	DCL-133
F\$ELEMENT	DCL-134
F\$ENVIRONMENT	DCL-135
F\$EXTRACT	DCL-137
F\$FAO	DCL-139
F\$FILE_ATTRIBUTES	DCL-144
F\$GETDVI	DCL-146
F\$GETJPI	DCL-160
F\$GETQUI	DCL-164
F\$GETSYI	DCL-181
F\$IDENTIFIER	DCL-185
F\$INTEGER	DCL-186
F\$LENGTH	DCL-187
F\$LOCATE	DCL-188
F\$MESSAGE	DCL-189
F\$MODE	DCL-189
F\$PARSE	DCL-190
F\$PID	DCL-192
F\$PRIVILEGE	DCL-194
F\$PROCESS	DCL-195
F\$SEARCH	DCL-195
F\$SETPRV	DCL-196
F\$STRING	DCL-197
F\$TIME	DCL-198
F\$TRNLNM	DCL-198
F\$TYPE	DCL-201
F\$USER	DCL-202
F\$VERIFY	DCL-202
DCL Commands	DCL-204
LIBRARY	DCL-204
LICENSE	DCL-204
LINK	DCL-204
LOGIN Procedure	DCL-208
LOGOUT	DCL-210
MACRO	DCL-211
MAIL	DCL-216
MERGE	DCL-216

MESSAGE	DCL-216
MONITOR	DCL-216
MOUNT	DCL-217
NCS	DCL-217
ON	DCL-217
OPEN	DCL-218
PASSWORD	DCL-220
PATCH	DCL-221
PHONE	DCL-222
PRINT	DCL-222
PURGE	DCL-229
READ	DCL-231
RECALL	DCL-234
RENAME	DCL-235
REPLY	DCL-238
REQUEST	DCL-242
RETURN	DCL-243
RUN (Image)	DCL-245
RUN (Process)	DCL-246
RUNOFF	DCL-251
RUNOFF/CONTENTS	DCL-258
RUNOFF/INDEX	DCL-261
SEARCH	DCL-264
SET ACCOUNTING	DCL-269
SET ACL	DCL-270
SET AUDIT	DCL-274
SET BROADCAST	DCL-284
SET CARD_READER	DCL-285
SET CLUSTER/EXPECTED_VOTES	DCL-286
SET COMMAND	DCL-286
SET CONTROL	DCL-286
SET DAY	DCL-287
SET DEFAULT	DCL-288
SET DEVICE	DCL-288
SET DEVICE/SERVED	DCL-290
SET DIRECTORY	DCL-291
SET DISPLAY	DCL-293
SET ENTRY	DCL-297
SET FILE	DCL-304
SET HOST	DCL-308
SET HOST/DTE	DCL-309
SET HOST/DUP	DCL-310
SET HOST/HSC	DCL-311
SET KEY	DCL-312

SET LOGINS	DCL-313
SET MAGTAPE	DCL-313
SET MESSAGE	DCL-315
SET ON	DCL-316
SET OUTPUT_RATE	DCL-317
SET PASSWORD	DCL-317
SET PRINTER	DCL-319
SET PROCESS	DCL-322
SET PROMPT	DCL-325
SET PROTECTION	DCL-326
SET PROTECTION/DEFAULT	DCL-327
SET PROTECTION/DEVICE	DCL-328
SET QUEUE	DCL-329
SET QUEUE/ENTRY	DCL-335
SET RESTART_VALUE	DCL-336
SET RIGHTS_LIST	DCL-337
SET RMS_DEFAULT	DCL-339
SET SYMBOL	DCL-340
SET TERMINAL	DCL-341
SET TIME	DCL-351
SET UIC	DCL-352
SET VERIFY	DCL-352
SET VOLUME	DCL-353
SET WORKING_SET	DCL-356
SHOW ACCOUNTING	DCL-357
SHOW ACL	DCL-358
SHOW AUDIT	DCL-359
SHOW BROADCAST	DCL-360
SHOW CLUSTER	DCL-360
SHOW CPU	DCL-360
SHOW DEFAULT	DCL-362
SHOW DEVICES	DCL-363
SHOW DEVICES/SERVED	DCL-364
SHOW DISPLAY	DCL-366
SHOW ENTRY	DCL-368
SHOW ERROR	DCL-370
SHOW INTRUSION	DCL-371
SHOW KEY	DCL-371
SHOW LICENSE	DCL-373
SHOW LOGICAL	DCL-374
SHOW MAGTAPE	DCL-376
SHOW MEMORY	DCL-377
SHOW NETWORK	DCL-378
SHOW PRINTER	DCL-379

SHOW PROCESS	DCL-380
SHOW PROTECTION	DCL-382
SHOW QUEUE	DCL-383
SHOW QUEUE/CHARACTERISTIC	DCL-385
SHOW QUEUE/FORM	DCL-386
SHOW QUOTA	DCL-387
SHOW RMS_DEFAULT	DCL-388
SHOW STATUS	DCL-389
SHOW SYMBOL	DCL-390
SHOW SYSTEM	DCL-391
SHOW TERMINAL	DCL-394
SHOW TIME	DCL-395
SHOW TRANSLATION	DCL-396
SHOW USERS	DCL-396
SHOW WORKING_SET	DCL-398
SORT	DCL-399
SPAWN	DCL-399
START/CPU	DCL-402
START/QUEUE	DCL-402
START/QUEUE/MANAGER	DCL-411
STOP	DCL-413
STOP/CPU	DCL-414
STOP/QUEUE	DCL-415
STOP/QUEUE/ABORT	DCL-416
STOP/QUEUE/ENTRY	DCL-416
STOP/QUEUE/MANAGER	DCL-417
STOP/QUEUE/NEXT	DCL-417
STOP/QUEUE/REQUEUE	DCL-418
STOP/QUEUE/RESET	DCL-419
SUBMIT	DCL-420
SUBROUTINE	DCL-426
SYNCHRONIZE	DCL-426
TYPE	DCL-427
UNLOCK	DCL-430
VIEW	DCL-431
WAIT	DCL-431
WRITE	DCL-432

DIGITAL Standard Runoff (DSR) Commands		DSR-1
1	DSR Command Format	DSR-1
2	Entering DSR Commands	DSR-2
3	DSR Commands	DSR-3
	.APPENDIX	DSR-3
	.AUTOJUSTIFY, .NO AUTOJUSTIFY	DSR-3
	.AUTOPARAGRAPH, .NO AUTOPARAGRAPH	DSR-3
	.AUTOSUBTITLE, .NO AUTOSUBTITLE	DSR-4
	.AUTOTABLE, .NO AUTOTABLE	DSR-4
	.BLANK	DSR-4
	.BREAK	DSR-4
	.CENTER (.CENTRE)	DSR-5
	.CHAPTER	DSR-5
	.CONTROL CHARACTERS, .NO CONTROL CHARACTERS	DSR-5
	.DATE, .NO DATE	DSR-5
	.DISPLAY APPENDIX	DSR-5
	.DISPLAY CHAPTER	DSR-6
	.DISPLAY ELEMENTS	DSR-6
	.DISPLAY LEVELS	DSR-6
	.DISPLAY NUMBER	DSR-7
	.DISPLAY SUBPAGE	DSR-7
	.ENABLE BAR, .DISABLE BAR, .BEGIN BAR, .END BAR	DSR-7
	.ENABLE BOLDING, .DISABLE BOLDING	DSR-8
	.ENABLE HYPHENATION, .DISABLE HYPHENATION	DSR-8
	.ENABLE INDEXING, .DISABLE INDEXING	DSR-8
	.ENABLE OVERSTRIKING, .DISABLE OVERSTRIKING	DSR-8
	.ENABLE TOC, .DISABLE TOC	DSR-8
	.ENABLE UNDERLINING, .DISABLE UNDERLINING	DSR-9
	.ENTRY	DSR-9
	.FIGURE DEFERRED, .FIGURE	DSR-9
	.FILL, .NO FILL	DSR-9
	.FIRST TITLE	DSR-10
	.FLAGS ACCEPT, .NO FLAGS ACCEPT	DSR-10
	.FLAGS ALL, .NO FLAGS ALL	DSR-10
	.FLAGS BOLD, .NO FLAGS BOLD	DSR-10
	.FLAGS BREAK, .NO FLAGS BREAK	DSR-10
	.FLAGS CAPITALIZE, .NO FLAGS CAPITALIZE	DSR-11
	.FLAGS COMMENT, .NO FLAGS COMMENT	DSR-11

.FLAGS CONTROL, .NO FLAGS CONTROL	DSR-11
.FLAGS HYPHENATE, .NO FLAGS HYPHENATE	DSR-11
.FLAGS INDEX, .NO FLAGS INDEX	DSR-11
.FLAGS LOWERCASE, .NO FLAGS LOWERCASE	DSR-12
.FLAGS OVERSTRIKE, .NO FLAGS OVERSTRIKE	DSR-12
.FLAGS PERIOD, .NO FLAGS PERIOD	DSR-12
.FLAGS SPACE, .NO FLAGS SPACE	DSR-12
.FLAGS SUBINDEX, .NO FLAGS SUBINDEX	DSR-12
.FLAGS SUBSTITUTE, .NO FLAGS SUBSTITUTE	DSR-12
.FLAGS UNDERLINE, .NO FLAGS UNDERLINE	DSR-13
.FLAGS UPPERCASE, .NO FLAGS UPPERCASE	DSR-13
.FOOTNOTE, .END FOOTNOTE	DSR-13
.HEADER LEVEL	DSR-13
.HEADERS ON, .NO HEADERS	DSR-14
.HEADERS UPPER, .HEADERS LOWER, .HEADERS MIXED	DSR-14
.IF, .IFNOT, .ELSE, .ENDIF	DSR-14
.INDENT	DSR-15
.INDEX	DSR-15
.JUSTIFY, .NO JUSTIFY	DSR-15
.KEEP, .NO KEEP	DSR-15
.LAYOUT	DSR-15
.LEFT MARGIN	DSR-15
.LIST, .END LIST	DSR-16
.LIST ELEMENT	DSR-16
.LITERAL	DSR-16
.NO SPACE	DSR-17
.NOTE, .END NOTE	DSR-17
.NUMBER APPENDIX	DSR-17
.NUMBER CHAPTER	DSR-17
.NUMBER LEVEL	DSR-18
.NUMBER LIST	DSR-18
.NUMBER PAGE, .NO NUMBER	DSR-18
.NUMBER RUNNING	DSR-19
.NUMBER SUBPAGE	DSR-19
.PAGE	DSR-19
.PAGE SIZE	DSR-19
.PAGING, .NO PAGING	DSR-19
.PARAGRAPH	DSR-20
.PERIOD, .NO PERIOD	DSR-20
.REPEAT	DSR-20
.REQUIRE	DSR-20
.RIGHT	DSR-20
.RIGHT MARGIN	DSR-21

.SAVE, .RESTORE	DSR-21
.SEND TOC	DSR-21
.SET DATE, .SET TIME	DSR-21
.SET LEVEL	DSR-22
.SET PARAGRAPH	DSR-22
.SKIP	DSR-22
.SPACING	DSR-22
.STYLE HEADERS	DSR-22
.SUBPAGE, .END SUBPAGE	DSR-23
.SUBTITLE, .NO SUBTITLE	DSR-23
.TAB STOPS	DSR-23
.TEST PAGE	DSR-23
.TITLE	DSR-24
.VARIABLE	DSR-24
.XLOWER, .XUPPER	DSR-24

EDT Keypad Commands

ADVANCE Function	EDT-1
APPEND Function	EDT-1
BACKSPACE Function CTRL/H	EDT-1
BACKUP Function	EDT-2
BOTTOM Function	EDT-2
CHAR (Character) Function	EDT-2
CHNGCASE (Change Case) Function	EDT-2
COMMAND Function	EDT-4
CTRL/A (Control A) Function	EDT-4
CTRL/C (Control C) Function	EDT-4
CTRL/D (Control D) Function	EDT-5
CTRL/E (Control E) Function	EDT-5
CTRL/K (Control K) Function	EDT-6
CTRL/L (Control L) Function	EDT-7
CTRL/M (Control M) Function	EDT-8
CTRL/R (Control R) Function	EDT-8
CTRL/T (Control T) Function	EDT-8
CTRL/U (Control U) Function	EDT-9
CTRL/W (Control W) Function	EDT-9
CTRL/Z (Control Z) Function	EDT-9
CUT Function	EDT-10
DEL C (Delete Character) Function	EDT-10
DEL EOL (Delete to End of Line) Function	EDT-11
DELETE Function	EDT-11
DEL L (Delete Line) Function	EDT-12
DEL W (Delete Word) Function	EDT-12
DO Function (LK201 only)	EDT-13

DOWN Arrow	EDT-13
ENTER Function	EDT-14
EOL (End of Line) Function	EDT-14
FILL Function (VT100)	EDT-14
FIND Function	EDT-15
FNDNXT (Find Next) Function	EDT-16
GOLD Function	EDT-16
HELP Function	EDT-17
LEFT Arrow	EDT-18
LINE Function	EDT-18
LINEFEED Function	EDT-18
OPEN LINE Function	EDT-19
PAGE Function	EDT-19
PASTE Function	EDT-20
REPLACE Function	EDT-21
RESET Function	EDT-21
RETURN Function	EDT-22
RIGHT Arrow	EDT-22
SECT (Section) Function	EDT-22
SELECT Function	EDT-23
SPECINS (Special Insert) Function	EDT-23
string specifier	EDT-24
SUBS (Substitute) Function	EDT-24
TAB Function	EDT-25
TOP Function	EDT-26
UND C (Undelete Character) Function	EDT-26
UND L (Undelete Line) Function	EDT-27
UND W (Undelete Word) Function	EDT-28
UP Arrow	EDT-28
WORD Function	EDT-29

EVE Commands

EVE Commands	EVE-1
@	EVE-1
ATTACH	EVE-2
BOTTOM	EVE-3
BUFFER	EVE-3
CAPITALIZE WORD	EVE-4
CENTER LINE	EVE-5
CHANGE DIRECTION	EVE-5
CHANGE MODE	EVE-6
COPY	EVE-6
CUT	EVE-7
DCL	EVE-7
DEFINE KEY	EVE-8

DELETE	EVE-9
DELETE BUFFER	EVE-10
DELETE WINDOW	EVE-11
DO	EVE-11
END OF LINE	EVE-12
ENLARGE WINDOW	EVE-12
ERASE CHARACTER	EVE-13
ERASE LINE	EVE-13
ERASE PREVIOUS WORD	EVE-14
ERASE START OF LINE	EVE-14
ERASE WORD	EVE-15
EXIT	EVE-15
EXTEND ALL	EVE-16
EXTEND EVE	EVE-17
EXTEND THIS	EVE-18
EXTEND TPU	EVE-18
FILL	EVE-18
FILL PARAGRAPH	EVE-19
FILL RANGE	EVE-19
FIND	EVE-20
FIND NEXT	EVE-21
FIND SELECTED	EVE-21
FORWARD	EVE-22
GET FILE	EVE-22
GO TO	EVE-23
HELP	EVE-24
INCLUDE FILE	EVE-25
INSERT HERE	EVE-26
INSERT MODE	EVE-26
INSERT PAGE BREAK	EVE-27
LEARN	EVE-27
LINE	EVE-28
LOWERCASE WORD	EVE-29
MARK	EVE-29
MOVE BY LINE	EVE-30
MOVE BY PAGE	EVE-31
MOVE BY WORD	EVE-31
MOVE DOWN	EVE-31
MOVE LEFT	EVE-32
MOVE RIGHT	EVE-33
MOVE UP	EVE-33
NEW	EVE-34
NEXT BUFFER	EVE-35
NEXT SCREEN	EVE-35

NEXT WINDOW	EVE-36
ONE WINDOW	EVE-37
OPEN	EVE-37
OPEN SELECTED	EVE-37
OTHER WINDOW	EVE-38
OVERSTRIKE MODE	EVE-38
PAGINATE	EVE-39
PASTE	EVE-39
PREVIOUS SCREEN	EVE-40
PREVIOUS WINDOW	EVE-40
QUIT	EVE-41
QUOTE	EVE-41
RECALL	EVE-42
REFRESH	EVE-43
REMEMBER	EVE-43
REMOVE	EVE-44
REPEAT	EVE-45
REPLACE	EVE-46
RESET	EVE-48
RESTORE	EVE-48
RESTORE CHARACTER	EVE-49
RESTORE LINE	EVE-49
RESTORE SELECTION	EVE-49
RESTORE SENTENCE	EVE-50
RESTORE WORD	EVE-50
RETURN	EVE-50
REVERSE	EVE-51
SAVE EXTENDED EVE	EVE-52
SAVE EXTENDED TPU	EVE-53
SAVE FILE	EVE-53
SAVE FILE AS	EVE-54
SELECT	EVE-55
SELECT ALL	EVE-57
SET BUFFER	EVE-57
SET CLIPBOARD	EVE-58
SET CURSOR BOUND	EVE-59
SET CURSOR FREE	EVE-59
SET FIND NOWHITESPACE	EVE-60
SET FIND WHITESPACE	EVE-60
SET GOLD KEY	EVE-61
SET KEYPAD EDT	EVE-62
SET KEYPAD NOEDT	EVE-66
SET KEYPAD NOWPS	EVE-66
SET KEYPAD NUMERIC	EVE-67

SET KEYPAD VT100	EVE-67
SET KEYPAD WPS	EVE-69
SET LEFT MARGIN	EVE-72
SET NOCLIPBOARD	EVE-73
SET NOGOLD KEY	EVE-73
SET NOPENDING DELETE	EVE-74
SET NOWRAP	EVE-74
SET PARAGRAPH INDENT	EVE-74
SET PENDING DELETE	EVE-76
SET RIGHT MARGIN	EVE-76
SET SCROLL MARGINS	EVE-77
SET TABS	EVE-78
SET WIDTH	EVE-80
SET WILDCARD ULTRIX	EVE-81
SET WILDCARD VMS	EVE-81
SET WRAP	EVE-82
SHIFT LEFT	EVE-82
SHIFT RIGHT	EVE-83
SHOW	EVE-84
SHOW BUFFERS	EVE-85
SHOW DEFAULTS BUFFER	EVE-86
SHOW KEY	EVE-86
SHOW SUMMARY	EVE-87
SHOW SYSTEM BUFFERS	EVE-87
SHOW WILDCARDS	EVE-88
SHRINK WINDOW	EVE-88
SPAWN	EVE-89
SPELL	EVE-90
SPLIT WINDOW	EVE-91
START OF LINE	EVE-91
STORE TEXT	EVE-92
TAB	EVE-93
TOP	EVE-93
TPU	EVE-94
TWO WINDOWS	EVE-94
UNDEFINE KEY	EVE-95
UPPERCASE WORD	EVE-96
WHAT LINE	EVE-96
WILDCARD FIND	EVE-97
WRITE FILE	EVE-98

Mail Utility

MAIL-1

Sort/Merge Utility

SORT-1

Appendix A TFF Facility

A.1	Using the Terminal Fallback Facility	A-1
A.1.1	The Purpose of Terminal Fallback	A-2
A.1.2	The Purpose of Compose Characters	A-2
A.1.3	Setting TFF Terminal Parameters	A-7

Appendix B Character Sets

B.1	ASCII Character Set	B-1
B.2	ASCII and DEC Multinational Character Set Tables	B-2

Appendix C Expressions

Appendix D Terminal Keys

D.1	VT300 and VT200 Terminal Series	D-1
D.2	VT100 Terminal Series	D-2

Index

Figures

3-1	Parts of a DCL Command Line	3-3
5-1	Directory Structure	5-2
6-1	Editing Keys—VT200-Series and VT300-Series Terminals	6-5
6-2	Editing Keys—VT100-Series Terminals	6-6
8-1	Sample Mail Message	8-1
EVE-1	EVE Default Keys for VT300- and VT200-Series Terminals	EVE-68
EVE-2	EVE Default Keys for VT100-Series Terminals	EVE-69
B-1	Graphical Representation of the ASCII Character Set	B-3
B-2	Graphical Representation of the DEC Multinational Extension to the ASCII Character Set	B-4

Tables

3-1	Commonly Used DCL Commands	3-2
3-2	Commonly Used DCL Key Combinations	3-2
3-3	Keys That Execute Terminal Functions	3-15
4-1	Default File Types	4-3
6-1	EVE Key Names	6-8
6-2	EVE Default Settings	6-15
6-3	EVE Commands and Default Predefined Keys	6-37
7-1	Symbols for EDT Functions	7-36
11-1	Default Process Logical Names	11-5
11-2	Default Job Logical Names	11-6
11-3	Default System Logical Names	11-7
11-4	Default Process Directory Logical Names	11-9
11-5	Default System Directory Logical Names	11-9
11-6	Equivalence Names for Process-Permanent Logical Names	11-15
12-1	Determining the Value of an Expression	12-20
DCL-1	Summary of Lexical Functions	DCL-121
DCL-2	Summary of FAO Directives	DCL-140
DCL-3	F\$FILE_ATTRIBUTES Items	DCL-144
DCL-4	F\$GETDVI Items	DCL-147
DCL-5	Values Returned by the DEVCLASS Item	DCL-154
DCL-6	Values Returned by the DEVTYPE Item	DCL-155
DCL-7	F\$GETJPI Items	DCL-161
DCL-8	F\$GETQUI Items	DCL-168
DCL-9	F\$GETSYI Items for the Local Node Only	DCL-182
DCL-10	F\$GETSYI Items for the Local Node or for Other Nodes in the VAXcluster	DCL-183
EVE-1	EVE Default GOLD Key Combinations	EVE-61
A-1	LATIN_1 Compose Sequence Table	A-3

Preface

The *VMS User's Manual* provides an overview of the VMS operating system.

Intended Audience

This manual is intended for all users of the VMS operating system.

Document Structure

This manual is organized into two major parts. Chapters 1 through 13 describe VMS concepts and procedures users need to perform basic computing tasks. The Reference Section contains the following VMS user reference information:

- DCL commands—In alphabetical order, describes all Digital Command Language (DCL) commands and lexical functions.
- DIGITAL Standard Runoff (DSR) commands—Contains the rules you must follow and commands, in alphabetical order, you use to format output with DSR.
- EDT Editor—Provides reference information about EDT keypad editing.
- EVE Commands—In alphabetical order, describes all EVE editing commands.
- MAIL—Describes the commands and qualifiers you can use to send messages to other users.
- VMS Sort/Merge—Describes VMS Sort/Merge Utility, which you can use to sort records or to merge input files.

Four appendixes contain tables that list the following information:

- Terminal Fallback Facility
- ASCII character set
- DCL Expressions
- Terminal Keys

Conventions

The following conventions are used in this manual:

Ctrl/x	A sequence such as Ctrl/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 x	A sequence such as PF1 x indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
RET	A key name is shown enclosed to indicate that you press a key on the keyboard.
...	In examples, a horizontal ellipsis indicates one of the following possibilities: <ul style="list-style-type: none">• Additional optional arguments in a statement have been omitted.• The preceding item or items can be repeated one or more times.• Additional parameters, values, or other information can be entered.
.	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
()	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[]	In format descriptions, brackets indicate that whatever is enclosed is optional; you can select none, one, or all of the choices.
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
red ink	Red ink indicates information that you must enter from the keyboard or a screen object that you must choose or click on. For online versions, user input is shown in bold .
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason.
<i>italic text</i>	Italic text represents information that can vary in system messages (for example, <i>Internal error number</i>).
UPPERCASE TEXT	Uppercase letters indicate that you must enter a command (for example, enter OPEN/READ). Uppercase letters can also indicate the name of a command, the name of a file, the name of a file protection code, or the abbreviation for a system privilege.

numbers

Hyphens in coding examples indicate that additional arguments to the request are provided on the line that follows.

Unless otherwise noted, all numbers in the text are assumed to be decimal. Nondecimal radices—binary, octal, or hexadecimal—are explicitly indicated.

Chapter 1

Introduction: VMS Concepts and Definitions

Your VAX computer operates under the control of the VMS (Virtual Memory System) operating system. The VMS operating system controls VAX computer system resources and schedules access to these resources.

VMS is an interactive operating system. While you are logged in to the computer, you and the system conduct a dialogue using the **DIGITAL Command Language**. You use DCL by entering commands, called **DCL Commands**, which are read and translated by the system. You enter a command by typing it on your keyboard and pressing the RETURN key; the system responds by executing your command or by displaying an error message on your screen if it cannot interpret what you entered.

In this manual, VMS is sometimes used to refer to the VMS operating system. **System** refers to a VAX computer that is running the VMS operating system.

1.1 How to Use This Manual

The *VMS User's Manual* is divided into two parts. Chapters 1 through 13 describe VMS concepts and procedures users need to perform basic computing tasks. The Reference Section contains VMS user reference information.

This chapter describes the concepts and definitions used in the *VMS User's Manual*. Chapters 2 through 13 describe the tasks you can perform on the VMS operating system. Read this chapter to understand basic concepts about the VMS operating system and its components. Read chapters 2 through 13 to find out how to do specific tasks on the VMS operating system.

A **system manager** performs the administrative tasks that create and maintain an efficient computing environment. If you are a system manager or want to understand system management concepts and procedures, refer to the *VMS System Manager's Manual*.

The topics discussed in this manual require you to be familiar with your terminal. For information on setting up or using your terminal, see the owner's manual supplied with your terminal.

1.2 Logging In to the System

To interact with the VMS operating system, you must log in to a user **account**. An account is a name or number that identifies a user to the system when the user logs in. That name or number tells the system where the user's files are and the type of access to other files the user has. User accounts are either **privileged** or **nonprivileged**. System managers grant privileges according to users' needs.

Logging in consists of getting the system's attention and identifying yourself as an authorized user. Your system manager (or whoever authorizes system use at your installation) usually sets up accounts. This person provides you with your user name and password. Your user name identifies you to the system and distinguishes you from other users. In many cases, a user name is your first or last name. Your password is for your protection. If you maintain its secrecy, other users cannot use system resources under your user name.

Chapter 2 describes how to log in to and out of the system.

1.3 Using a Network

Your system might be part of a DECnet-VAX **network**. When computer systems are linked together, they form a network. VMS systems in a DECnet network are able to communicate with each other and share information and resources. Each system in a network is called a network **node** and is identified by a unique node name.

When you are logged in to a network node, you can communicate with other nodes in the network. The node at which you are logged in is called the **local node**; other nodes on the network are called **remote nodes**. If you have access to an account on a remote node, you can log in to that account from your local node and perform tasks on that node while remaining connected to your local node.

Chapter 2 describes how to log in to a remote node. Additional tasks you can perform on remote nodes are described in the appropriate chapters of this book.

NOTE: In the examples of remote operations in this manual, proxy accounts enable users to perform operations on remote systems. Proxy accounts are one way users can access remote systems. For more information, about additional ways to access remote systems, see the *VMS System Manager's Manual*.

1.4 The DIGITAL Command Language (DCL)

DCL (DIGITAL Command Language) is a set of English-like instructions that tell the VMS operating system to perform specific operations. DCL provides you with over 200 commands and functions to use in communicating with the VMS operating system to accomplish your computing tasks.

You can use DCL in the following two modes:

- **Interactive**—In interactive mode, you enter commands from your terminal. One command has to finish executing before you can enter another.
- **Batch**—In batch mode, the system creates another **process** to execute commands on your behalf. **Batch** jobs and network processes use DCL in batch mode. A process is an environment created by the system that makes it possible for you to work with the system. (See Section 1.7 and Chapter 10 for more information about processes.) A batch job is a command procedure or program that is submitted to the operating system for execution as a separate user process. After you submit the command procedure for batch execution, you can continue to use your terminal interactively.

When you enter a DCL command, it is read and translated by the DCL interpreter. The way the command interpreter responds to a command is determined by the type of command entered. The three types of DCL commands are as follows:

- **Built-in commands**—These commands are built into the DCL interpreter and are executed internally.
- **Commands that invoke programs**—DCL calls another program to execute the command rather than executing it internally. The program invoked to execute a command is referred to as a **command image**. This command image can be either an interactive program like MAIL or a noninteractive program like COPY.
- **Foreign commands**—A symbol that executes an image is referred to as a **foreign command**. (See Section 1.13 for more information about symbols.) A foreign command executes an image whose name is not recognized by the command interpreter as a DCL command. The following example defines the symbol FUN as a foreign command. (No DCL command FUN exists.)

```
$ FUN := $DISK1:[ROY.PROGRAMS]GAMES.EXE
```

1.4.1 The DCL Command Line

DCL, like any language, has its own vocabulary and usage rules. The vocabulary consists of commands, parameters, and qualifiers, which are put together in a way that DCL can interpret. The way in which the parts of a command line are put together is referred to as the **command line syntax**. A DCL command line contains the following information in the format shown:

```
[$] [label:] command [/qualifier[=value]...] [parameter[/qualifier...]]
```

NOTE: Items in square brackets [] are optional and might not be required by a specific command.

The following table briefly describes the components of a DCL command line:

\$	The dollar sign is the DCL prompt. When you work interactively with DCL, DCL displays the prompt when it is ready to accept a command.
Label	Identifies a line in a command procedure. Labels are not used for commands that are entered interactively.
Command	Specifies the name of the command.
Qualifier	Modifies the action taken by the command. Some qualifiers can modify parameters. Some can accept values.
Parameter	Specifies what the command acts upon. You must position parameters in a specified order within the command.
Value	Modifies a qualifier and is often preceded by an equal sign. A value can be a file specification, a character string, a number, or a DCL keyword . A keyword is a word reserved for use in certain specified syntax formats.

Chapter 3 shows a sample command line and describes how to use DCL commands.

The Reference Section lists in alphabetical order and describes all DCL commands and **lexical functions**. (Lexical functions are command language constructs that the DCL interpreter evaluates and substitutes before it interprets a command string. Chapter 12 discusses lexical functions in more detail.)

1.5 Files and Directories

A **file** contains information. This information can be machine-readable data that the computer understands. It can also be text you enter and manipulate. The text in the file might be the text of a document, a program, or a list of addresses. You can examine the data in these files by displaying the files on a terminal screen or by printing them on paper.

A file is listed in a **directory**. A directory is a special kind of file that contains the names and locations of files. Directory files are stored on **disks**. Disks are the one of the hardware devices the VMS operating system uses to store information. See Section 1.6 for more information about disks.

Chapter 4 describes how to create and organize files to store information. Chapter 5 describes how to use directories to organize and manage files.

1.5.1 File and Directory Specifications

Every file must have a file name or file type to identify it to both the system and you. A file also has a version number. You can have several versions of a file. Unless you specify a version number, the system uses the highest existing version number of a file. When you edit a file, the system saves the original file and produces a modified output file. By default, the output file has the same name and type as the original, but the version number is incremented by one. The file name, type, and version number form a **file specification**. This information is specified in the following format:

filename.type;version

A directory file has the following format:

directory.DIR;1

For example, DOG.DIR;1 is a directory file. You cannot edit a directory file.

A **full file specification** contains the following information in the format shown:

node-name::device:[directory]filename.type;version

Because a full file specification describes the network node on which the file resides, a full file specification is also known as a **network file specification**.

A full file specification completely describes the access path the system uses to locate and identify a file. In addition to the file name, a file specification can include the directory in which the file is located. For example, in the following command line, the file STAFF_VACATIONS.TXT is located in the directory [JONES]:

```
$ PRINT [JONES]STAFF_VACATIONS.TXT
```

If you omit the directory name from the file specification, the current directory is assumed by default.

When using file and directory specifications to create and manipulate files and directories, you can use **wildcard characters**. A wildcard character is a nonalphanumeric character, such as an asterisk or a percent sign, that is used within, or in place of, a file name, file type, directory name, or version number in a file specification to indicate (all) for the given field. Chapter 4 and Chapter 5 describe how to use wildcard characters in file and directory operations.

As mentioned previously, a directory stores files on a disk in a special format. This format is called a **directory structure**; Section 1.5.2 describes the components of a directory structure.

1.5.2 Directory Structures

Each disk contains a main directory, which can be set up by a system manager or by the system itself. This main directory is called the master file directory (MFD) and contains a list of user file directories (UFDs). User file directories are files in the master file directory that point to top level directories. Your top level directory is usually your **login** or **default directory**. Unless your account has been modified to do otherwise, by default the system places you in your top level directory when you log in.

In most cases, a UFD exists for each user on the system. It contains the names of and pointers to files cataloged in a user's directory. A **subdirectory** is any directory file that is not an MFD or a UFD. Subdirectories let you organize files into meaningful groups. Like a directory, a subdirectory contains names and pointers for the files cataloged within it. It can contain an entry for another

1-6 Introduction: VMS Concepts and Definitions

subdirectory, which can contain an entry for another subdirectory, and so on to seven levels of subdirectories. This structure (a top level directory plus a maximum of seven levels of subdirectories) is called a **hierarchical directory structure**. Chapter 5 contains more information about directory structures.

1.6 Devices

In the VMS operating system, devices are classified as follows:

- **Mass Storage Devices**—These devices, such as disks and magnetic tapes, save the contents of files on a magnetic medium. Files saved this way can be accessed, updated, modified, or reused at any time.
- **Record-oriented Devices**—These devices, such as terminals, printers, mailboxes, and card readers read and write only single physical units of data at a time and do not provide online storage of the data. (Printers and card readers are also called unit-record devices.)

A device name has the following three parts:

- The device type, which identifies the hardware device. (For example, an RP06 disk has the device type DB, and a TE16 magnetic tape has the device type MT.)
- A controller designator, which identifies the hardware controller to which the device is attached.
- The unit number, which uniquely identifies a device on a particular controller.

The files you commonly access are stored on disks or magnetic tape. Your user file directory (UFD) and your default directory with all your files and subdirectories are located on a disk. You can use a file specification that contains directory information only if the file is located on a disk. Magnetic tapes do not have directory structures. To obtain a file stored on tape, use a file specification that contains only file information.

If you want to access a file that is not located on your default device, you must specify the device name. For files on disks, you must also specify the directory where the file is cataloged.

You can use physical, logical, or generic names, described in the following sections, to refer to devices.

1.6.1 Physical Device Names

Each physical device known to the system is uniquely identified by a **physical device name**. The physical device name identifies the kind of device, for example, a storage disk or a terminal. A device name has the following format:

ddcu

The fields are as follows:

- dd Device code that represents a device type.
- c Controller designation. The controller designation, along with the unit number, identifies the location of the device within the hardware configuration of the system. Controllers are designated with alphabetic letters A through Z.
- u Unit number. The unit number, along with the controller designation, identifies the location of the device within the hardware configuration of the system. Unit numbers are decimal numbers from 0 through 65535.

The maximum length of the device name field, including the controller and the unit number, is 15 characters. When you specify a device name as part of a file specification, end it with a colon (:). If you do not specify a logical or physical device name, your default device name is supplied.

1.6.2 Logical Device Names

Your system manager has probably set up logical device names to represent the devices on your system. **Logical device names** can be used to equate a somewhat cryptic device name to a short, meaningful name. Use these logical device names, rather than the physical device names, to refer to devices.

Chapter 11 describes how to use logical names.

1.6.3 Generic Device Names

A **generic device name** consists of the device code and omits the specific controller or unit number. When you use a generic device name, the system locates the first available controller or device unit whose physical name satisfies the portions of the generic device name you specified.

1.7 Processes

When you log in, the system creates an environment from which you can enter commands. This environment is called your **process**. The system obtains the characteristics that are unique to your process from the **user authorization file (UAF)**. The UAF lists those users permitted to access the system and defines the characteristics for each user's process. The system manager usually maintains the UAF. It is within your process that the system executes your programs (also called images or executable images) one at a time.

A process can be a **detached process** (a process that is independent of other processes) or a **subprocess** (a process that is dependent on another process for its existence and resources). Your main process, also called your parent process, is a detached process.

Chapter 10 describes how to use processes to perform computing tasks.

1.8 Programs

A program, also called an **image** or an **executable image**, is a file that contains instructions and data in machine-readable format. A program can be either a command image or a noncommand image as follows:

- **Command image**—A command image is a program associated with and invoked by a DCL command. For example, when you type the DCL command COPY, the system executes the program SYS\$SYSTEM:COPY.EXE. COPY.EXE is a command image. A system directory named SYS\$SYSTEM contains a number of command image files, most of which are VMS-supplied. Use the DCL command DIRECTORY SYS\$SYSTEM to examine this system directory.
- **Noncommand image**—A noncommand image is a program not associated with a DCL command. To invoke a noncommand image, name the file containing the program as the parameter to the RUN command.

Image files can be VMS- or user-supplied and usually have a file type of EXE. You cannot examine an image file with the DCL commands TYPE, PRINT, or EDIT because image files do not consist of ASCII characters. (Text files contain ASCII characters, which are a standard method of representing the alphabet, punctuation marks, numerals, and other special symbols.) Chapter 10 contains more information about using programs.

1.9 Utilities

A utility is a computer program that provides a service. Utilities are invoked with DCL commands. Some utilities—**interactive utilities**—provide a special environment from which you can perform a specific set of tasks. You work interactively with these utilities by entering subcommands and other information in response to the utility's prompt. For example, MAIL is an interactive utility; it has its own prompt and subcommands.

Other utilities are noninteractive. Noninteractive utilities look and act like DCL commands; when you invoke a noninteractive utility, it occupies your terminal and executes a task. When the task is complete, you are returned to DCL level and your terminal is once again available. The SORT/MERGE and the LIBRARIAN utilities are two examples of noninteractive utilities.

Some utilities, both interactive and noninteractive, prompt you for a file name or other information. When you are using such a utility (for example, BACKUP, MESSAGE, PATCH, and SORT/MERGE), you can add qualifiers to the DCL command line to tailor the utility to your specific needs, as shown in the following example:

```
$ BACKUP/RECORD/IMAGE/LOG [RET]
_ From:
```


1.9.1 MAIL

MAIL allows you to send messages to and receive messages from other users on your system or on any VAX computer that is connected to your system by DECnet-VAX.

Chapter 8 describes how to use MAIL.

1.9.2 VMS SORT/MERGE

The VMS Sort Utility (SORT), invoked with the DCL command SORT, sorts records from one or more input files according to the fields you select and generates one reordered output file. The Sort Utility reorders records in a file (or files) so that they are in alphabetic or numeric order, either low to high (ascending) or high to low (descending), based on a portion of each record that you define to be the **key**.

The VMS Merge Utility (MERGE), invoked with the DCL command MERGE, combines up to ten previously sorted files into one ordered output file.

For information about using SORT/MERGE, see Chapter 9 and the Reference Section.

1.10 Text Editors

Text editors allow you to create and modify text files. With a text editor, you can enter text from a keyboard and modify the text using text editing commands. For example, you can type in data for a report and then rearrange sections, duplicate information, substitute phrases, or format text. You can use text editors to create and modify source files for programming languages (such as PASCAL or VAX BASIC) or text formatters (such as VAX DOCUMENT or DIGITAL Standard Runoff). The VMS operating system supports several text editors. Chapter 6 describes how to use EVE; and Chapter 7 describe how to use EDT. The Reference Section lists EVE and EDT commands in alphabetical order.

1.11 DIGITAL Standard Runoff (DSR)

DIGITAL Standard Runoff (DSR) is a text formatter that processes source files into formatted text and intermediate files, and creates tables of contents and indexes. You use a text editor to create a source file, to which you should give a file type of RNO. This file contains text, DSR formatting commands, flags (special instruction characters you insert), and control characters.

The Reference Section describes how to use DSR and lists each DSR command.

1.12 Logical Names

A logical name is a name equated to an equivalence string name or to a list of equivalence strings. When you define a logical name, you equate one character string to an **equivalence name**, which is usually a full or partial file specification, another logical name, or any other character string. Once you have equated a logical name to one or more equivalence names, you can use the logical name to refer to those equivalence names. For example, you might assign a logical name to your default disk and directory. Logical names serve two main functions:

- **Shorthand and readability**—You can define commonly used files, directories, and devices with short, meaningful logical names. Such names are easier to remember and type than the full file specifications. Names that you use frequently can be defined in your login command procedure. Names that most users on your system use frequently can be defined by a system manager in the site-specific system startup command procedure.
- **File independence**—You can use logical names to keep your programs and command procedures independent of physical file specifications. For example, if a command procedure references the logical name ACCOUNTS, you can equate ACCOUNTS to any file on any disk before executing the command procedure.

Chapter 11 contains more information about logical name tables and describes how to use logical names.

1.13 Symbols

Entering DCL command lines that include parameters, multiple qualifiers, and values can make for much typing and can be time-consuming. To simplify your interaction with DCL and save time, you can establish **symbols** to use in place of command names and entire command strings you type frequently. A symbol is a name that represents a numeric, character, or logical value. When you use a symbol in a DCL command line, DCL uses the value you assign to the symbol. By defining a symbol as a command line, you can execute the command by typing only the symbol name.

Symbols can also be used (especially in command procedures) to collect, store, and manipulate certain types of data.

Chapter 12 describes how to use symbols in DCL commands and command procedures.

1.14 Command Procedures

A **command procedure** is a file that contains a series of DCL commands. Some simple command procedures might only contain one or two DCL commands; complex command procedures can function as sophisticated computer programs. When a command procedure is executed, the DCL interpreter reads the file and executes the commands it contains.

If your system manager has set up a system **login command procedure**, it is executed when you log in. A login command procedure allows your system manager to ensure that certain commands are always executed when you and other users on your system log in.

After executing the system login command procedure, the system executes your personal login command procedure, if one exists. Your personal login command procedure allows you to customize your computing environment. The commands contained in it are executed every time you log in. Each time you log in, the system automatically executes up to two login command procedures.

The person who set up your account might have placed a login command procedure in your top level directory. (Unless your account has been specially modified to do otherwise, the system automatically places you in your top level directory when you log in.) If a login command procedure is not in your top level directory, you can create one yourself, name it LOGIN.COM, and place it in your top level directory unless your system manager tells you otherwise. A sample personal login command procedure is described in Chapter 13.

1.15 Account and System Security

The VMS operating system provides two related mechanisms to control the access that users have to system objects as follows:

- **UIC-based protection**—Each user process in the system is assigned a user identification code (UIC) in the user authorization file (UAF) with the Authorize Utility. Each object on the system, such as a file, is also assigned a UIC (typically the UIC of its creator). Each object also maintains a protection mask, a structure which defines the type of access allowed to users, based upon the relationship between the user UIC and the object UIC.
- **ACL-based protection**—An access control list (ACL) specifying the type of access to be granted or denied to a particular user or group of users can be associated with a system object. An ACL is an optional form of protection that is typically created by the object owner using the ACL editor (invoked with the DCL command EDIT/ACL) or the SET ACL command.

The system objects for which ACL-based protection can be specified are files, directories, devices, batch and print queues, logical name tables, and global sections. Users are specified by identifiers in the rights database that are assigned with the Authorize Utility.

1-12 Introduction: VMS Concepts and Definitions

Each VMS system site has unique security requirements. For this reason, every site should have a system security policy that outlines physical and software security requirements for system managers and users. The *VMS System Manager's Manual* describes the security features available with the VMS operating system and tasks system managers can perform to maintain account and system security. Chapter 4 describes how users can protect their files from unauthorized access.

Chapter 2

Getting Started: Interacting with VMS

This chapter describes the following basic tasks you use to interact with the VMS operating system:

- Logging in to the system
- Logging in to a remote node
- Changing your password
- Recognizing system responses
- Getting help about the system
- Terminating a remote session
- Logging out of the system

The way you log in to and out of the VMS operating system depends on how the system is set up at your site. This section provides a general description of logging in to and out of the VMS operating system. Check with your system manager for the procedures specific to your site.

2.1 Logging In to the System

You need two pieces of information to log in to the system: your user name and your password. Your system manager usually sets up accounts and gives you your user name and password.

To log in to the system, use the following procedure:

1. Make sure your terminal is plugged in and the power is turned on.
2. Press the RETURN key to signal the system that you want to log in. (You might need to press RETURN several times.) The system displays a prompt for your user name:

Username:

2-2 Getting Started: Interacting with VMS

3. Enter your user name and press RETURN. (You have about 30 seconds to do this, otherwise the system “times out” and you must start the login procedure again.) The system displays your user name on the screen as you type it. For example:

```
Username: CASEY [RET]
```

The system prompts you for your password as follows:

```
Password:
```

4. Enter your password and press RETURN. The system does not display your password.

The following example shows a successful login:

```
[RET]
Username: CASEY [RET]
Password: [RET]
Welcome to VAX/VMS Version 5.2 on node MARS
Last interactive login on Friday, 19-APR-1990 08:41
Last non-interactive login on Thursday, 19-APR-1990 11:05
$
```

If you make a mistake entering your user name or password, or if your password has expired, the system displays the message “User authorization failure,” and you are not logged in. If you make a mistake, press RETURN and try again. If your password has expired, you need to change your password using the procedure in Section 2.3. If you have any other problems logging in, get help from the person who set up your account.

If your login is successful, the system displays a dollar sign (\$) in the left margin of your screen. The dollar sign symbol is the DCL prompt; it indicates that the system is ready to use.

2.2 Logging In to a Remote Node

If you have access to an account on a remote node, you can log in to that account from your local node and use the facilities of that remote node while remaining physically connected to your local node.

For example, to access a remote node HUBBUB on the network using the DCL command SET HOST, enter the following command:

```
$ SET HOST HUBBUB
```

You can then log in to your account on the remote node using the remote node’s login procedure. When you use the SET HOST command to log in to a remote node, you can perform any operation on the remote node as though it were your local node. Note that the remote node need not be a VMS system. If the network link cannot be established, you receive an error message.

To abort the login procedure, enter CTRL/Z at the user name or password prompt or enter CTRL/Y twice. The host system should respond with the question, "Are you repeating ^Y to abort the remote session?" Answering Y (uppercase or lowercase) aborts the remote session.

See the Reference Section for more information about the DCL command SET HOST.

NOTE: In the examples of remote operations in this chapter, proxy accounts enable users to perform operations on remote systems. Proxy accounts are one way users can access remote systems. For more information about additional ways to access remote systems, see the *VMS System Manager's Manual*.

2.3 Changing Your Password

Change your password after you log in for the first time or if your password is soon to expire. You should also change your password frequently to ensure system security.

To change your password, use the following procedure:

1. At the DCL prompt (\$), enter SET PASSWORD and press RETURN.

The system prompts you for your current password as follows:

Old password:

2. Enter your current password and press RETURN. (The system does not display what you enter.)

The system prompts you for a new password as follows:

New password:

3. Enter your new password and press RETURN.

The system prompts you to confirm your new password as follows:

Verification:

4. Enter your new password again to verify that you have entered it correctly and press RETURN.

The following example shows the series of set password prompts:

```
$ SET PASSWORD
Old password:
New password:
Verification:
$
```

NOTE: If you are managing your own system, see the *VMS System Manager's Manual* for instructions about setting up a user account and establishing a password.

2.4 Recognizing System Responses

The system responds to the commands you enter in several ways. It can execute the command. Generally, you know your command has executed successfully when the system prompt returns (by default, the dollar sign). It can execute the command and inform you in a message what it has done. It can, if execution is not successful, inform you of errors. It can even act for you, supplying values (defaults) you have not supplied yourself.

Understanding Defaults

A default is the value supplied by the operating system when you do not specify one yourself. For instance, if you do not specify the number of copies as a qualifier for the PRINT command, the system uses the default value of 1. By not explicitly stating a value, the system assumes that you have chosen the default. The VMS operating system supplies default values in several areas, including command qualifiers and parameters. The defaults used with individual commands are specified with each command's description in the *VMS DCL Dictionary*.

Looking at Informational Messages

The system responds to some commands by displaying information about what it has done. For example, when you use the PRINT command, the system displays the job identification number it assigned to the print job and shows the name of the print queue the job has entered.

```
$ PRINT MYFILE.LIS [RET]
      Job MYFILE (queue SCALE_PRINT, entry 210) started on SYS$PRINT
```

Not all commands display informational messages. Successful completion of a command is usually indicated when the dollar sign prompt returns. Unsuccessful completion is always indicated by one or more error messages.

Looking at Error Messages

If you enter a command incorrectly, the system displays an error message and prompts you for the correct command string, as the following example shows:

```
$ CAPY [RET]
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
\CAPY\
$
```

The three-part code preceding the text of the message indicates the following information:

- *DCL* means that the message is from DCL, the default command interpreter.
- *W* is a warning message.

- *IVVERB* shows the type of message. The message can be identified by the mnemonic *IVVERB* in the *VMS System Messages and Recovery Procedures Reference Volume*

You can also receive error messages during command execution if the system cannot perform the function you have requested. For example, if you type a **PRINT** command correctly, but the file you specify does not exist, the **PRINT** command informs you of the error with a message like the following:

```
$ PRINT NOFILE.DAT [RET]
%PRINT-E-OPENIN, error opening CLASS1:[MAYMON]NOFILE.DAT; as input
-RMS-E-FNF, file not found
$
```

The first message is from the **PRINT** command. It tells you it cannot open the specified file. The second message indicates the reason for the first, that is, the file cannot be found. **RMS** refers to the VMS file handling facility, Record Management Services; error messages related to file handling are generally VMS **RMS** messages.

Checking Your Current Process

If you suspect that your system is not doing what you think it should be doing, press **CTRL/T**. **CTRL/T** displays a single line of statistical information about the current process. When you press **CTRL/T** during an interactive terminal session, it momentarily interrupts the current command, command procedure, or image in order to display statistics.

Although **CTRL/T** disrupts the characters on the screen, it does not impact any procedure or editing session. To refresh the screen, press **CTR/W**. The statistical information includes node and user name, current time, current process, **CPU** usage, number of page faults, level of I/O activity, and memory usage. The following example shows a user named **BEAN** on node **GREEN** using the **EDT** editor:

```
GREEN::BEAN 13:45:02 EDT CPU=00:00:03.33 PF=778 IO=295 MEM=315
```

If you know that your system is running, and **CTRL/T** does not display statistical information, enter the **SET CONTROL=T** at the dollar sign (\$) prompt, then press **CTRL/T** again.

2.5 Getting Help

When you are logged in to the VMS operating system, you can obtain information about using the system and available commands by using the **HELP** command.

Use the following procedure to get help:

1. Enter **HELP** at the **DCL** prompt and press **RETURN**:

```
$ HELP [RET]
```

2-6 Getting Started: Interacting with VMS

HELP displays a list of topics and the **Topic?** prompt.

```
HELP
.
.(HELP message text and subtopics)
.
Topic?
```

2. To see information about one of the topics, type the topic name after the prompt.

```
Topic? NAME
```

HELP displays information about that topic. If the topic has subtopics, **HELP** lists the subtopics and displays the **Subtopic?** prompt.

```
NAME
.
.(HELP message text and subtopics)
.
NAME Subtopic?
```

3. If you want information on one of the subtopics, type the name after the prompt.

```
NAME Subtopic? Subtopic Name
```

HELP displays information about that subtopic.

```
Subtopic
Name
.
.(HELP message text and subtopics, if any)
.
```

4. If you want information on another topic, press RETURN.
5. To exit **HELP**, press RETURN until you return to the DCL prompt.

If you know the command you need information about, type **HELP** and the command name. For example, to get help about the **SHOW USERS** command enter the following command:

```
$ HELP SHOW USERS
```

HELP displays the following information:

```
SHOW
```

```
USERS
```

Displays the terminal name, username, and process identification code (PID) of either specific interactive users or all interactive users on the system.

Format:

```
SHOW USERS [username]
```

Additional information available:

Parameters Command_Qualifiers
/OUTPUT
Examples

SHOW USERS Subtopic?

If you need help but do not know what command or system topic to specify, enter the command **HELP** with the word **HINTS** as a parameter. Each task name listed in the **HINTS** text is associated with a list of related command names and system information topics.

The Reference Section contains more information about the **HELP** command.

2.6 Ending a Remote Session

You can end a remote session in two ways:

- Use the remote system's logout procedure (for example, on a VMS system, use the **LOGOUT** command).
- Press **CTRL/Y** twice to obtain the host system's prompt, which asks whether you want to abort the remote session. Answer **Y** if you want to abort the remote session. This method works regardless of the system running on the remote node.

When you end a remote session, the message “%REM-S-END, control returned to node _NODENAME::” is displayed, and you are returned to the system from which you made the remote node connection.

If the DECnet network has made intermediate connections for you and one of the intermediate systems goes down, DECnet either attempts to reroute the connection or waits a few seconds to determine whether the system will recover. If DECnet is able to recover the connection, the interruption might be so brief that you do not notice it, or it may last as long as 60 seconds. If DECnet cannot recover the connection, the remote session is ended and the message “Path lost to partner” may be displayed.

2.7 Logging Out of the System

When you finish using the system, always log out. This prevents unauthorized users from accessing your account and the system. It is also a wise use of system resources; the resources you no longer need are available for other users.

To log out, enter **LOGOUT** at the DCL prompt. For example:

```
$ LOGOUT [RET]
```

2-8 Getting Started: Interacting with VMS

The system displays a message, similar to the following message, confirming that you are logged out of the system:

```
$ LOGOUT
HARRIS logged out at 19-APR-1990 12:42:48.12
```

NOTE: You can log out of the system only when you are at the DCL prompt (\$). You cannot enter the LOGOUT command while you are compiling or executing a program, using a text editor (such as EDT or EVE), or running a utility (such as MAIL). First you must exit the program, editor, or utility. When the system displays the DCL prompt, you can log out.

To find out how much time you spent at the terminal (elapsed time), how much computer time you used (charged CPU time), and other accounting information, enter LOGOUT/FULL at the DCL prompt. For example:

```
$ LOGOUT/FULL 
```

The system displays information similar to the following:

```
SIMPSON logged out at 19-APR-1990 12:42:48.12
```

Accounting information:

Buffered I/O count:	8005	Peak working set size:	212
Direct I/O count:	504	Peak virtual size:	770
Page faults:	1476	Mounted volumes:	0
Charged CPU time:0 00:00:50.01		Elapsed time:0 02:27:43.06	

Chapter 3

The DIGITAL Command Language: Communicating with VMS

This chapter describes how to use the DIGITAL Command Language. The DIGITAL Command Language (DCL) is a limited set of English-like instructions that tell the VMS operating system to perform specific operations.

DCL commands let you do the following:

- Get information about the system
- Work with files
- Work with disks, magnetic tapes, and other devices
- Modify your work environment
- Develop and execute programs
- Provide security and ensure that resources are used efficiently

3.1 Using DCL Commands

To enter a DCL command, type the command (in uppercase or lowercase letters) at the DCL prompt (\$) and press RETURN. For example, to use the DCL command SHOW TIME, enter the following command:

```
$ SHOW TIME 
```

The system responds by displaying the current date and time and returns the DCL prompt to indicate it is ready to accept another command:

```
19-APR-1990 15:41:43  
$
```

Table 3-1 lists a few common computing tasks and the DCL commands you need to perform them. The Reference Section describes DCL commands in alphabetical order.

Table 3-1: Commonly Used DCL Commands

Task	Command
Displaying the contents of a current directory (list of files)	DIRECTORY
Making a copy of a specified file	COPY
Erasing a specified file and removing it from a directory	DELETE
Changing the name of a specified file	RENAME
Sending a specified file to a printer for printing	PRINT
Viewing and changing the contents of a text file	EDIT
Ending your VMS session	LOGOUT
Creating files or directories	CREATE
Controlling how you see the system	SET
Displaying the status of the system	SHOW
Displaying the contents of a specified file on the screen	TYPE

In addition to these English-like commands, the VMS operating system understands specific key combinations. A **key combination** is a shortcut or a way to get the system's attention while it is processing another command.

To enter a key combination, hold down the first key while you press and release the second key.

Table 3-2 describes a few key combinations. (Additional key combinations are listed in Section 3.7.)

Table 3-2: Commonly Used DCL Key Combinations

Function	Use
CRTL/C	During command entry, cancels command processing. CRTL/C is displayed as (Cancel).
CRTL/Y	Interrupts command processing. CRTL/Y is displayed as (Interrupt).
CTRL/T	Displays information about current process.

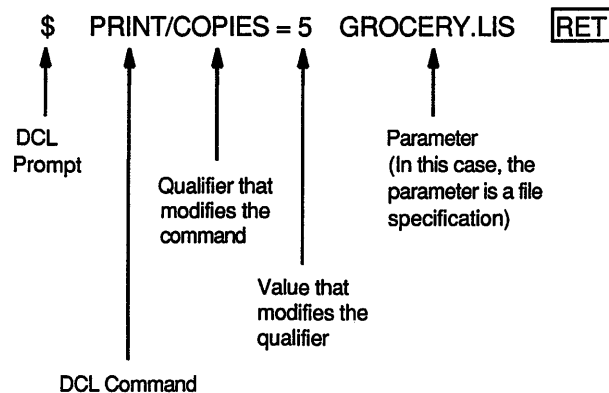
3.2 Constructing a DCL Command

Like a spoken language, DCL is made up of *words* (vocabulary) and *word order* (syntax). The following sections describe these two elements and explain how to construct a valid DCL command.

3.2.1 Vocabulary of a DCL Command

Figure 3-1 shows the general format and parts of a DCL command line:

Figure 3-1: Parts of a DCL Command Line



ZK-0950A-GE

The following sections describe the parts of a DCL command line.

DCL Prompt

The dollar sign (\$) is the DCL prompt. When you work interactively with DCL, DCL displays the prompt when it is ready to accept a command. When you write a command procedure, you must type the dollar sign at the beginning of each line.

Label

Identifies a line in a command procedure. Use labels only within command procedures, which are described in Chapter 13.

DCL Command

Specifies the name of the command.

3-4 The DIGITAL Command Language: Communicating with VMS

Parameter

Specifies what the command acts upon. You must place parameters in a specified order within the command. The DCL command descriptions in the Reference Section describe what parameter values are allowed for each command and where they must be placed. Examples of parameter values include file specifications, queue names, and logical names.

Qualifier

Modifies the action taken by the command. Some qualifiers modify the whole command, while others can modify specific parameters. Some qualifiers can accept values. The DCL command descriptions in the Reference Section indicate whether a specific qualifier can accept a value and what kind of value is acceptable.

Value

Modifies a qualifier and is often preceded by an equal sign (=). A value can be a file specification, a character string, a number, or a DCL **keyword**.

Keyword

A keyword is a word defined for use in certain specified syntax formats. You must use keywords exactly as listed in the description of the particular DCL command you want to specify. For example, SYSTEM, OWNER, GROUP, and WORLD are DCL keywords for the /PROTECTION qualifier of the SET FILE command. (A DCL keyword can also have a value.)

Wildcard character

A wildcard character is a nonalphanumeric character such as an asterisk (*) or a percent sign (%) that is used within, or in place of, a file name, file type directory name, or version number in a file specification to indicate "all" for the given field. For information about using wildcard characters with files and directories, see Chapter 4 and Chapter 5. For information about using wildcard characters with a particular DCL command, see the Reference Section.

3.2.2 Putting the Parts in Order: Syntax

Just as a spoken language depends on the order of words to create meaning, DCL requires that you put the correct elements of the command line in a specific word order. This word order or **syntax** is shown in a syntax diagram.

The following syntax diagrams show the structure of typical DCL commands:

```
$ label: command/qualifier=value
```


or

\$ label: command parameter/qualifier

3.3 Entering a DCL Command

When you enter a DCL command, some items must be entered on the command line. If you do not enter them, the system prompts you to supply the missing information.

In the following example, the **TYPE** command expects a file specification. Because a file specification is a required parameter, if you do not enter one, the system requests it. A line beginning with an underscore (**_**) means that the system is waiting for your response.

```
$ TYPE
_ File:  WATER.TXT
```

When you are prompted for an optional parameter, press **RETURN** to omit it. At any prompt, you can enter one or more of the remaining parameters and any additional qualifiers.

If you press **CTRL/Z** after a command prompt, DCL ignores the command and redisplay the DCL prompt.

Some items need not be specified on the command line. These are called defaults. When DCL does something by **default**, it assumes that you want a command to use certain values or to take certain actions without your having to explicitly specify them. In general, the values and actions are those considered typical or expected by users.

For example, if you do not specify the number of copies as a qualifier for the **PRINT** command, DCL uses the default value of 1. Unless you specify otherwise, DCL assumes that you have chosen the default. You can override this default behavior and print multiple copies of a file by specifying the following command:

```
$ PRINT/COPIES=4 MYFILE.TXT
```

DCL supplies default values in several areas, including command parameters and qualifiers. Parameter defaults are described in the following section; qualifier defaults are described in Section 3.3.5.2.

3.3.1 Rules for Entering a DCL Command

Use the following rules to enter a DCL command:

- Use any combination of uppercase and lowercase letters. The DCL interpreter translates lowercase letters to uppercase. Uppercase and lowercase characters in parameter and qualifier values are equivalent unless enclosed in quotation marks.

3-6 The DIGITAL Command Language: Communicating with VMS

- Separate the command name from the first parameter with at least one blank space.
- Separate each additional parameter from the previous parameter qualifier with at least one blank space.
- Begin each qualifier with a slash (/); the slash serves as a separator and need not be preceded by blank spaces or tabs.
- A command line can contain a maximum of 128 elements (for example, a file specification or qualifier).
- You can abbreviate a command name as long as the abbreviated name remains unique among all DCL command names.

For example, the following commands are equivalent:

```
$ PR/C=2 FORMAL_ART.TXT  
$ PRINT/COPIES=2 FORMAL_ART.TXT
```

Do not, however, abbreviate commands in command procedures.

Additional rules govern the format of commands when they are used in command procedures. See Chapter 13 for more information about using commands in command procedures.

3.3.2 Entering an Incomplete Command Line

If you do not enter all the information that the system needs to process a command, the system displays a prompt for the missing information. A line beginning with an underscore (_) means that the system is waiting for your response.

In the following example, the system displays a prompt because the name of the file is a required parameter for the TYPE command.

```
$ TYPE  
_File: WATER.TXT
```

3.3.3 Entering a Command Longer Than One Line

If you enter a command longer than one line, you can continue the command onto the next line. To continue a command line onto the next line, use the following procedure:

1. End the command line with a hyphen and press RETURN.

The system displays an underscore (_) followed by the DCL prompt (\$).

2. Enter the rest of the command line after this prompt. A line beginning with an underscore means that the system is waiting for your response, as shown in the following example:

```
$ COPY/LOG FORMAT.TXT,FIGURE.TXT,ARTWORK.TXT -  
_ $ SAVE.TXT
```

Note that you must include the appropriate spaces between command names, parameters, and so on. Pressing RETURN after the hyphen does not add a space.

3.3.4 Entering Parameters

DCL supplies default values for some command parameters. The parameters accepted by a command as well as the specific command parameter defaults supplied by DCL are described in each command description in the Reference Section.

The following rules apply when specifying parameters in a command line:

- Square brackets ([]) in commands indicate optional items. For example, you do not have to enter a file specification in the following command:

```
DIRECTORY [file-spec]
```

- Anything not enclosed in square brackets is required. For example, you must enter a device name in the following command:

```
SHOW PRINTER device-name
```

- In general, precede an output file parameter with an input file parameter. For example, to copy the input file, `LISTS.TXT`, to the output file, `FORMAT.TXT`, enter the following command:

```
$ COPY LISTS.TXT FORMAT.TXT
```

- A parameter can be one item or a series of items. If you enter a series of items, separate them with commas (,) or plus signs (+). Any number of spaces or tab characters can precede or follow a comma or a plus sign. Note that some commands regard the plus sign as a concatenator, not as a separator. The parameter section of each DCL command description in the Reference Section describes how each command interprets commas and plus signs.

The following command syntax line shows that you can optionally enter a list of files as the parameter:

```
DELETE file-spec[,...]
```

The following example shows how to specify a list of parameters. Here, three files are copied to a fourth file. The three file specifications—`PLUTO.TXT`, `SATURN.TXT`, and `EARTH.TXT`—constitute the first parameter. `PLANETS.TXT` is the second parameter.

```
$ COPY PLUTO.TXT, SATURN.TXT, EARTH.TXT PLANETS.TXT
```

3.3.5 Entering Qualifiers

The qualifiers accepted by a command are described in each command description in the Reference Section. The DCL command description also indicates whether a qualifier accepts a value and what kind of value is required.

You can abbreviate any qualifier name as long as the abbreviated name remains unique among all qualifier names for the same command. However, you should not abbreviate commands in command procedures.

Commands have default qualifiers; you do not have to specify a qualifier unless it is different from the command default. The following sections describe types of qualifiers and qualifier defaults. The Reference Section contains default information for specific commands.

3.3.5.1 Types of Qualifiers

The three types of qualifiers are as follows:

- **Command qualifiers**—A command qualifier modifies a command and can appear anywhere in the command line. However, it is a good practice to place the qualifier after the command name. If you are specifying multiple qualifiers, you should place a command qualifier after other command qualifiers that follow the command name.

In the following example, `/QUEUE` is a command qualifier. The files `SATURN.TXT` and `EARTH.TXT` are queued to the print queue `LN03_PRINT`.

```
$ PRINT/QUEUE=LN03_PRINT SATURN.TXT,EARTH.TXT
```

- **Positional qualifiers**—A positional qualifier can modify commands or parameters and has different meanings depending on where you place it in the command string. If you place a positional qualifier after the command but before the first parameter, it affects the entire command string. If you place a positional qualifier after a parameter, it affects only that parameter.

In the following example, the first `PRINT` command requests two copies of the files `SPRING.SUM` and `FALL.SUM`. The second `PRINT` command requests two copies of the file `SPRING.SUM`, but only one copy of `FALL.SUM`.

```
$ PRINT/COPIES=2 SPRING.SUM,FALL.SUM
$ PRINT SPRING.SUM/COPIES=2,FALL.SUM
```

- **Parameter qualifiers**—A parameter qualifier can be used only with certain types of parameters, such as input files and output files.

For example, the `BACKUP` command accepts several parameter qualifiers that apply only to input and output file specifications. In the following example, the `/CREATED` and `/BEFORE` qualifiers, which can be specified only

with input files, select specific input files for the backup operation. (For this example, multiple copies of the file MYFILE.TXT exist. Only those versions that were created before April 19, 1990 are selected for the backup operation.)

```
$ BACKUP MYFILE.TXT/CREATED/BEFORE=19-APR-1990 NEWFILE.TXT
```

3.3.5.2 Qualifier Defaults

When you omit a specific qualifier from the command line, the system responds with default behavior. For example, when you delete a file with the **DELETE** command, the system by default does not request confirmation of each delete operation. However, by specifying the **DELETE/CONFIRM** command, you can override that default behavior and request that you be prompted for confirmation before each file is deleted.

You can specify qualifiers in several ways. The qualifier syntax required by a specific DCL command is given in the command descriptions in the Reference Section. The following paragraphs explain the syntax used to describe qualifiers and their defaults:

- **Qualifiers with positive and negative forms**—These qualifiers have a value of true or false. You indicate a true value by naming the qualifier. Negate the qualifier by inserting the prefix **NO**.

For example, the **/CONFIRM** qualifier can be expressed positively or negatively. If you omit the qualifier from the command line, the default action is **/NOCONFIRM**. The syntax for the **/CONFIRM** qualifier is given in a DCL command description as follows:

```
/CONFIRM
/NOCONFIRM (default)
```

- **Qualifiers that require values**—If you use a qualifier that accepts a value, you must specify a value. If you omit the qualifier completely, the default value is applied. For example, if you use the **/COPIES** qualifier, you must provide a numeric value. If you omit the **/COPIES** qualifier, the default is **/COPIES=1**. The syntax for the **/COPIES** qualifier is given in a DCL command description as follows:

```
/COPIES=n
```

If the qualifier accepts a list of values, you must enclose the values in parentheses and separate them with commas as follows:

```
$ DELETE/ENTRY=(230,231) LN03_PRINT
```

The command deletes jobs 230 and 231 from the queue LN03_PRINT.

3-10 The DIGITAL Command Language: Communicating with VMS

- **Qualifiers that accept value and positive/negative combinations**—Some qualifiers combine value and positive/negative characteristics so that the qualifier both accepts a value and allows you to negate the qualifier by inserting the prefix **NO**. For example, the **SET TERMINAL** command permits the following choices for the **/PARITY** qualifier:

```
$ SET TERMINAL/PARITY=EVEN
$ SET TERMINAL/PARITY=ODD
$ SET TERMINAL/NOPARITY
```

- **Qualifiers that affect command execution only if specified**—The qualifier has no corresponding default. For example, the **/BY_OWNER** qualifier does not affect the command if it is not specified. The syntax for the **/BY_OWNER** qualifier is given in a DCL command description as follows:

```
/BY_OWNER
```

- **Qualifiers that override other qualifiers**—Sometimes a command has a qualifier that is automatically applied as a default. Other qualifiers are available to override the default qualifier.

For example, the **/BRIEF** qualifier is applied by default when you specify the **DIRECTORY** command. That is, the **DIRECTORY** command generates a listing that includes only the file name, file type, and version number of each file in the directory. You must specify the **/FULL** qualifier to generate a listing that includes the file name, file type, and version number as well as the number of blocks used, the date of the file's creation, the date the file was last backed up, and so on.

Some commands contain conflicting qualifiers that cannot be specified in the same command line. If you use incompatible qualifiers, the system usually displays an error message. The command descriptions in the Reference Section indicate which qualifiers cannot be used together.

3.4 Recalling Commands

At DCL level, you can recall previously typed command lines and avoid the inconvenience of retyping long command lines. The recall buffer holds up to 20 previously entered commands. Once a command is displayed, you can reexecute or edit it.

You can display your previously entered commands by using one of the following methods:

- Pressing **CTRL/B**
- Using up and down arrow keys
- Entering the **RECALL** command

Pressing **CTRL/B** once recalls the previous command line. Pressing **CTRL/B** again recalls the line before the previous line, and so on to the last saved command line.

Pressing the up and down arrow keys recalls the previous and successive command, respectively. Press the arrow keys repeatedly to move through the commands.

To examine up to 20 previously typed command lines, type **RECALL/ALL**. Following is a sample display generated by typing **RECALL/ALL**:

```
$ RECALL/ALL
1 SET DEFAULT DISK2:[MARSHALL]
2 EDIT ACCOUNTS.COM
3 PURGE ACCOUNTS.COM
4 DIRECTORY/FULL ACCOUNTS.COM
5 COPY ACCOUNTS.COM [ .ACCOUNTS] *
6 SET DEFAULT [ .ACCOUNTS]
```

Having reviewed the available commands, you can recall a particular command line by typing **RECALL** and the number of the desired command. The following example shows how to recall the fourth command line:

```
$ RECALL 4
```

After you press **RETURN**, the fourth command in the list is displayed at the DCL prompt. (The **RECALL** command itself is not placed in the buffer.)

You can also follow **RECALL** with the first characters of the command line you want to display. **RECALL** scans the previous command lines (beginning with the most recent one) and returns the first command line that begins with the characters you typed. For example, to recall a previously entered command, **EDIT ACCOUNTS.COM**, enter the following command:

```
$ RECALL E
```

After you press **RETURN**, the system displays the following command line:

```
$ EDIT ACCOUNTS.COM
```

TIP: If you are running a utility or an application program that uses VMS screen management software, you can use **CTRL/B** and the up and down arrow keys to perform command recall. Line editing must be enabled. Some utilities that have this feature are **MAIL**, **DEBUG**, **SHOW CLUSTER**, the System Dump Analyzer (**SDA**), and the **VAXTPU** editor.

To erase the contents of the recall buffer, enter the **RECALL** command with the **ERASE** qualifier. For example:

```
$ RECALL/ERASE
```

3.5 Entering Dates and Times as Values

Certain commands and qualifiers accept date and time values. You can specify these values in one of the following formats:

- Absolute time
- Delta time
- Combination time (combines absolute and delta time formats)

The DCL command descriptions in the Reference Section indicate the time formats accepted by individual commands and qualifiers.

3.5.1 Absolute Time

Absolute time is a specific date or time of day. The format for an absolute time is as follows:

[dd-mmm-yyyy][:][hh:mm:ss.cc]

The fields are as follows:

Field	Meaning
dd	Day of the month; an integer in the range 1 to 31
mmm	Month; JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC
yyyy	Year; an integer
hh	Hour; an integer in the range 0 to 23
mm	Minute; an integer in the range 0 to 59
ss	Seconds; an integer in the range 0 to 59
cc	Hundredths of a second; an integer in the range 0 to 99

You can truncate the date or the time on the right. However, if you are specifying both date and time, you must include a colon between them. The date must contain at least one hyphen. You can omit any of the fields within the date and time as long as you include the punctuation marks that separate the fields. A truncated or omitted date field defaults to the corresponding fields for the current date. A truncated or omitted time field defaults to zero. If you specify a past time in a command that expects the current or a future time, the current time is used.

You can also specify an absolute time as one of the following keywords:

Keyword	Meaning
TODAY	The current day, month, and year at 00:00:00.0 o'clock
TOMORROW	00:00:00.00 o'clock tomorrow
YESTERDAY	00:00:00.00 o'clock yesterday

The following table shows some examples of absolute time specifications:

Time Specification	Result
19-APR-1990:13	1 P.M. on April 19, 1990
19-APR	Midnight at the beginning of April 19 this year
15:30	3:30 P.M. today
19-	Midnight on the 19th day of the current year and month
19-::30	12:30 A.M. on the 19th of this month

3.5.2 Delta Time

Delta time is an offset (a time interval) from the current date and time to a time in the future. The general format of a delta time is as follows:

[dddd-][hh:mm:ss.cc]

The fields are as follows:

Field	Meaning
dddd	Number of days; an integer in the range 0 to 9999
hh	Number of hours; an integer in the range 0 to 23
mm	Number of minutes; an integer in the range 0 to 59
ss	Number of seconds; an integer in the range 0 to 59
cc	Number of hundredths of seconds; an integer in the range 0 to 99

You can truncate a delta time on the right. If you specify the number of days, include a hyphen. You can omit fields within the time as long as you include the punctuation that separates the fields. If you omit the time field, the default is zero.

The following table shows some examples of delta time specifications:

Time Specification	Result
3-	3 days from now (72 hours)
3	3 hours from now
:30	30 minutes from now
3-:30	3 days and 30 minutes from now
15:30	15 hours and 30 minutes from now

3.5.3 Combination Time

To combine absolute and delta time, specify an absolute time plus or minus a delta time. The format for combination time is as follows:

"[absolute time][+delta time]"

or

[absolute time][-delta time]

The variable fields and default fields for absolute and delta time values are the same as those described in the preceding sections. The delta time value must always be preceded by a plus sign (+) or minus sign (-). (Note that the minus sign is the same keyboard key as the hyphen.) Whenever a plus sign precedes the delta time value, enclose the entire time specification in quotation marks. In addition, you can omit the absolute time value. If you do, the delta time is offset from the current date and time.

The following table shows some examples of combination time specifications:

Time Specification	Result
"+5"	5 hours from now
"+:5"	5 minutes from now
-:5	Current time minus 5 minutes
-1-00	Current time minus 1 day. The minus sign (-) indicates a negative offset. The dash (-) separates the day from the time field.

If a qualifier is described as a value that can be expressed as an absolute time, a delta time, or a combination of the two, you must specify a delta time as if it were part of a combination time. For example, to specify a delta time value of five minutes from the current time, use "+:5" (not "0-0:5").

3.6 Defining Terminal Keys

Using key definitions, you can customize your keyboard so that you can enter DCL commands with fewer keystrokes. A **key definition** is a string of characters that you assign to a particular terminal key. When a key is defined, you can press it instead of typing the string of characters. A key definition usually contains all or part of a command line. When you press a defined key, the command is either displayed on your terminal or executed.

Some definable keys are automatically enabled for definition (like keys PF1 through PF4 and keys F17 through F20 on VT200- and VT300-series terminals). However, before you can define other keys, including keypad 0 (KP0) through KP9 and the keypad keys PERIOD, COMMA, MINUS, and ENTER, you must enable them for definition by entering either the SET TERMINAL/APPLICATION_KEYPAD or the SET TERMINAL/NUMERIC command.

For a complete list of definable keys and for more information on how to create key definitions, see the description of the DCL command DEFINE/KEY in the Reference Section.

3.7 Summary of Key Combinations

Table 3-3 lists and describes the key combinations that let you enter and edit DCL commands.

Table 3-3: Keys That Execute Terminal Functions

Key	Function
Keys That Enter DCL Commands	
CTRL/Z and F10 ¹	Signals the end of the file for data entered from the terminal. CTRL/Z is displayed as "Exit."
RETURN	Sends the current line to the system for processing. (On some terminals, the RETURN key is labeled CR.) Before a terminal session, RETURN initiates a login sequence.

¹This key is available only on an LK201 keyboard.

(continued on next page)

Table 3-3 (Cont.): Keys That Execute Terminal Functions

Key	Function
Keys That Interrupt DCL Commands	
CTRL/C and F6 ¹	During command entry, cancels command processing. CTRL/C is displayed as "Cancel."
CTRL/T	<p>Momentarily interrupts terminal output to display a line of statistical information about the current process. This display includes your node and user name, the time, the name of the image you are running, and information about system resources used during your current terminal session.</p> <p>You can also use the CTRL/T key to determine whether the system is operating. CTRL/T does not return information if the system is temporarily unresponsive or if your terminal is set to NOBROADCAST. In order to use CTRL/T, SET CONTROL=T must be enabled either in the system login command procedure or by you, either interactively or in your login command procedure.</p>
CTRL/Y	<p>Interrupts command processing. CTRL/Y is displayed as "Interrupt." You can disable CTRL/Y with the command SET NOCONTROL=Y.</p> <p>Under most conditions, CTRL/Y returns you to the DCL prompt. The program running is still active. You can enter any built-in command then continue the program with the CONTINUE command. (Press CTRL/W to refresh the screen after you enter the CONTINUE command.)</p>
Keys That Recall Commands	
CTRL/B and Up arrow	Recalls up to 20 previously entered commands.
Down arrow	Displays the next line in the recall buffer.

¹This key is available only on an LK201 keyboard.

(continued on next page)

Table 3-3 (Cont.): Keys That Execute Terminal Functions

Key	Function
Keys That Control Cursor Position	
<X>, DELETE	Deletes the last character entered at the terminal. (On some terminals, the DELETE key is labeled RUBOUT.) The DELETE key also works when line editing is disabled.
CTRL/A, F14 ¹	Switches between overstrike and insert mode. The default mode (as set with the SET TERMINAL/LINE_EDITING command) is reset at the beginning of each line.
CTRL/D and Left arrow	Moves the cursor one character to the left.
CTRL/E	Moves the cursor to the end of the line.
CTRL/F and Right arrow	Moves the cursor one character to the right.
CTRL/H, BACKSPACE, and F12 ¹	Moves the cursor to the beginning of the line.
CTRL/I and TAB	Moves the cursor to the next tab stop on the terminal. The system provides tab stops at every eighth character position on a line. Tab settings are hardware terminal characteristics that, in general, you can modify. The TAB key also works when line editing is disabled.
CTRL/J, LINEFEED, and F13 ¹	Deletes the word to the left of the cursor.
CTRL/K	Advances the current line to the next vertical tab stop.
CTRL/L	Causes the cursor to go to the beginning of the next page. This use of this key is ignored when line editing is enabled.
CTRL/R	Repeats the current command line and leaves the cursor positioned where it was when you pressed CTRL/R.
CTRL/U	Cancels the current input line.
CTRL/V	Turns off some of the line editing function keys. For example, if you press CTRL/V followed by CTRL/D, a CTRL/D is generated instead of the cursor moving left one character. CTRL/D is a line terminator at DCL level. When combined with CTRL/V, characters that are not line terminators have no effect. Examples are CTRL/H and CTRL/J. However, certain control keys, such as CTRL/U, retain their line editing functions.
CTRL/X	Cancels the current line and deletes data in the type-ahead buffer.
F7, F8, F9, F11	Reserved by Digital.

¹This key is available only on an LK201 keyboard.

Table 3-3 (Cont.): Keys That Execute Terminal Functions

Key	Function
Keys That Control Screen Display	
CTRL/O	Alternately suspends and continues display of output to the terminal. CTRL/O is displayed as "Output off" and "Output on."
CTRL/S	Suspends terminal output until CTRL/Q is pressed.
CTRL/Q	Resumes terminal output suspended by CTRL/S.
HOLD SCREEN ¹ and NO SCROLL ²	Suspends terminal output until the key is pressed again.

¹This key is available only on an LK201 keyboard.

²This key is available only on a VT100 keyboard.

Chapter 4

Files: Storing Information

A file is a unit the VMS operating system uses to store human-readable and machine-readable data. This chapter describes the following tasks you can perform with files locally, and if you have sufficient privileges, over a DECnet-VAX network.

- Understanding file names and file specifications
- Using wildcard characters to access files
- Creating files
- Modifying files
- Copying files
- Renaming files
- Displaying the contents of text files
- Deleting files
- Protecting a file from other users
- Printing files

The descriptions of the DCL commands in the Reference Section describe specific file operations you can perform locally and over the network.

NOTE: In the examples of remote operations in this chapter, proxy accounts enable users to perform operations on remote systems. Proxy accounts are one way users can access remote systems. For more information, about additional ways to access remote systems, see the *VMS System Manager's Manual*.

4.1 Understanding File Names and Specifications

When you create a file, you must specify certain information so that the system can locate and identify the file. A complete file specification has the following format:

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

You must provide a complete file specification if you are performing file operations over a network.

When you name a file, you do not have to include all the elements of a complete file specification. However, you must include a file name or file type to identify it to both the system and you. The system automatically assigns a version number unless you specify one. To name a file, use the following format:

`filename.type;version`

Use the following rules to specify the elements of a file specification:

- Give the file a name that is meaningful to you. The file name can be up to 39 characters chosen from the letters A through Z (uppercase or lowercase), the numbers 0 through 9, an underscore (`_`), a hyphen (`-`), or a dollar sign (`$`).
- Do not use a hyphen as the first or last character in the file name.
- Do not begin a file name with a dollar sign; you can use a dollar sign only within the file name.
- A file type, which identifies the kind of file, can be from 0 through 39 characters.
- Precede a file type by a period.
- Precede version numbers with a semicolon or a period. (When the system displays file specifications, it displays a semicolon in front of the file version number.)

Including a file type is optional. With certain commands, if you omit the file type, the system applies a default value. Table 4-1 lists some of the more common default file types used by DCL commands. It also lists the default file types for some high-level language source programs.

Table 4-1: Default File Types

File Type	Contents
Default File Types for DCL Commands	
CLD	Command description file
COM	Command procedure file
DAT	Data file
DIS	Distribution list file for the MAIL command
DIR	Directory file
EDT	Startup command file for the EDT editor
EXE	Executable program image file created by the linker
HLP	Input source file for help libraries
JOU	Journal file created by the EDT editor
LIS	Listing file created by a language compiler or assembler; default input file for the PRINT and TYPE commands
LOG	Batch job output file
MAI	MAIL message file
MEM	Output file created by DIGITAL Standard Runoff (DSR)
OBJ	Object file created by a language compiler or assembler
RNO	Input source file for DIGITAL Standard Runoff
SIX	Sixel graphic file
SYS	System image
TJL	Journal file created by the VAXTPU and ACL editors
TMP	Temporary file
TPU	Command file for the VAXTPU editor
TXT	Input file for text libraries or MAIL command output

(continued on next page)

4-4 Files: Storing Information

Table 4-1 (Cont.): Default File Types

File Type	Contents
Default File Types for Language Source Programs	
ADA	Input source file for the VAX Ada compiler
BAS	Input source file for the VAX BASIC compiler
B32	Input source file for the VAX BLISS-32 compiler
C	Input source file for the VAX C compiler
COB	Input source file for the VAX COBOL compiler
FOR	Input source file for the VAX FORTRAN compiler
MAR	Input source file for the VAX MACRO compiler
PAS	Input source file for the VAX Pascal compiler
PLI	Input source file for the VAX PL/I compiler

In addition to a file name and file type, every file has a version number. Version numbers are decimal numbers from 1 to 32,767 that differentiate versions of a file. When you initially create a file, the system assigns it a version number of 1.

You may have several versions of a file. Unless you specify a version number, the system uses the highest existing version number of that file. When you modify that file, the system saves the original file and produces a modified output file. By default, this output file has the same name and type as the original, but the version number is incremented by one.

Version numbers must be preceded with a semicolon or a period. When the system displays file specifications, it generally displays a semicolon in front of the file version number.

4.2 Using Wildcards with Files

Use wildcard characters to apply a DCL command to multiple files rather than to one file at a time. The command applies to all files that match the portion of the file specification entered.

With many DCL commands you can use an asterisk (*) or a percent sign (%) as a wildcard in directory names, file names, and file types. You can also use the asterisk, but not the percent sign, in version numbers.

Many examples in this chapter show the use of wildcard characters in file operations. The use of wildcard characters in DCL commands varies with the individual command. For more information about using wildcards with a particular DCL command, see the Reference Section.

4.2.1 The Asterisk (*) Wildcard Character

Use the asterisk wildcard character to match the following:

- An entire field, or a portion of it, in the directory, file name, and file type fields
- The entire version number field, but not a portion of it

4.2.2 The Percent (%) Wildcard Character

Use the percent sign wildcard character as a substitute for any single character in a file specification. You can use the percent sign in the directory, file name, and file type fields. You cannot, however, use the percent sign in the version number field.

The following example displays the latest versions of all DAT files whose names begin with DISTRICT:

```
$ TYPE [JONES.TAXES.PROPERTY]DISTRICT%.DAT
```

This display would include the files DISTRICT1.DAT, DISTRICT2.DAT, and DISTRICT3.DAT. The file DISTRICT4_5.DAT would not be displayed because it has more than one character after DISTRICT, nor would the file DISTRICT.DAT be displayed. The percent sign replaces one character position in a field, but there must be a character to replace.

4.3 Creating and Modifying Files

The most versatile interactive tool for creating and modifying text files is the interactive text editor. EVE and EDT are two text editors that are included in VMS; other text editors may also be available on your system.

You can also create and modify files by using the DCL commands CREATE, COPY, and RENAME. The following sections describe how to create and modify files using these commands.

4.3.1 Creating Files

The CREATE command creates a text file. For example, to create a file named POUND.LIS, enter the CREATE command and then type lines of text:

```
$ CREATE POUND.LIS
Tag #23, Elmer Doolittle, notified
Tag #37, James Watson, notified
No tag, light brown, 30 lbs., looks part beagle
CTRL/Z
```

Pressing CTRL/Z signals the end of the file and returns you to DCL command level. You cannot modify a file with the CREATE command; after you have pressed RETURN, you cannot return to a previous line to modify a word. You

4-6 Files: Storing Information

must use a text editor such as EDT or EVE to modify a file created with the CREATE command.

4.3.2 Copying Files

The COPY command duplicates the contents of an existing file in a new file. For example, to copy FEES.DAT to RECORDS.DAT, enter the following command:

```
$ COPY FEES.DAT RECORDS.DAT
```

The COPY command can duplicate many files at a time. For example, to copy all TXT files in the default directory to another directory, enter the following command:

```
$ COPY *.TXT;* [SAVETEXT]*.*;*
```

Concatenating Files

The COPY command can concatenate files. For example, to append FEES1.DAT to FEES.DAT (forming a new version of FEES.DAT) in your default directory, enter the following command:

```
$ COPY FEES.DAT,FEES1.DAT FEES.DAT
```

Copying Files from a Remote Node to Your Node

Use the COPY command to copy files from another node to your node. For example, to copy the latest version of all files in DISK2:[PUBLIC] on node CHAOS to files with the same names in your default directory, enter the following command:

```
$ COPY CHAOS::DISK2:[PUBLIC]*.* *
```

Copying Files from Your Node to a Remote Node

Use the COPY command to copy files from your node to another node. For example, to copy the latest version of all files in your default directory to files with the same names in the directory DISK2:[STAFF_BACKUP] on node CHAOS, enter the following command:

```
$ COPY *.* CHAOS::DISK2:[STAFF_BACKUP]
```

If you receive a protection violation or DECnet-VAX error message when you attempt to copy a file across systems, you have two recourses:

- If the file is yours, you can use MAIL to send it to a user account on the other node.
- You can follow the node name in the file specification with an access control string.

Use the /SINCE qualifier with the COPY command to select only those files that meet the specified criterion. For example, to copy to the default directory only those files in the directory [JONES.LICENSES.DOG] that have been modified since April 19, 1990, enter the following command:

```
$ COPY/SINCE=19-APR-1990/MODIFIED [JONES.LICENSES.DOG]*.* *
```

4.4 Renaming Files

Use the RENAME command to give the file a new name and optionally to locate it in a different directory. For example, to give the file FEES.DAT the new name RECORDS.DAT and move it from the default directory to another directory, enter the following command:

```
$ RENAME FEES.DAT;4 [SAVETEXT]RECORDS.DAT
```

Note that after being renamed, the file FEES.DAT;4 no longer exists in the default directory. When you use the RENAME command, the input and output locations must be on the same device.

4.5 Displaying the Contents of Files

To display the contents of a file on your screen, enter the TYPE command and the file name at the DCL prompt. For example, to display the latest version of the file STAFF_VACATIONS.TXT, enter the following command:

```
$ TYPE STAFF_VACATIONS.TXT
```

You do not have to specify the version number in the file specification because the system displays the latest version of a file by default.

Displaying a File on a Remote Node

To display the contents of a file on a remote node, include the node name, disk, and directory in the file specification. For example, to display the file COMPANY_HOLIDAYS.TXT, which is located on remote node CHAOS, enter the following command:

```
$ TYPE CHAOS::DISK2:[PUBLIC]COMPANY_HOLIDAYS.TXT
```

Displaying Files with Wildcards

You can use the asterisk wildcard (*) to display all versions of a specific file. For example, to display all versions of the file LOGIN.COM in the directory [JONES], enter the following command:

```
$ TYPE [JONES]LOGIN.COM;*
```

To display all versions and all file types of all files that begin with the word *STAFF* in the directory [JONES], enter the following command:

```
$ TYPE [JONES]STAFF*.*;*
```

Displaying More Than One File

If more than one file is listed in the `TYPE` command, the files are displayed in the order specified; if wildcard characters are used, the files are displayed in alphabetical order.

To stop the scrolling of the text on the screen temporarily, press the `HOLD SCREEN` key (F1 on VT200- and VT300-series terminals); to resume scrolling, press the `HOLD SCREEN` key again. To stop the display and return to DCL command level, press `CTRL/Y` or `CTRL/O`.

If you specify the `/PAGE` qualifier to the `TYPE` command, you can view one screen at a time. The system prompts you to press `RETURN` when you want to see the next screen.

TIP: By invoking an interactive text editor (for example, `EVE` or `EDT`) with the `/READ_ONLY` qualifier, you can use interactive editing commands to move around in a file and search for specific sequences of characters. The `/READ_ONLY` qualifier prevents you from modifying the file as you display it.

4.6 Deleting Files

The `DELETE` command removes files from directories and releases the disk space they occupy for use by other files. When you use the `DELETE` command, you must specify a version number or the asterisk wildcard character as a version number in each file specification. For example, to delete version 17 of the file `POUND.LIS`, enter the following command:

```
$ DELETE POUND.LIS;17
```

To delete versions 16 and 17 of the file `POUND.LIS`, enter the following command:

```
$ DELETE POUND.LIS;16,;17
```

To delete all versions of the file `POUND.LIS`, enter the following command:

```
$ DELETE POUND.LIS;*
```

When you delete many files with wildcard characters, you might want to confirm each deletion by using the `/CONFIRM` qualifier. For example, to confirm the deletion of all the files in the subdirectory `[JONES.LICENSES.DOG]`, enter the following command:

```
$ DELETE/CONFIRM *.*;*
DISK1:[JONES.LICENSES.DOG]FEES.DAT;4, delete? [N]: Y
DISK1:[JONES.LICENSES.DOG]FEMALE.LIS;6, delete? [N]: Y
DISK1:[JONES.LICENSES.DOG]MALE.LIS;3, delete? [N]: N
DISK1:[JONES.LICENSES.DOG]POUND.LIS;17, delete? [N]: Y
```

Similarly, you might want to display the names of files as they are deleted. To do this, specify the /LOG qualifier with the DELETE command. For example, if you enter the command in the following example, the system displays the names of the files after they are deleted:

```
$ DELETE/LOG *.LIS;*
_%DELETE-I-FILDEL, DISK1:[JONES.LICENSES.DOG]FEMALE.LIS;6 deleted (35 blocks)
_%DELETE-I-FILDEL, DISK1:[JONES.LICENSES.DOG]MALE.LIS;3 deleted (5 blocks)
_%DELETE-I-FILDEL, DISK1:[JONES.LICENSES.DOG]POUND.LIS;17 deleted (9 blocks)
```

The PURGE command deletes all except the latest version of the specified file (or all files) in the default directory or any other specified directory. Purging old versions of files after updating them enables you to retain more free space on your disk.

For example, to purge all except the latest two versions of each file in your default directory, enter the following command:

```
$ PURGE/KEEP=2
```

4.7 Protecting a File from Other Users

To prevent other users from accessing your files, you can set protection or modify the access control list (ACL) of your files. To set protection or modify the ACL of a file, you must own the file, have control access to the file, or have GRPPRV, SYSPRV, BYPASS, or READALL privilege.

NOTE: To protect a file completely, you must apply the same or greater protection to the directory in which the file resides. See Chapter 5 for information on directory protection.

4.7.1 Default File Protection

A new file receives default UIC-based protection and the default **access control list entries** (if any) of its parent directory. (Access control list entries (ACEs) may specify identifiers and the access rights to be granted or denied the holders of the identifiers, defaults protection for directories, or security alarm details.)

A renamed file's protection is unchanged. A new version of an existing file receives the UIC-based protection and ACL of the previous version. (Use the /PROTECTION qualifier of the BACKUP, COPY, CREATE, and SET FILE commands to override the default UIC-based protection.)

You can use either of the following methods to override the default UIC-based protection given to new files:

- **Default UIC protection**—The operating system provides each process with a default UIC-based protection of (S:RWED,O:RWED,G:RE,W). This indicates that SYSTEM users and the owners of objects have full access to the object, users in the same UIC group as the object owner have read and execute access to the object, and all other users are denied access to the object.

4-10 Files: Storing Information

To change the default protection for files that you create, invoke the **SET PROTECTION** command with the **/DEFAULT** qualifier. For example, if you enter the following command in your login command procedure, you grant all processes read and execute access to any files that you create. (Remember that you must execute the login command procedure for this command to execute.)

```
$ SET PROTECTION = (S:RWED,O:RWED,G:RE,W:RE) /DEFAULT
```

- **Default ACL protection**—You can override default UIC protection for specified directories or subdirectories by placing a default protection ACE in the ACL of the appropriate directory file. The default protection specified in the ACE is applied to any new file created in the specified directory or subdirectory of the directory. The following ACE, which must be in the ACL of a directory file, specifies that the default protection for that directory and the directory's subdirectories allow system and owner processes full access, group processes read and execute access, and world users no access.

```
(DEFAULT_PROTECTION, S:RWED, O:RWED, G:RE, W:)
```

To specify a default identifier ACE to be copied to the ACL of any file subsequently created in the directory, specify the **DEFAULT** option in the directory file's identifier ACL. For example, the following ACE, applied to a directory file, denies network users access to all files created in the directory:

```
(IDENTIFIER=NETWORK, OPTIONS=DEFAULT, ACCESS=NONE)
```

4.7.2 Explicit File Protection

You can explicitly specify UIC-based protection for a new file with the **/PROTECTION** qualifier (valid with the **BACKUP**, **COPY**, and **CREATE** commands) as shown in the following example:

```
$ CREATE MAST12.TXT /PROTECTION=(S:RWED,O:RWED,G,W)
```

You can change the UIC-based protection on an existing file with the **SET PROTECTION** command. For example, to change the UIC-based protection on the file **MAST12.TXT**, enter the following command:

```
$ SET PROTECTION=(S:RWED,O:RWED,G,W) MAST12.TXT
```

After a file is created and you have created an ACL for the file, you can modify the ACL and add as many ACEs to the ACL as you want. The protection specified by the ACL overrides the file's UIC protection.

4.8 Printing Files

To print a file or files, use the PRINT command. For example, to place a print job containing three files in the default print queue, SYS\$PRINT, enter the following command:

```

$ PRINT POUND,MALE,FEES.DAT
Job POUND (queue SYS$PRINT, entry 202) started on SYS$PRINT

```

The file types of the files named in the PRINT command default to LIS or the last explicitly named file type; thus, the preceding example queues POUND.LIS, MALE.LIS, and FEES.DAT to SYS\$PRINT. The system displays the job name (POUND), the queue name (SYS\$PRINT), and the job number (202). The system also indicates whether the job has started or is pending. By default, the job name is the name of the first (or only) file specification in the PRINT command. After a job is submitted to a queue, you reference it using the job number. After the job is queued, it will be printed when no other jobs precede it in the queue and when the printer is physically ready to print.

A print queue can execute only one job at a time. Print jobs are scheduled for printing according to their priority, and the job with the highest priority is printed first. If more than one job exists with the same priority, the smallest job is usually printed first. Jobs of equal size having the same priority are selected for printing according to their submission time.

4.8.1 Displaying Queue Information

The default print queue, SYS\$PRINT, is usually initialized and started as part of the site-specific system startup procedure. To display the queues that are initialized at your site, enter the SHOW QUEUE command as follows:

```

$ SHOW QUEUE

```

To display the status of your print jobs, enter the SHOW ENTRY command as follows:

```

$ SHOW ENTRY

```

The system display the following list:

Jobname	Username	Entry	Blocks	Status
-----	-----	-----	-----	-----
POUND	JONES	202	38	Printing

On printer queue SYS\$PRINT

To see jobs queued by other users, specify the USERNAME parameter to the SHOW ENTRY command.

4.8.2 Stopping and Deleting a Print Job

To stop a print job and delete it from the print queue, enter the **entry-number** parameter to the **DELETE/ENTRY** command. For example, to delete entry 202, enter the following command:

```
$ DELETE/ENTRY=202
```

4.8.3 Printing a File on Another Node

To print a file on another system, copy that file to the remote node and specify the **/REMOTE** qualifier to the **PRINT** command. For example, to copy the file **COMPANY_HOLIDAYS.TXT** from your local node to the remote node **CHAOS** and queue the file for printing to the default system print queue (**SY\$PRINT**) on node **CHAOS**, enter the following commands:

```
$ COPY COMPANY_HOLIDAYS.TXT CHAOS"JONES PANDEMONIUM":DISK2:[JONES]*  
$ PRINT/REMOTE CHAOS::DISK2:[JONES]COMPANY_HOLIDAYS.TXT
```

In the previous example, an access control string was specified to indicate that you are authorized to copy files to the directory **[JONES]** on node **CHAOS**. However, if you have a proxy account on that remote node, the asterisk wildcard at the end of the file specification in the previous command instructs the system to duplicate the file name **COMPANY_HOLIDAYS.TXT** when that file is copied to the remote node.

NOTE: Not all qualifiers to the **PRINT** command are compatible with the **/REMOTE** qualifier. For example, you cannot queue a job to a specific print queue; all jobs are queued to the default system print queue (**SY\$PRINT**). See the description of the **/REMOTE** qualifier to the **DCL** command **PRINT** in the Reference Section for a list of **PRINT** command qualifiers compatible with **/REMOTE**.

4.8.4 DCL Commands That Control Print Jobs

The **DCL** commands listed in the following table allow you to control print jobs in various ways. For example, you can specify the number of copies printed or you can request that the system notify you when your print job is complete. For more information on any of these commands, see the descriptions of the **DCL** commands in the Reference Section.

Print Operations	Print Job Commands and Qualifiers
Number of copies	
By job	PRINT/JOB_COUNT=n ¹
By file	PRINT/COPIES=n ¹
Specified file only	file-spec/COPIES=n ¹
Number of pages	PRINT/PAGES=1
Print features	
Flag pages	PRINT/FLAG=1
Type of forms (paper)	PRINT/FORM=1
Special features	PRINT/CHARACTERISTICS=1
Double-spacing	PRINT/SPACE ¹
Page heading	PRINT/HEADER ¹
Notification of job execution	PRINT/NOTIFY
Delay execution of a job	
For a specified time	PRINT/AFTER
Indefinitely	PRINT/HOLD
Release a delayed job	SET QUEUE/ENTRY/RELEASE
Display your print jobs	SHOW ENTRY
Stop a print job	
Delete job	DELETE/ENTRY=job-number
Stop currently printing job and begin printing the next job in the queue	STOP/ABORT
Stop currently printing job and requeue it for printing	STOP/REQUEUE

¹Parallel qualifiers for the SET QUEUE/ENTRY command allow you to specify these operations for print jobs that are already queued but not yet printing.

Chapter 5

Directories: Organizing and Managing Files

Directories are files that store the names of files. Well-organized directories help you manage files efficiently.

This chapter describes how files are stored in directories and describes the following directory tasks you can perform to organize and manage your files:

- Creating directories
- Displaying directories
- Setting a default directory
- Deleting directories
- Protecting a directory from other users
- Searching the directory structure with wildcards

The descriptions of the DCL commands in the Reference Section describe specific directory tasks you can perform locally and over the network.

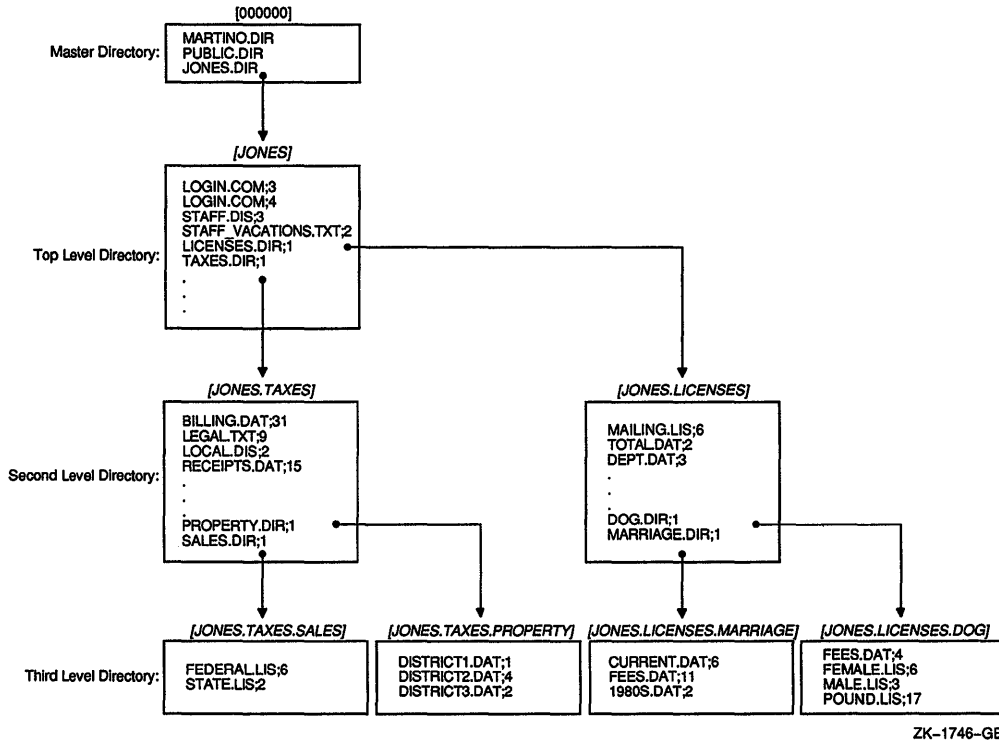
NOTE: In the examples of remote operations in this chapter, proxy accounts enable users to perform operations on remote systems. Proxy accounts are one way users can access remote systems. For more information about additional ways to access remote systems, see the *VMS System Manager's Manual*.

5.1 Understanding Directory Structures

Figure 5–1 shows a sample directory hierarchy. At the top of the structure is the master file directory (MFD). Its directory name is [000000]. Figure 5–1 contains entries for user file directories including MARTINO.DIR, PUBLIC.DIR, SCHULTZ.DIR, and JONES.DIR. The top level directory [JONES] is a user file directory named JONES.DIR;1 in [000000].

5-2 Directories: Organizing and Managing Files

Figure 5-1: Directory Structure



ZK-1746-GE

Assume that you are user JONES. When you log in, the system places you in [JONES], your default directory. [JONES] contains the following four nondirectory files:

- LOGIN.COM;3
- LOGIN.COM;4
- STAFF.DIS;3
- STAFF_VACATIONS.TXT

[JONES] also contains the following two directory files:

- TAXES.DIR
- LICENSES.DIR

The directory file TAXES.DIR;1 points to the [JONES.TAXES] subdirectory; LICENSES.DIR;1 points to the [JONES.LICENSES] subdirectory. (Subdirectories are specified by concatenating the subdirectory name to the name of the directory one level above it.)

The [JONES.LICENSES] subdirectory contains three nondirectory files and two directory files. The directory file DOG.DIR;1 points to the [JONES.LICENSES.DOG] subdirectory; MARRIAGE.DIR points to the [JONES.LICENSES.MARRIAGE] subdirectory.

This sample directory structure is the basis for the examples in this chapter.

5.2 Understanding Directory Names and Specifications

Use a named directory specification to refer to a directory. A named directory specification consists of a top level directory name that can be followed by a maximum of seven subdirectory names.

A named directory specification has the following format:

```
[directory.subdirectory[.subdirectory...]]
```

A directory name can contain up to 39 alphanumeric characters. Any characters valid for file names are also valid for directory names. Enclose the directory name in either square brackets ([]) or angle brackets (<>).

5.3 Creating Directories

To create a directory, enter the CREATE/DIRECTORY command. For example, to create a directory [JONES.LICENSES], enter the following command:

```
$ CREATE/DIRECTORY [JONES.LICENSES]
```

If you want to create a subdirectory under your current directory, you do not have to specify the current directory name; you can enter the subdirectory name preceded by a period. For example, if your current default directory is [JONES], enter the following command:

```
$ CREATE/DIRECTORY [JONES.LICENSES]
```

5.4 Displaying Directories

To display the names of files in a directory, enter DIRECTORY at the DCL prompt. For example, to list the files in [JONES], enter the following command:

```
$ DIRECTORY
```

5-4 Directories: Organizing and Managing Files

The system displays the contents of [JONES] as follows:

```
Directory DISK1:[JONES]
LICENSES.DIR;1
LOGIN.COM;3
LOGIN.COM;4
STAFF.DIS;3
STAFF_VACATIONS.TXT;2
TAXES.DIR;1
Total of 5 files.
```

This example shows that [JONES] contains two subdirectories—[JONES.LICENSES] and [JONES.TAXES]—and four nondirectory files—STAFF.DIS, STAFF_VACATIONS.TXT, and two versions of LOGIN.COM.

To list the files in a subdirectory, enter the DIRECTORY command and the subdirectory name preceded by a period. For example, assuming that the default directory remains [JONES], to list the contents of the subdirectory [JONES.LICENSES], enter the following command:

```
§ DIRECTORY/[.LICENSES]
```

The system displays the contents of [.LICENSES] as follows:

```
Directory DISK1:[JONES.LICENSES]
MAILING.LIS;6
TOTAL.DAT;2
DEPT.DAT;3
DOG.DIR;1
MARRIAGE.DIR;1
Total of 6 files.
```

TIP: If you want to move one level down the directory structure, you need to specify only the next subdirectory name preceded by a period, as shown in the previous example.

5.5 Setting a Default Directory

To create a file in a subdirectory, you must be located at that directory, making it your new default directory. To change your default directory, use the SET DEFAULT command. The default remains in effect until you enter another SET DEFAULT command.

For example, to set default to the directory [JONES] and then display the file STAFF_VACATIONS.TXT, enter the following commands:

```
§ SET DEFAULT [JONES]
§ TYPE STAFF_VACATIONS.TXT
```


To specify a subdirectory, combine the subdirectory name to the name of the directory one level above it. For example, to display the file BILLING.DAT located in the subdirectory [JONES.TAXES], enter the following commands:

```
$ SET DEFAULT [JONES.TAXES]
$ TYPE BILLING.DAT
```

To display your current default directory, enter the command SHOW DEFAULT, as shown in the following example:

```
$ SHOW DEFAULT
DISK1:[JONES.TAXES]
$ SET DEFAULT [PUBLIC]
$ SHOW DEFAULT
DISK1:[PUBLIC]
```

5.6 Deleting Directories

To delete a directory, use the following procedure:

1. Make sure that the directory contains no files. To find out if the directory contains files, enter the DIRECTORY command, as shown in the following example:

```
$ DIRECTORY
```

If there are no files in the directory, the system displays the following message:

```
No files found.
```

If the directory contains files, copy them to another directory to save them; delete them if you do not want to save them. If the directory contains subdirectories, examine those subdirectories, copy or delete their files, and delete the subdirectories.

2. Move to the directory one level above the directory you want to delete. For example, if you want to delete [JONES.LICENSES], you should set default to [JONES]. Remember that the subdirectory [JONES.LICENSES] exists as a file named LICENSES.DIR;1 in the directory [JONES]. When you delete a directory, you delete the file that points to that directory.
3. Change the file protection of a directory to allow delete access to the file. (See Chapter 4 for more information about file protection.) For example, to change the file protection of LICENSES.DIR, enter the following command:

```
$ SET PROTECTION=OWNER:D LICENSES.DIR
```

4. Delete the directory file. For example, to delete the directory file LICENSES, enter the following command:

```
$ DELETE LICENSES.DIR;*
```

5-6 Directories: Organizing and Managing Files

The following example shows how to delete the subdirectory [JONES.LICENSES]:

```
$ SET DEF [JONES.LICENSES]
$ DIR
NO FILES FOUND
$ SET DEFAULT [JONES]
$ SET PROTECTION=OWNER:D LICENSES.DIR
$ DELETE LICENSES.DIR;1
```

The directory files (for example, JONES.DIR;1) in the master file directory require SYSPRV privilege to delete. See the *VMS System Manager's Manual* for a discussion of user privileges.

5.7 Protecting a Directory from Other Users

You cannot completely protect a file without applying at least the same protection to the directory in which the file resides. For example, if you deny a user all access to a file but allow that user read access to the file's directory, the user cannot access the contents of the file but can see that it exists. Conversely, a user allowed access to a file and denied access to the file's directory (or one of the parent directories) cannot see that the file exists.

NOTE: To protect sensitive files, directory protection alone is not adequate. You must also protect each file within the directory.

By default, top level directories receive UIC-based protection (S:RWE,O:RWE,G:RE,W:E) and no ACL. Subdirectories receive UIC-based protection from the parent directory.

To specify UIC-based protection explicitly when creating a directory, use the /PROTECTION qualifier of the CREATE/DIRECTORY command. You cannot specify an ACL for the directory until the directory is created. To change the UIC-based protection of an existing directory, use the SET PROTECTION command (apply this command to the directory file).

You can limit but not prohibit directory access by specifying execute access but not read access. Execute access on a directory permits you to examine and read files that you know are contained in the directory (that is, you know the file specifications), but prevents you from displaying a list of the files in the directory.

5.8 Using Wildcards to Search the Directory Structure

From any point in a directory structure, you can refer to another directory or subdirectory in the structure. Do this by specifically naming the directory or subdirectory you want or by using the ellipsis (. . .) and hyphen (-) wildcard characters.

5.8.1 The Ellipsis (. . .) Wildcard Character

Use the ellipsis wildcard character to search down into the directory hierarchy. To search the current directory and all the subdirectories below it, use the ellipsis by itself as shown in the following command:

```
$ DIRECTORY [...]
```

For example, assuming the current directory is [JONES], to display the latest versions of all files named FEES.DAT in [JONES] and all subdirectories under [JONES], enter the following command:

```
$ TYPE [JONES...]FEES.DAT
```

If you begin the directory specification with an ellipsis, the search begins from your current directory. For example, assuming the current default directory is [JONES], to search all subdirectories that end in .SALES and display the latest versions of the file FEDERAL.LIS, enter the following command:

```
$ TYPE [...SALES]FEDERAL.LIS
```

The following command displays the latest versions of all files named DEPT.DAT in [JONES] and all subdirectories under [JONES]:

```
$ TYPE [...]DEPT.DAT
```

However, if you begin the directory specification with a period, only the subdirectory that is one level lower than the current directory is searched. Assuming the current directory is [JONES], the following command searches only the [.LETTERS] subdirectory that is one level lower than [JONES] for the file INVITATION.TXT. The subdirectory [JONES.LETTERS] is searched, but [JONES.WORK.LETTERS] is not:

```
$ TYPE [.LETTERS]INVITATION.TXT
```

Assuming the current directory is [JONES], the following command displays the latest versions of all files named DEPT.DAT in the [.LICENSES] subdirectory under [JONES] and all subdirectories under the [.LICENSES] subdirectory:

```
$ TYPE [...LICENSES...]DEPT.DAT
```

To search all top level directories and their subdirectories from wherever you are in the directory structure, use an asterisk (*) followed by an ellipsis (. . .). The following command (which requires READALL privilege) searches as many as eight levels of directory names (the top level directory and seven subdirectories), if they exist. It does not search the MFD.

```
$ DIRECTORY [*...]
```

5.8.2 The Hyphen (–) Wildcard Character

Use the hyphen wildcard character to move up through the directory structure. Each hyphen refers to the directory one level up from the current one. You can follow the hyphens with directory and subdirectory names to move down the directory structure on another path.

For example, if your current directory is [JONES.LICENSES], enter the following command to display the latest version of STAFF.DIS in [JONES]:

```
$ TYPE [-]STAFF.DIS
```

If your current directory is [JONES.LICENSES], enter the following command to display the latest version of BILLING.DAT in [JONES.TAXES]:

```
$ TYPE [-.TAXES]BILLING.DAT
```

You can specify more than one hyphen. The following command moves you up two levels in the directory hierarchy. From there, you are placed in the top level directory [JONES].

```
$ SET DEFAULT [--JONES]
```

If you enter so many hyphens that you point above the master file directory (MFD), the system displays an error message.

Chapter 6

Editing Text Files: Using EVE

EVE is a general-purpose text editor that is included with the VMS operating system. You can use EVE to create and edit new files or to edit existing files. EVE is interactive, so you see the changes to a file as you make them. You can use EVE on VT300-, VT200-, or VT100-series terminals but not on hardcopy terminals.

EVE provides extensive online help. For more information about online help, see Section 6.2. EVE also provides two optional keypads: an EDT keypad or a WPS keypad. The EDT and WPS keypads provide most (but not all) of the EDT and WPS key functions. For more information about the EDT and WPS keypads, see Section 6.6.1 and Section 6.6.2.

6.1 Beginning an Editing Session

You can start an EVE editing session either by creating and editing a new file or by editing an existing file. To begin an editing session, enter the DCL command EDIT/TPU followed by the name of the new file you want to create or the existing file you want to edit. For example, to invoke EVE and create a new file named NEWFILE.DAT, enter the following command:

```
$ EDIT/TPU NEWFILE.DAT
```

(If you wanted to edit an existing file, you would use the same format, substituting the name of the existing file for *NEWFILE.DAT*.)

This command produces a screen that appears as follows:

```
[End of file]
```

```
Buffer: NEWFILE.DAT | Write | Insert | Forward
```

```
Editing new file: could not find WORKDISK:[USER]NEWFILE.DAT
```

6-2 Editing Text Files: Using EVE

If you specify a file on the command line, EVE inserts the text of the file you are editing into a temporary holding area called a **buffer**. The contents of the buffer are shown in an area of your screen called a **window**. EVE buffers exist only during the editing session.

The end-of-file marker marks the end of an EVE buffer. It is visible only on the screen and does not become part of your file. When you add text to the buffer, the end-of-file marker moves down. Depending on the length of your terminal screen, the marker may not be visible when you view the beginning of a buffer that contains many lines of text.

A highlighted **status line** appears at the bottom of the EVE window and provides information about the buffer you are viewing in the window. The status line shows the buffer name, editing status (write or read-only), current mode (insert or overstrike), and current direction (forward or reverse).

When you invoke EVE to edit a file, an informational message appears in the **message window** beneath the highlighted status line. The message states either that the file is a new file or that a certain number of lines were read from an existing file. During the editing session, EVE displays other messages in the message window.

6.2 Using Online Help

EVE has online help that supplies information on editing commands and keys without disturbing your work. You can get help by entering the HELP command or by pressing the Help key.

Use the Previous Screen and Next Screen keys (PERIOD and KP0 on the keypad of a VT100-series terminal) to scroll through the list of EVE topics. To get information on a particular command, type a command name after the help prompt and press Return. The help text appears on the screen.

If you know the name of a specific command for which you want help, press the Do key, type HELP followed by the name of the command, and press Return. The help text for that command appears on the screen. For example, to get help on the MOVE BY LINE command, enter the command HELP MOVE BY LINE.

6.3 Ending an Editing Session

When you are finished with the editing session, you can either save your edits or discard them.

6.3.1 Saving Your Edits

To end the editing session and save your edited text, use the EXIT command. You can enter the EXIT command by pressing the F10 key (on VT200-series or VT300-series terminals) or by pressing CTRL/Z. When you use the EXIT command, EVE produces a new version of the edited file.

If you have modified the current buffer, EVE creates a new version of the file with the same file name and file type as the original version, with the version number incremented by one. For example, if you use the EXIT command after modifying a file named FUN.DAT;1, the output file is named FUN.DAT;2:

```
Command: EXIT
4 lines written to file WORKDISK:[USER]FUN.DAT;2
```

6.3.2 Ending the Session Without Saving Your Edits

To end the editing session *without* saving the edits that you made, use the QUIT command. With the QUIT command, the editing session ends and any edits that you made are ignored. Any existing versions of the files remain unchanged regardless of how the editing session is ended. To execute the QUIT command, do the following:

1. Press the Do key (PF4 on VT100-series terminals).
2. Type QUIT at the *Command:* prompt.
3. Press Return.

For example, if you have made edits to a file named FUN.DAT and enter the QUIT command, the system displays the following:

```
Command: QUIT
Buffer modifications will not be saved, continue quitting (Y or N)?
```

Type Y and press Return if you want to quit without saving the edits. If you change your mind and decide to save your edits, type N, press Return, and exit from the file using the EXIT command.

6.4 Entering EVE Commands

After you invoke EVE, you can enter EVE commands to edit text, move the cursor, and perform other operations. You can enter EVE commands in either of two ways:

- By pressing predefined keys
- By typing the commands themselves

The following sections describe how to enter commands. Table 6-3 at the end of this chapter lists many EVE commands that you can use and the keys that are predefined to execute them.

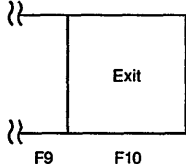
6.4.1 Using Defined Keys to Enter EVE Commands

EVE defines some keys by default. The predefined keys on VT300-series and VT200-series terminals include the minikeypad (located between the main keyboard keys and the numeric keypad), certain function keys, and certain control key sequences. Figure 6-1 shows the predefined keys for the VT300-series and VT200-series terminals. On VT100-series terminals, EVE automatically defines most of the numeric keypad keys, the four arrow keys, and certain control keys. Figure 6-2 shows the predefined keys for the VT100-series terminal.

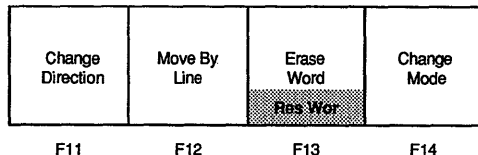
Control keys, arrow keys, the Tab, Return, and Delete keys have the same definitions on all three types of terminal. For example, the Tab key on a VT100-series terminal does the same thing as the Tab key on a VT300- or VT200-series terminal.

Throughout this chapter, EVE editing keys are referred to by their names, rather than by their locations on the keyboard. For example, on a VT300-series or VT200-series terminal, two keys are defined as the Do key: the key located at the top of the editing keypad, labeled Do, and the PF4 key located on the upper right of the numeric keypad. On a VT100-series terminal, the Do key is the PF4 key located at the upper right of the numeric keypad.

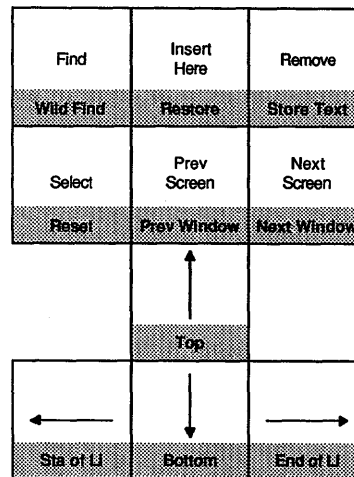
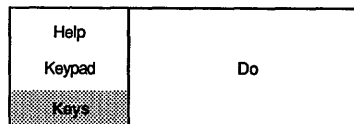
Result of SET KEYPAD NOEDT or SET KEYPAD NOWPS Commands



- ⌘ DELETE
- Tab TAB
- Return RETURN
- Enter RETURN
- PF4 DO

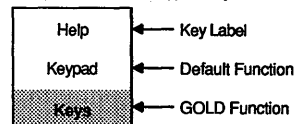


- CTRL/A CHANGE MODE
- B RECALL
- E END OF LINE
- H START OF LINE
- I TAB
- J ERASE WORD
- L INSERT PAGE BREAK
- M RETURN
- R REMEMBER
- U ERASE START OF LINE
- V QUOTE
- W REFRESH
- Z EXIT



GOLD key functions are shown in gray shading.

Sample Function or Keypad Key

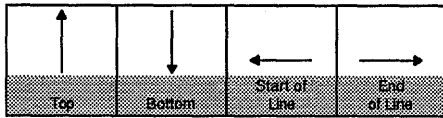


ZK-1055A-GE

Figure 6-1: Editing Keys—VT200-Series and VT300-Series Terminals

Figure 6-2: Editing Keys—VT100-Series Terminals

Default on VT100 Terminal
Available on VT200 Terminal With SET KEYPAD VT100 Command



Delete DELETE
Tab TAB
Return RETURN
Backspace START OF LINE
Linfeed ERASE WORD

CTRL/A CHANGE MODE
B RECALL
E END OF LINE
H START OF LINE
I TAB
J ERASE WORD
L INSERT PAGE BREAK
M RETURN
R REMEMBER
U ERASE START OF LINE
V QUOTE
W REFRESH
Z EXIT

Find	Help Keypad	Change Direction	Do
Select	Remove	Insert Here	Move By Line
	↑		Erase Word
←	↓	→	Change Mode
Next Screen	Prev Screen		

GOLD key functions are shown in shaded boxes.

ZK-6301-GE

You can also use DCL line-editing keys in an EVE session. For example, use CTRL/U to erase to the beginning of the line, CTRL/E to move to the end of the line, and CTRL/B to recall the last command entered. By default, the editing mode (insert or overstrike) of the EVE command line is the same as the editing mode of your terminal. (You can change the default with the DCL command SET TERMINAL prior to invoking EVE. Once in EVE, you can change the editing mode by pressing CTRL/A.)

In addition to providing the default predefined keys, EVE lets you do any of the following:

- Use an EDT-like keypad
- Use a WPS-like keypad
- Define your own keys
- Redefine any of the keys predefined by default

See Section 6.6 for more information about these features.

6.4.2 Typing EVE Commands

In addition to using defined keys to enter commands, you can type commands at the *Command:* prompt. To type EVE commands, do the following:

1. Press the Do key. EVE displays the *Command:* prompt in the command window beneath the highlighted status line.
2. Type the EVE command after the prompt. The following example shows how to enter the EXIT command:

```
Command: EXIT
```

3. Press Return or the Do key to enter the command.

To save keystrokes when you are typing EVE commands, do the following:

- To recall the last EVE command you entered, press CTRL/B. Pressing CTRL/B again recalls the next to the last command and so forth. Continue pressing CTRL/B until the command you want appears on your screen, and press Return to enter it.
- To abbreviate EVE command names, use the first letters of each command term, make sure to use enough letters to uniquely identify the command. For example, if you wanted to give the OVERSTRIKE command, you could enter *OVER* at the *Command:* prompt.
- To repeat an EVE command or keystroke a specified number of times, use the REPEAT command. Enter the REPEAT command and the number of times it is to be repeated. EVE repeats the next character or command you enter the specified number of times. For example, the following commands erase five words (the current word and the four previous):


```
Command: REPEAT 5
Will repeat next command 5 times.
Command: ERASE WORD
```
- To repeat the last command entered, press the Do key twice.

6.4.3 EVE Key Names

You can type the name of a key as a parameter for the DEFINE KEY, SET GOLD KEY, SHOW KEY, and UNDEFINE KEY commands. Generally, EVE key names are the same as DCL key names. For example, the 7 on the numeric keypad is named KP7. Key names cannot be abbreviated; they are not case sensitive. In specifying control keys or GOLD key combinations, use a slash, dash, or underscore in the key name—for example, CTRL/N or GOLD-F20. Thus, in an initialization file, you can use commands with typed key names such as the following:

```
DEFINE KEY= CTRL/P MOVE BY PAGE
DEFINE KEY= GOLD-N NEXT BUFFER
DEFINE KEY= KP7 CENTER LINE
SET GOLD KEY F17
```

Table 6-1 lists EVE key names and how the keys are labeled on the keyboard or keypads. Note that some keys may not appear on some terminals. (For example, VT100-series terminals do not have the F1 through F20 keys. VT200- and VT300-series terminals do not have BACKSPACE and LINEFEED keys.)

Table 6-1: EVE Key Names

Key Name	Label
F7 ... F20	F7 ... F20
HELP or F15	Help
DO or F16	Do
E1	Find
E2	Insert Here
E3	Remove
E4	Select
E5	Prev Screen
E6	Next Screen
UP	↑
LEFT	←
DOWN	↓
RIGHT	→
PF1 ... PF4	PF1 ... PF4
KP0 ... KP9	0 ... 9 (numeric keypad)
MINUS	- (numeric keypad)
PERIOD	. (numeric keypad)
COMMA	, (numeric keypad)
DELETE	⌫ or DELETE
TAB or CTRL/I	Tab or TAB
BS or CTRL/H	BACKSPACE (VT100-series terminals)
LF or CTRL/J	LINEFEED (VT100-series terminals)

6.5 Editing Text

Once you know how to invoke the EVE editor and how to enter commands, you can use EVE commands to create and edit files. Editing keys and commands let you move the cursor and perform editing operations such as moving, erasing, and restoring text.

Before you begin typing text, look at the highlighted status line to check whether the buffer is in insert or overstrike mode. If the buffer is in insert mode, text is inserted at the cursor position, and text that already appears in the file moves to accommodate your insertions. If the buffer is in overstrike mode, text that you type at the keyboard is inserted at the cursor position, and the text that already appears in the file is overwritten as the cursor moves through it. Press CTRL/A to change from one mode to the other.

You can add text to your buffer in the following ways:

- **Text**—You can type characters at the keyboard. EVE adds the characters to the buffer at the current cursor position according to the current mode of the buffer (insert or overstrike). In insert mode, the new characters move existing characters to the right and down. In overstrike mode, the new characters replace existing characters.
- **Files**—You can add an entire file by pressing the Do key and entering the EVE command `INCLUDE FILE`. Type the file specification at the *File to include:* prompt and press Return. Regardless of the current mode (insert or overstrike) of the buffer, EVE inserts the entire contents of the specified file into the buffer just before the line where the cursor currently appears.
- **Inserting or restoring text**—You can include text that you erased (deleted) or removed (cut). To include text that you erased, use the appropriate `RESTORE` command; to include text that you removed, use the `INSERT HERE` command.

Table 6-3, located at the end of this chapter, lists many EVE commands that you can use to move the cursor and manipulate text. This table also indicates when the command has corresponding predefined keys. Note that the keys correspond to the commands only when the default EVE keypad is used; other key definitions might apply if you use alternate keypads (EVE or WPS) or if the keys have been redefined.

6.5.1 Locating Text

To locate specific text in the current buffer, enter the `FIND` command. Then type the text you want to locate, which is called the **search string**. For example, enter the following commands to find the search string *rhymes with* in the forward direction.

```
Command: FIND
Forward Find: rhymes with
```

If the string is found, EVE moves the cursor to the beginning of the specified string.

If the search string contains all lowercase letters, EVE disregards the case of letters and locates any occurrence of the string. Thus, the search string *the* matches *the*, *THE*, and *thE*. If the search string contains one or more uppercase letters, EVE finds only the occurrences of the string in which the case of each

6-10 Editing Text Files: Using EVE

letter is exactly the same. Therefore, the only match for the search string *tHis* is *tHis*.

EVE is sensitive to accent marks and locates only those occurrences of the string in which the accent marks are exactly the same. For example, in searching for *ë*, EVE does not locate occurrences of *e*, *é*, *è*, or *ê*.

The current direction of the buffer determines whether EVE first searches in a forward or reverse direction.

If the editor cannot find the string in the current direction but finds it in the opposite direction, EVE prompts you to change direction. The following example shows the system response when the string *rhymes with* is found in the opposite direction from the search:

```
Forward Find: rhymes with  
Found in reverse direction. Go there?
```

To search in the opposite direction, type *Y*. EVE moves the cursor to the first occurrence of the string in the opposite direction. The current direction in the highlighted status line does not change, however.

When EVE finds the search string, the editor highlights it and moves the cursor to the first letter of the string. You can use any one of the following commands on a highlighted search string:

- CAPITALIZE WORD
- COPY
- CUT
- FILL
- FILL RANGE
- FIND NEXT
- FIND SELECTED
- LOWERCASE WORD
- OPEN SELECTED
- REMOVE
- STORE TEXT
- UPPERCASE WORD

Some EDT or WPS keypad keys

To cancel the highlighting, move the cursor off the search string or use the **RESET** command.

To find the next occurrence of the search string, press the **Find** key twice or enter the **FIND NEXT** command.

6.5.2 Replacing Text

The REPLACE command lets you replace a text string in the current buffer with another text string. This is useful if you have spelled a word incorrectly throughout a long file and you want to fix every occurrence of the misspelled word.

For example, to use the REPLACE command to replace every occurrence of the string *ee* with the string *oo*, use the following procedure:

1. Move the cursor to the top of the buffer.
2. Press the Do key, type REPLACE, and press Return.
3. Type *ee* at the highlighted *Old string:* prompt and press Return.
4. Type *oo* at the highlighted *New string:* prompt and press Return.
5. Type *all* and press Return. All occurrences of the string *ee* are replaced with the string *oo*.

If the old string is found, EVE highlights the text and asks you to choose one of the following; you need only type the first letter of the response and press Return:

Response	Effect
Yes	Replace this occurrence and find the next one. (Default. You can simply press Return.)
No	Skip this occurrence and find the next one.
All	Replace all the occurrences (no further prompting unless EVE finds an occurrence in the opposite direction).
Last	Replace this occurrence and stop here.
Quit	Skip this occurrence and stop here.

The REPLACE command is case sensitive. If the old string has any uppercase letters, EVE searches for exact case matches. If the old string is all lowercase, EVE searches for any occurrence of the string regardless of its case. If the new string has any uppercase letters, EVE replaces the string exactly. If the old and new strings are all lowercase, EVE replaces the string according to the following rules:

- A capitalized version of the old string (first letter uppercase, others lowercase) is replaced by a capitalized version of the new string.
- An all-uppercase version of the old string is replaced by an all-uppercase version of the new string.
- Otherwise, the old string is replaced by an all-lowercase version of the new string.

6-12 Editing Text Files: Using EVE

The following table shows how EVE uses the case of the strings:

Old String	New String	Highlights	Replacements
butter	margarine	butter	margarine
		Butter	Margarine
		BUTTER	MARGARINE
		BUtteR	margarine
Butter	margarine	Butter	margarine
butter	Margarine	butter	Margarine
		Butter	Margarine
		BUTTER	Margarine
		BUtteR	Margarine
Butter	Margarine	Butter	Margarine

6.5.3 Recovering from System Interruptions

EVE has recovery procedures for two types of system interruptions. You can remove extraneous characters that appear on your screen, and you can recover a “lost” editing session with the journaling facility.

6.5.4 Refreshing the Screen

If extraneous characters, such as an operator message, appear on your terminal screen while you are editing, press `CTRLW` to refresh the screen. The screen becomes blank, and then all characters are redrawn, minus any extraneous characters.

6.5.5 Using the Journal File

If you are editing a file and a system interruption occurs (that is, a break in communication between your terminal and the computer), you can recover your “lost” editing session. By default, EVE records every keystroke you enter during an editing session in a journal file that has the same file name as the file you are editing and a file type of `TJL`.

Typically, an editing session ends without interruption, so the system deletes the journal file. When the system fails, however, the journal file is saved. EVE can use the journal file to reconstruct your editing session so that only the last few keystrokes of your editing session are lost and sometimes nothing is lost.

To recover an editing session, enter the DCL command you used to invoke EVE plus the /RECOVER qualifier. For example, to recover an editing session you began with the command EDIT/TPU LETTER.RNO, type the following command and file name and press Return:

```
$ EDIT/TPU/RECOVER letter.rno
```

You must recover an editing session at a terminal of the same type as the one you used for your editing session. When EVE finishes recovering the session, check to be sure that the last few keystrokes of your editing session were recovered and continue editing the file. If another system interruption occurs before you exit, a journal file containing the keystrokes from both editing sessions is saved.

The journal file is saved in the current default directory. However, you can create a journal file in another directory by using the /JOURNAL qualifier, as in the following DCL command:

```
$ EDIT/TPU/JOURNAL=[alexis.travels]letter.tj1 letter.rno
```

If you use the /JOURNAL qualifier to create a journal file with a different file name or a different directory, then you must use the /JOURNAL qualifier and the file name when you recover the file. For example, to recover the file LETTER.RNO when the journal file is in directory [ALEXIS.TRAVELS], enter the following command:

```
$ EDIT/TPU/JOURNAL=[alexis.travels]letter.tj1/RECOVER letter.rno
```

The journaling facility has the following two restrictions:

- All relevant files and terminal settings must be the same as they were before the system interruption or the recovery might fail or might not work as expected. For example, if you used the WRITE FILE command during your editing session to copy the contents of the buffer to another file then, in recovering your edits, you must specify the original version number. In this example, you are editing an existing file called LETTER.RNO;1 and use the WRITE FILE command; EVE creates LETTER.RNO;2. The system then fails and you must enter the original version of LETTER.RNO on the EDIT/TPU/RECOVER command line. In this example you would type LETTER.RNO;1. (See Section 6.5.8 for more information on the WRITE FILE command.)
- If you press CTRL/C during an editing session, immediately exit from the editing session and invoke EVE again. EVE does not include CTRL/C in the journal file so the recovery will not work as expected.

6.5.6 Listing Buffers

To display a list of all the buffers you have created during an editing session, enter the `SHOW BUFFERS` command. To display a list of all buffers that EVE has created, enter the `SHOW SYSTEM BUFFERS` command. You can scroll through the list and specify the buffer you want to view by moving the cursor to the buffer name and pressing the `Select` key. EVE puts the buffer in your current window.

NOTE: Do not delete system buffers, such as `Insert Here`, `Messages`, `$RESTORE$`, or `$DEFAULT$`, because these buffers are necessary for some commands to work properly.

6.5.7 Editing Two Buffers

During an editing session you can use several buffers if you want to edit more than one file or if you want temporary storage areas for manipulating blocks of text.

You can create a new buffer using one of the following commands: `GET FILE` or `OPEN`, `OPEN SELECTED`, or `BUFFER`. If the buffer you specify does not already exist, EVE creates a new buffer. You can use the asterisk wildcard character (`*`) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (`%`) as a substitute for one character in the file name and file type, and you can use the ellipsis wildcard (`[. . .]`) as a substitute for a directory specification.

If the specified file exists, EVE reads the contents of the file into a new buffer and displays the buffer in the current window. If there is more than one match for a file specification with a wildcard, EVE displays a list of choices and prompts you to provide a more complete file specification. Otherwise, EVE creates an empty buffer and displays the buffer in the current window.

To change the buffer in the current window, press the `Do` key, type `BUFFER` and the name of the buffer you want to display on the screen, and press `Return`. If you forget a buffer name, enter the `SHOW BUFFERS` command to display the names of active buffers in your editing session and specify a buffer with the `Select` key.

6.5.8 Reading and Writing Files

There are four ways to read a file into an EVE buffer.

- Invoke EVE with a file specification.
- Enter the `INCLUDE FILE` command and the name of the file you want to include. EVE reads the entire contents of the file into the buffer just before the line where the cursor is located. Using the `INCLUDE FILE` command does not change the name of the buffer on the status line.

- Enter the GET FILE or OPEN command and the name of the file you want to use. Either command creates a new buffer and reads the contents of an existing file into the buffer. The name of the buffer on the status line is the same as the file name you specify with the GET FILE or OPEN command. (See Section 6.5.7.)
- Select or find a file name, then enter the OPEN SELECTED command.

To write the contents of the current buffer to a file, enter the WRITE FILE command. You can include a file specification with the WRITE FILE command. If you do not include a file specification, EVE writes the file using the input file specification. If you created the current buffer with the BUFFER or NEW command, EVE prompts you for a file specification to which it writes the file.

6.5.9 EVE Default Settings

Table 6-2 lists the EVE default settings—the settings EVE uses unless you specify otherwise. You may want to refer to this table in creating an initialization file, to check which settings you want to change. Note that some settings are global (applying in all buffers you edit), and others are buffer-specific. For example, the type of cursor motion (bound or free) and tab mode (insert, spaces, or movement) are the same in all buffers you edit, whereas margins, paragraph indent, and tab stops can be set differently for each buffer. (You may want one buffer to have a right margin of 75 and another to have a right margin of 68.)

Table 6-2: EVE Default Settings

Default Setting	Effects
Global Settings (Applying to All Buffers)	
SET NOCLIPBOARD	Copy, cut, and paste operations use the EVE Insert Here buffer. On DECwindows, you can enable the clipboard, which lets you transfer text between EVE and other DECwindows applications. WPS keypad keys do not use the clipboard, regardless of the setting.
SET CURSOR FREE	You can move the cursor anywhere in the buffer and enter text there, as opposed to a bound cursor, which cannot move into the unused portion of the buffer. Note that using SET KEYPAD WPS automatically enables a bound cursor.
SET FIND NOWHITESPACE	FIND and WILDCARD FIND commands match spaces and tabs in the search string exactly as entered, and do not search across a line break.

(continued on next page)

Table 6-2 (Cont.): EVE Default Settings

Default Setting	Effects
Global Settings (Applying to All Buffers)	
SET NOGOLD KEY	EVE does not have a default GOLD key. Setting the EDT or WPS keypad makes PF1 the GOLD key, overriding any current definition of PF1, unless you set a different key as GOLD.
SET KEYPAD NUMERIC or SET KEYPAD VT100	On VT300-series and VT200-series terminals, keys on the numeric keypad are undefined, except for the PF4 and ENTER keys. On VT100-series terminals, the numeric keypad is used for the EVE default key bindings. Control keys are defined the same on either type of terminal. Also, you can set the EDT keypad or WPS keypad on either type of terminal.
SET NOPENDING DELETE	Using DELETE or typing new text does <i>not</i> erase a select range.
SET SCROLL MARGINS 0 0	Scrolling begins automatically when you move past the top or bottom of the window.
SET TABS INSERT	Using TAB inserts a tab character. You can set the tab mode to insert spaces instead of a tab character, or to move the cursor without inserting anything.
SET TABS INVISIBLE	Tab characters appear during editing as blank space, as opposed to visible tabs, which appear as a small \overline{H}_T (horizontal tab).
SET WIDTH 80	The width of the EVE screen layout is the same as your terminal setting—typically 80 columns.
SET WILDCARDS VMS	The WILDCARD FIND command uses VMS-style wildcards, such as the asterisk (*) to match any amount of text on a line, the percent sign (%) to match a single character on a line, and so on. You can enable ULTRIX-style wildcards.
Buffer-Specific Settings	
FORWARD	Commands like FIND and MOVE BY LINE move the cursor to the right and down. You can change the direction to reverse (left and up).
INSERT MODE	Characters you type are inserted at the current position, pushing existing text to the right and down. You can change the mode to overstrike.
SET BUFFER MODIFIABLE	Buffers you create can be modified (edited). You can set the buffer to unmodifiable.

(continued on next page)

Table 6-2 (Cont.): EVE Default Settings

Default Setting	Effects
Buffer-Specific Settings	
SET BUFFER WRITE	On exiting, EVE writes out (saves) your buffers if you have made any changes. You can set the buffer to read-only.
SET LEFT MARGIN 1	This is the leftmost column. When you press Return or use FILL commands or when EVE wraps text, new lines start at the left margin of the buffer.
SET PARAGRAPH INDENT 0	Paragraphs you create or ones you reformat with FILL commands start at the current left margin of the buffer—with no indent.
SET RIGHT MARGIN 79	The default right margin is one column less than the width set for your terminal. If the width is 80 columns, the default right margin is 79. When you use FILL commands or when you type at the end of a line, EVE wraps text at the right margin of the buffer.
SET TABS EVERY 8	Tab stops are set at columns 9, 17, 25, 33, 41, and so on. You can set tab stops at different intervals.
SET WRAP	As you type text at the end of a line, EVE wraps text at the right margin of the buffer, without your having to press the Return key or use FILL commands.

Note: For editing EVE command lines—such as when you recall a command—the default direction is reverse, the default mode matches your terminal setting, and the cursor is bound.

EVE settings such as margins, tabs, and type of cursor motion are not usually saved when you create a section file. Instead, to save these editing preferences, you can use an initialization file, which contains EVE commands—effectively, setting your own private defaults. You can also put key definitions in an initialization file instead of (or in addition to) saving them in a section file. For example, the following EVE initialization file sets the EDT keypad, defines some keys, sets bound cursor motion, sets the right margin to 70, and sets the tab mode to insert spaces:

```
! MYINIT.EVE initialization file
!
SET KEYPAD EDT
DEFINE KEY= gold-c center line
DEFINE KEY= f20 show buffers
!
SET CURSOR BOUND
SET RIGHT MARGIN 70
SET TABS SPACES
```

When you invoke EVE using an initialization file, commands in the initialization file for margins, tabs stops, and other buffer-specific settings apply to the main (or first) buffer and to an EVE system buffer named \$DEFAULTS\$. The \$DEFAULTS\$ buffer is a template buffer: when you create a buffer—for example, by using the GET FILE command—EVE uses the settings of the \$DEFAULTS\$ buffer, so that each new buffer has the same settings. Thus, if your initialization file contains the command SET RIGHT MARGIN 70, each buffer you create will have that right margin.

To find out the default settings, use the SHOW DEFAULTS BUFFER command. To find out the settings of the buffer you are editing, use the SHOW command.

6.6 Saving Time and Keystrokes—Defining Keys in EVE

Although only a few keys are defined when you first use EVE, there are several ways that you can define keys to perform a wide range of editing functions. This section describes three ways to define keys in EVE:

- Emulating a keypad similar to the EDT or WPS editor
- Defining keys while using EVE
- Making permanent key definitions

Read this section if you want to learn about some of the methods of defining keys in EVE.

6.6.1 Using EVE to Emulate EDT

If you are familiar with the keypad available with the EDT editor, then you can readily set a similar environment while using EVE.

To use a keypad similar to the EDT keypad, do the following:

1. Enter the EVE editor using the following DCL-level command:

```
EDIT/TPU file-name
```
2. Press the DO key (or PF4). The *Command:* prompt is displayed at the bottom of the terminal.
3. Type SET KEYPAD EDT (as shown) and press Return.

```

Buffer: FILE-NAME.DAT | Write | Insert | Forward
Command: set keypad edt_

```

After you give this command, your keypad resembles the EDT keypad. Note that although your keypad is similar to the EDT keypad, you are still in the EVE

environment and some EDT functions (for example, line editing) are not available. However, you can continue to use any of the features that EVE provides.

For example, you can use the EDT keypad and take advantage of the multiple windows in EVE. You could set your keypad to EDT, as described above, create two windows, and then define a key to switch back and forth between the two windows.

Use the following procedure to set up your editing in this way:

1. Enter the EVE editor and set your keypad to EDT, as described in the previous example.
2. To create two windows on your terminal, press the DO key.

NOTE: Before you gave the SET KEYPAD EDT command, the PF4 key gave you the *Command:* prompt. To emulate EDT, the *Command:* prompt is redefined as the sequence PF1-KP7 after you give the SET KEYPAD EDT command.

At the *Command:* prompt, type TWO WINDOWS and press Return. Your screen will look like this:

```
[End of file]
Buffer: FILE-NAME.DAT      | Write | Insert | Forward
[End of file]
Buffer: FILE-NAME.DAT      | Write | Insert | Forward
```

3. Define a key that will let you switch back and forth between the windows, using the following procedure:
 - a. Press CTRL/K to indicate that you want to define a key. The bottom of your terminal looks as follows:

```
[End of file]
Buffer: FILE-NAME.DAT      | Write | Insert | Forward
Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.
```

6-20 Editing Text Files: Using EVE

- b. Press DO or the PF1-KP7 sequence to get the *Command:* prompt, and then type OTHER WINDOW, as shown, and then press Return.

```
[End of file]
Buffer: FILE-NAME.DAT | Write | Insert | Forward
Command: other window_
Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.
```

- c. Press CTRL/R to end the key definition. The bottom of your screen displays the following:

```
[End of file]
Buffer: FILE-NAME.DAT | Write | Insert | Forward
Press the key you want to use to do what was just learned: _
```

- d. Press any valid CTRL key or PF1- sequence, for example, CTRL/A.

Now, when you press CTRL/A, the cursor switches between the two windows on the screen.

If you want to edit (or read) two files at the same time, use the GET file-name command after you have created the two windows. The file that you name in the GET command is displayed in one window, and the original file that you were editing is displayed in the other window.

If you want to save this environment (that is, the EDT keypad emulation and the key that you defined to switch between windows), then do the following before you exit from EVE:

1. Get the *Command:* prompt by pressing DO or the sequence PF1-KP7.
2. Type SAVE EXTENDED TPU [directory]file-name, without giving a file type (for example, SAVE EXTENDED TPU [THOMAS]EVE-KEYDEFS), as shown, and then press Return:
3. Exit from EVE by pressing CTRL/Z to write the files that you have edited, or by typing QUIT at the *Command:* prompt. If you have made edits to more than one file when you press CTRL/Z, you will be asked if you want to write each file (except for the buffer in which the cursor is located—that file is automatically written).
4. After you exit from EVE, give the following command (or define a symbol) to use EVE with the EDT keypad and CTRL/A defined as switching between windows:

```
$ EDIT /TPU /SECTION=[THOMAS]EVE-KEYDEFS filename.ext
```


When you enter EVE, the EDT keypad is in place, and the CTRL/A key is defined as switching between windows. Note that only one window is on the screen when you enter EVE, and that you must first create two windows (with the TWO WINDOWS command) before you can meaningfully use CTRL/A.

6.6.2 Using EVE to Emulate WPS

If you are familiar with the keypad available with the WPS editor, then you can readily set a similar environment while using EVE. To use a keypad similar to the WPS keypad, do the following:

1. Enter the EVE editor using the following DCL-level command:
EDIT/TPU file-name
2. Press the DO key (or PF4). The "Command:" prompt is displayed at the bottom of the terminal.
3. Type SET KEYPAD WPS (as shown) and press Return.

```

Buffer: FILE-NAME.DAT | Write | Insert | Forward
Command: SET KEYPAD WPS_

```

After you give this command, your keypad resembles the WPS keypad, and you can use any of the features that EVE provides (for example, multiple windows). However, although your keypad resembles the WPS keypad, you are still in the EVE environment and some WPS functions may not be available (for example, only one ruler is active per document, rulers cannot be embedded in documents, and scrolling functions slightly differently).

If you want to automatically use the WPS keypad each time you enter the EVE editor, follow this procedure:

1. Enter the EVE editor and set your keypad to WPS, as described above.
2. Get the "Command:" prompt by pressing DO or the sequence PF1[.
3. Type SAVE EXTENDED TPU [directory]file name, without giving a file extension (for example, SAVE EXTENDED TPU [THOMAS]EVE-KEYDEFS), as shown, and then press Return.

```

Buffer: FILE-NAME.DAT | Write | Insert | Forward
Command: SAVE EXTENDED TPU [THOMAS]EVE-KEYDEFS_
WPS keypad defined (for more information, see help on WPS DIFFERENCES).

```

4. Exit from EVE by pressing CTRL/Z to write the files that you have edited, or by typing QUIT at the "Command:" prompt.

5. After you exit from EVE, give the following command (or define a symbol) to use EVE with the WPS keypad:

```
$ EDIT /TPU /SECTION=[THOMAS]EVE-KEYDEFS filename.ext
```

6.6.3 Defining a Key While Using EVE

There are two basic types of key definitions that you can make while using EVE:

1. A key that is defined to be a single, specific EVE function
2. A key that replicates a series of keystrokes, such as inserting text, a series of EVE functions, or both

You can define keys by the methods described in this section regardless of the type of keypad that you might be using.

6.6.3.1 Defining a Key to Perform a Single EVE Function

To define a key that performs a single EVE function, use the following procedure:

1. At the "Command:" prompt (press the DO key), type DEFINE and press Return.
2. Type the EVE command that you want to use, for example OTHER WINDOW, as shown, and then press Return. (The Reference Section contains a list of EVE commands.)

```
Buffer: FILE-NAME.DAT | Write | Insert | Forward  
EVE command: OTHER WINDOW
```

3. You are prompted to press the key that you want to define, as shown:

```
Buffer: FILE-NAME.DAT | Write | Insert | Forward  
Press the key you want to define: 
```

4. Press any valid control or PF1 sequence, and the key definition is complete.

When you define a key during an EVE editing session, the key definition is normally valid only until you exit from the EVE session. However, one way to make a key definition permanent is to use the following procedure, described in more detail in Section 6.6.1 and Section 6.6.2:

1. At the "Command:" prompt, type SAVE EXTENDED TPU [directory]filename, without giving a file extension (for example, SAVE EXTENDED TPU [THOMAS]EVE-KEYDEFS), and then press Return.
2. After you exit from EVE, give the following command (or define a symbol) to use EVE with the keys that you have defined (substituting the proper directory and file name):

```
$ EDIT /TPU /SECTION=[THOMAS]EVE-KEYDEFS filename.ext
```

6.6.3.2 Defining a Key to Perform a Series of Keystrokes

If you have a series of keystrokes that you repeat frequently, then you can save time and keystrokes by using a feature of EVE that lets you associate a set of keystrokes with a particular key sequence. With this feature, you can define a key to output a string of text, to execute a series of EVE functions, or to combine one or more text strings with one or more EVE functions.

When you are already in an EVE editing session, and you want to define a key that executes a series of keystrokes, you always follow the same general process:

1. At the "Command:" prompt, type LEARN and press Return.
2. Type the keystrokes that you want the defined key to repeat. You can insert text, give EVE commands, or use any keys that are defined.
3. When you have finished typing the keystrokes, press CTRL/R to signal EVE that the sequence is complete.
4. Press the key that you want to define, and the key definition is complete.

For example, suppose that you often type the words *International Development Organization*. To include this expression in your text simply by pressing CTRL/A, use the following steps:

1. Be sure your cursor is at a point where you want the text to appear first.
2. At the "Command:" prompt (press the DO key), type LEARN and press Return.
3. Type the text *International Development Organization*. Your screen now looks like the following:

```

                                     19 April 1990
Dr. Yvonne Curry
President
International Development Organization_

[End of file]
Buffer: FILE-NAME.DAT | Write | Insert | Forward
Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.

```

4. Press CTRL/R. The bottom of the terminal looks as follows:

```

[End of file]
Buffer: FILE-NAME.DAT | Write |
Press the key you want to use to do what was just learned:

```

5. Now press CTRL/A. The key definition is complete, and the message "Key sequence remembered" is displayed at the bottom of your screen. If you press

CTRL/A, the text *International Development Organization* is inserted in your buffer.

You follow exactly the same procedure to define a key that includes EVE functions. For example, suppose that you are editing the following text file, which has four columns of data. In this example, you want to eliminate the last two columns ("Price" and "Total") in each row:

Item	Quantity	Price	Total
Apples	20	1.00	20.00
Bananas	40	1.50	60.00
Beets	25	2.00	50.00
Carrots	30	2.00	60.00
Oranges	20	4.00	80.00
Peaches	10	3.00	30.00
Pears	5	6.00	30.00
Potatoes	50	1.00	50.00

[End of file]

Buffer: WEEKLY-ORDER.DAT

You could go through the procedure manually, which could be painstaking if your file contained a great deal of data. Alternatively, you could define a key to do most of the work for you.

To define the CTRL/D key to do the work, use the following steps:

1. Move the cursor to the beginning of the first line (the word *Item*).
2. Press the DO key to get the "Command:" prompt, and type LEARN.
3. Move the cursor ahead two words by giving the command "MOVE BY WORD" twice at the "Command:" prompt. If you have a key defined that already moves the cursor ahead one word (for example, if you have set your keypad to emulate the EDT or WPS editor, or if you have defined a key as described earlier in this section), you can simply press that key twice.
4. Now you must delete the rest of the text in the line. In EVE, a simple way to do this is to use the REMOVE (cut) function. Press the SELECT key, give the command END OF LINE (at the "Command:" prompt), and then press the REMOVE key. (If your terminal does not have the SELECT and REMOVE keys, then you can either give the commands SELECT and REMOVE at the "Command:" prompt, or you can use keys that have been defined as SELECT and REMOVE.)
5. Give the MOVE BY WORD command once again (or use a key defined as such), which moves the cursor to the beginning of the next line.
6. Press CTRL/R to signal the end of the learn sequence.
7. Press CTRL/D to define that key, and the key definition is complete.

When CTRL/D is pressed, the cursor moves forward two words, removes the remainder of the line, and moves the cursor to the beginning of the next line.

You can repeat virtually any series of keystrokes with the learn sequence, including searching for text, writing the contents of a buffer to a file, moving from one window to another, substituting text, or any other EVE function.

You can use most CTRL/ keys, PF1- sequences, and function keys F7 through F20 for the keys that you define. However, you cannot redefine the DO key, and you should not redefine any of the following keys:

```
HELP
CTRL/C
CTRL/O
CTRL/Q
CTRL/R
CTRL/S
CTRL/T
CTRL/U
CTRL/X
CTRL/Z
```

As with other keys defined during an EVE editing session, keys that you define with the LEARN command would normally not be valid after you exit from the EVE editor. However, you can keep and reuse keys that you define during an EVE session by following this same process that was described in preceding sections:

1. Give the command SAVE EXTENDED TPU [directory]filename at the "Command:" prompt; for example:

```
Peaches 10
Pears 5
Potatoes 50
[End of file]
Exit from: HEFNL:ORDER.DAT
Command: SAVE EXTENDED TPU [THOMAS]EVE-KEYDEFS_
```

2. After you exit from EVE, give the following command (or define a symbol) to use EVE with the keys that you have defined:

```
$ EDIT /TPU /SECTION=[THOMAS]EVE-KEYDEFS filename.ext
```

6.6.4 Using Startup Files to Define Keys

The TPU/EVE editor provides a wide range of features, and your editing environment can be defined in a number of ways. This section discusses two methods that you can use to set your editing environment:

- An EVE initialization file
- A TPU section file

6.6.4.1 EVE Initialization Files

An **EVE initialization file** lets you set your editing environment and define keys to be specific EVE commands. For example, you could use an EVE initialization file to set your keypad to an EDT-like environment whenever you use EVE, to set the left and right margins of your buffers, and to define keys to create two buffers and switch between them.

To use an EVE initialization file, use the following procedure:

1. Create the EDT initialization file, using any text editor.
2. Specify the initialization file that you want by using the /INITIALIZATION= qualifier in your EDIT/TPU command line. For example, if your EVE initialization file is [THOMAS]EVE\$INIT.EVE, then you would use the following DCL-level command to edit a file named REPORT.TEXT using your initialization file:

```
$ EDIT /TPU /INITIALIZATION=[THOMAS]EVE$INIT.EVE REPORT.TEXT
```

You can of course define a symbol in your login command file that would reduce the number of keystrokes that you need; for example:

```
$ EDIT :== EDIT /TPU /INITIALIZATION=[THOMAS]EVE$INIT.EVE
```

In the EVE initialization file, each line should contain a single command or key definition. The following sample shows an EVE initialization file that you could use or adapt to meet your needs:

```
SET KEYPAD EDT ①
SET LEFT MARGIN 8 ②
SET RIGHT MARGIN 72 ③
DEFINE KEY=CTRL/D TWO_WINDOWS ④
DEFINE KEY=CTRL/A OTHER_WINDOW ⑤
DEFINE KEY GOLD/Q QUIT ⑥
```

In this EVE initialization file, the lines have the following meanings:

- ① EVE provides an EDT-like keypad.
- ② The left margin of your text is set at 8.
- ③ The right margin of your text is set at 72.
- ④ When you press the CTRL/D key, two windows are created on your terminal.
- ⑤ When you press the CTRL/A key, the cursor switches from one window on your terminal to the other window.
- ⑥ When you press the GOLD-Q sequence, you quit the editing session without writing the file.

Initialization files are simple to create and use. However, you may also want to use section files to define your EVE editing environment, because they allow you to define more complicated key sequences.

When you use the TPU/EVE editor, you can specify an initialization file (and no section file), a section file (and no initialization file), or you can specify both an initialization file and a section file.

6.6.4.2 EVE/TPU Section Files

A **section file** lets you define your EVE/TPU editing environment much more fully than with an EVE initialization file. This section describes how to create one type of section file.

To create and use an EVE/TPU section file, use the following procedure:

1. Create an **EVE/TPU command file** using a text editor. The command file is a text file that can be read on your terminal or printed.
2. Use the SAVE command in EVE/TPU to generate a section file from the command file. You cannot read a section file on your terminal, but EVE/TPU uses the information in the section file to create your editing environment.
3. Specify the section file that you want by using the /SECTION qualifier in your EDIT/TPU command line. For example, if your EVE/TPU section file is [THOMAS]EVE-KEYDEFS.TPU\$SECTION, then you would use the following DCL-level command to edit a file named REPORT.TEXT using your section file:

```
$ EDIT /TPU /SECTION=[THOMAS]EVE-KEYDEFS REPORT.TEXT
```

(Note that you do not need to specify a file extension for a section file when the default, TPU\$SECTION, is used.)

As always, you can define a symbol similar to the following in your login command file to expedite the process:

```
$ EDIT := EDIT /TPU /SECTION=[THOMAS]EVE KEYDEFS
```

The EVE/TPU section file is built from a command file that you create with a text editor. One type of command file (and subsequent section file) that sets up your environment and defines keys uses the following process:

1. Begin your command file with the following line:

```
procedure tpu$local_init
```

2. Use valid TPU commands (such as DEFINE_KEY) and the proper syntax for each command. The sample command file shown later in this section includes examples of syntax for defining keys that perform EVE commands, insert text, and set certain parameters.

3. End your command file with the following two lines:

```
endprocedure;
tpu$local_init;
```

4. Edit the command file using EVE. At the "Command:" prompt, type **EXTEND TPU ***, as shown, and press Return.

```
define_key ("eve_set_left_margin(8)", key_name('g', shift_key));
define_key ("eve_set_right_margin(79)", key_name('r', shift_key));
define_key ("eve_buffer('')", f17, "buffer", eve$x_user_keys);
set(margins, current_buffer, 1, 72);
endprocedure;
tpu$local_init;
[End of file]
Buffer: EVE-KEYDEFS.TPU | Write | Insert | Forward
Command: EXTEND TPU *_
```

5. At this point, a series of messages flashes across the bottom of your screen. If no serious errors are found, the message *EVE extended.* is displayed. You can see the list of messages displayed by typing **EVE BUFFER MESSAGES** at the "Command:" prompt. If the attempt to extend EVE was unsuccessful because of one or more errors, edit your command file to correct the errors and redo this step.

If you are ready to proceed with the next step, make sure that you are in the buffer with your command file. If your command file is not displayed on the terminal, press the DO key and type the command **BUFFER command-file-name** (for example, **BUFFER EVE KEYDEFS** if your command file is named **EVE-KEYDEFS.TPU**).

6. Press the DO key, and give the command **SAVE section-file-name**; it is not necessary to specify a file extension, because the default file extension (**TPU\$SECTION**) is automatically supplied. For example, if you wanted your section file to be named **[THOMAS]EVE-KEYDEFS.TPU\$SECTION**, you type the command as shown, and then press Return:

```
define_key ("eve_set_left_margin(8)", key_name('g', shift_key));
define_key ("eve_set_right_margin(79)", key_name('r', shift_key));
define_key ("eve_buffer('')", f17, "buffer", eve$x_user_keys);
set(margins, current_buffer, 1, 72);
endprocedure;
tpu$local_init;
[End of file]
Buffer: EVE-KEYDEFS.TPU | Write | Insert | Forward
Command: SAVE [THOMAS]EVE-KEYDEFS_
EVE extended.
```

7. Exit from EVE by pressing the DO key, typing **EXIT**, and pressing Return.
8. Now, to use the key definitions that you have created, include the following lines in your login command file:

```
$ ED*IT ::= EDIT /TPU /SECTION=[THOMAS]EVE-KEYDEFS
```

If you want to use the keys that you defined in your command file with an EDT-like keyboard, do the following:

1. Create the command and section files as described in the previous example.
2. Create an initialization file that includes only the following line:

```
SET KEYPAD EDT
```
3. If the section and initialization files are named [THOMAS]EVE-KEYDEFS.TPU\$SECTION and [THOMAS]EVE-INIT.EVE, respectively, then include the following line in your login command file:

```
EDIT ::= EDIT /TPU /SECTION=[THOMAS]EVE-KEYDEFS /INIT=[THOMAS]EVE-INIT
```
4. After you execute your login command file, you use the key definitions specified in your section file and the EDT-like keyboard specified in your initialization file when you use the EVE editor (by using the command EDIT).

NOTE: If you want to use this same editing environment by default whenever you send or reply to mail, then you should also include the following two lines in your login command file:

```
$ assign callable_tpu      mail$edit:
$ mail ::= mail /edit=(send,reply)
```

6.6.4.3 Nondefinable Keys

You cannot define any of the following keys:

```
F1 through F6
COMPOSE CHARACTER
CTRL (by itself)
Return or CTRL/M
BREAK
ESCAPE or CTRL/[
LOCK or CAPS LOCK
NO SCROLL
SET-UP
SHIFT
```

Also, EVE does not let you define typing keys on the main keyboard (except in combination with the GOLD key), a key defined as DO if it is the only key defined as DO, or the key currently set as GOLD, if any.

Digital recommends that you do not define the following keys and control keys. Some of these control keys cannot be defined unless you set terminal characteristics accordingly.

```
DELETE or <X> (which EVE defines as DELETE)
HELP or on VT100 terminals, PF2
CTRL/B (which EVE defines as RECALL)
CTRL/C
CTRL/O
CTRL/Q
CTRL/R (which EVE defines as REMEMBER, to end a learn sequence)
CTRL/S
CTRL/T
```

CTRL/U (which EVE defines as ERASE START OF LINE)

CTRL/V (which EVE defines as QUOTE)

CTRL/X

CTRL/Y

If you redefine CTRL/B or CTRL/R, you should define other keys as RECALL and REMEMBER, because those commands can only be executed by a key press.

6.6.4.4 Sample EVE/TPU Command File

The following example shows an EVE command file that you can use or adapt to meet your needs. To use the key definitions and margin settings in this file, you should follow the steps described in Section 6.6.4.2.

Note the following syntax rules for defining a key in a command file:

- Use the DEFINE_KEY command to begin the key definition.
- The key definition begins with an open parenthesis.
- The first part of the key definition uses EVE and TPU commands to describe the text to be inserted and/or the actions to be taken; this part of the key definition must be enclosed in quotation marks. To use an EVE command, use the syntax EVE_command_name (for example, EVE_OTHER_WINDOW). To insert text, use the COPY_TEXT command, with the text enclosed in parentheses and single quotation marks, as shown throughout the sample command file. If you use more than one command in this section, separate commands using a semicolon.
- The second part of the key definition specifies the key that is to be defined.
- End the key definition with a close parenthesis and a semicolon.

Note the syntax that you use for describing a key sequence with the DEFINE_KEY statement in a command file, as in this example for specifying the sequence PF1-M:

```
KEY_NAME('U', SHIFT_KEY)
```

```
procedure tpu$local_init ❶
```

```
set (shift_key, pf1); ❷
```

```
define_key ("copy_text('International Development Organization')", ctrl_n_key); ❸
```

```
define_key ("eve_return; eve_return; copy_text('<p>')", ctrl_p_key); ❹
```

```
define_key ("eve_return; eve_return; copy_text('<list>(unnumbered)');" + ❺  
"eve_return; copy_text('<le>')", key_name('m', shift_key));
```

```
define_key ("eve_return; eve_return; copy_text('<endlist>'); eve_return;" + ❻  
"copy_text('<p>'); eve_return; eve_return",  
key_name('n', shift_key));
```

```

define_key ("eve_two_windows", ctrl_d_key); 7
define_key ("eve_other_window", ctrl_a_key); 8
define_key ("eve_one_window", key_name('o', shift_key)); 9
define_key ("eve_fill_paragraph", ctrl_f_key); 10
define_key ("eve_lowercase_word", key_name('l', shift_key)); 11
define_key ("eve_uppercase_word", key_name('u', shift_key)); 12
define_key ("eve_quit", key_name('q', shift_key)); 13
define_key ("eve_exit", ctrl_z_key); 14

define_key ("eve_move_left; eve_move_left; eve_select; eve_move_right;" + 15
           "eve_remove; eve_move_right; eve_insert_here", ctrl_e_key);
define_key ("eve_set_left_margin(8)", key_name('g', shift_key)); 16
define_key ("eve_set_right_margin(79)", key_name('r', shift_key)); 17
define_key ("eve_buffer('')", f17, "", eve$x_user_keys); 18

set(margins,current_buffer,1,72); 19
endprocedure; 20
tpu$local_init; 21

```

In this example, line 1 is the text with which you should begin this command file. Line 2 defines PF1 (also known as the GOLD key) as the key used in a 2-key sequence. You should define the shift key before you make any other key definitions in your command file.

The following table lists the keys that are defined by the command file in the previous example:

Line Number	Key or Sequence	Action Taken When Key or Sequence is Pressed
3	CTRL/N	The text <i>International Development Organization</i> is inserted in the editing buffer.
4	CTRL/P	Two carriage returns and the text <P> are inserted in the current editing buffer.
5	PF1-M	Two carriage returns and the following text are inserted in your editing buffer:
		<list> (unnumbered)
		<le>

Line Number	Key or Sequence	Action Taken When Key or Sequence is Pressed
6	PF1-N	Two carriage returns, the following text, and then two more carriage returns are inserted in your editing buffer: <pre><endlist> <p></pre>
7	CTRL/D	A new window is created on your terminal screen.
8	CTRL/A	The cursor moves from the current window to the other window.
9	PF1-O	A single window is displayed on the terminal, using the current buffer.
10	CTRL/F	The current paragraph is filled.
11	PF1-L	The current word is set to all lowercase.
12	PF1-U	The current word is set to all uppercase.
13	CTRL/Q	You exit from the file and your changes are not saved.
14	CTRL/Z	The file is written and you exit from EVE.
15	CTRL/E	The two characters that immediately precede the cursor are transposed. For example, to change <i>teh</i> to <i>the</i> with a single keystroke, you could use this CTRL/E key definition.
16	PF1-G	The left margin is set to 8. (The default left margin setting is set to 1 in a subsequent statement in this command file.)
17	PF1-R	The right margin is set to 79. (The default right margin setting is set to 72 in a subsequent statement in this command file.)
18	F17	EVE prompts you for a buffer name. When you type a buffer name and press Return, EVE switches to that buffer. (Note that this DEFINE_KEY statement uses a third and fourth section that had not previously been used in this example. These parts of the DEFINE_KEY statement are optional, but they must be used in this example as shown.)

The final lines of the sample command file (line numbers 19 through 21) set the default left and right margins for any editing buffer to 1 and 72, and provide the closing context.

6.7 Using DCL Within EVE

You can execute a DCL command from within EVE, or you can use a subprocess to switch between the DCL command level and an EVE editing session quickly.

6.7.1 Executing a DCL Command

To enter a DCL command from within EVE, enter the EVE command DCL with the DCL command you want to execute, and press Return. The message *Creating DCL subprocess . . .* appears in the message window.

6.7.2 Creating a Subprocess

You can create a subprocess to switch between an EVE editing session and DCL command level without terminating your editing session. To create a subprocess, enter the SPAWN command. EVE suspends the current editing session and connects your terminal to a new VMS subprocess. The DCL prompt (\$) appears on your terminal screen.

NOTE: The SPAWN and ATTACH commands are supported on DECwindows only if you invoke EVE with the /DISPLAY=CHARACTER_CELL qualifier (which is the default).

The most common reasons to spawn a subprocess are to invoke the Mail Utility and to run screen-oriented programs, although your subprocess can invoke any VMS utility or execute any DCL command.

To return to your editing session, log out of the subprocess by entering the DCL command LOGOUT. EVE resumes the editing session, and the cursor appears in the location it occupied before you spawned the subprocess.

6.8 Converting from EDT to EVE

If you are accustomed to the EDT editor, you can customize EVE to work in similar ways by using a section file or an initialization file (or both), or by using VAXTPU procedures.

Typically, you save key definitions, learn sequences, and other extensions in a section file (created with the SAVE EXTENDED EVE command), and use an EVE initialization file to set editing preferences or private defaults, such as margins and tabs, that are not saved in the section file. The following are hints on converting from EDT to EVE.

Use the SET KEYPAD EDT Command

The SET KEYPAD EDT command defines several keys to emulate EDT. You can put the command in your EVE initialization file, or save the keypad setting in your section file.

Define Keys for EVE Commands

Use **DEFINE KEY** commands to define keys that are not otherwise defined by **SET KEYPAD EDT**. Put the key-definition commands in your initialization file, or save the definitions in your section file. For example, the following sets of EDT and EVE key definitions are equivalent:

- In EDT:

```
DEF KEY gold b AS "ext show buffer."
DEF KEY gold l AS "chglw."
DEF KEY gold u AS "chguw."
DEF KEY gold 2 AS "iInteroffice Memo^Z."
DEF KEY gold 10 AS "ext find=?.."
DEF KEY 7 AS "50l."
DEF KEY gold 9 AS "cutsr paste."
DEF KEY cont n AS "ext quit."
DEF KEY func 34 AS "shl."
```

- In EVE:

```
DEF KEY= gold-b show buffers
DEF KEY= gold-l lowercase word
DEF KEY= gold-u uppercase word
DEF KEY= gold-e2 tpu eve$insert_text ("Interoffice Memo")
DEF KEY= gold-pf2 buffer
DEF KEY= kp7 tpu move_vertical (+50)
DEF KEY= gold-kp9 store text
DEF KEY= ctrl/n quit
DEF KEY= f20 shift right 8
```

Note the differences between EDT and EVE in some key names, as well as differences in command names. For more information about key names, see Section 6.4.3.

Set Bound Cursor Motion

Put the **SET CURSOR BOUND** command in your EVE initialization file to enable an EDT-style bound cursor that follows the shape or flow of your text. By default, EVE uses a free cursor, which can move anywhere in the buffer regardless of whether text is already there.

Set the Right Margin for Wrapping Text

Put the **SET RIGHT MARGIN** command in your EVE initialization file to set a wrap limit for entering text and for **FILL** commands. For example, the following EDT and EVE commands are equivalent:

- In EDT:

```
SET WRAP 70
```

- In EVE:

```
SET RIGHT MARGIN 70
```

(The EVE command SET WRAP corresponds to the EDT command SET NOTRUNCATE.)

Set Scroll Margins for Moving the Cursor

Put the SET SCROLL MARGINS command in your EVE initialization file to set distances for scrolling to begin automatically as you move the cursor up or down. For example, with a 24-line terminal screen (21-line main window), the following EDT and EVE commands are equivalent:

- In EDT:

```
SET CURSOR 5:15
```

- In EVE:

```
SET SCROLL MARGINS 5 6
```

Note that EVE scroll margins are measured from the top and bottom respectively, whereas in EDT, both are measured from the top. You can specify numbers of lines or percentages of the window size. Also, the size of the EVE main window depends on your terminal settings. For example, on a workstation, the EVE main window may be longer than 21 lines.

Convert EDT Macros to VAXTPU Procedures

Use VAXTPU procedures in place of EDT macros. Create a buffer containing the procedures and then compile the procedures with EXTEND commands, or put the procedures in a VAXTPU command file and then invoke EVE with the /COMMAND qualifier. In either case, you can save the compiled procedures in your section file. The following examples show a macro from an EDT startup file translated into a VAXTPU procedure. Each creates a new command, WIDEN, which sets the display to 132 columns and sets the right margin to 120.

- EDT macro:

```
FIND -widen
INSERT;SET SCREEN 132
INSERT;SET WRAP 120
FIND =main.
```

- VAXTPU procedure:

```
PROCEDURE eve_widen;
  EVE_SET_WIDTH (132);
  EVE_SET_RIGHT_MARGIN (120);
ENDPROCEDURE;
```

To execute the macro or procedure, do the following commands:

- In EDT:


```
* DEFINE MACRO widen
* WIDEN
```
- In EVE:


```
Command: EXTEND EVE widen
Command: WIDEN
```

Alternatively, use the learn command to bind the corresponding EVE commands to a single key; you can then save the key definition in your section file. Another method is to put the corresponding EVE commands in an initialization file that you can use during an editing session (see the description of the @ command).

Convert EDT Nokeypad Statements to VAXTPU Procedures

EDT macros and key definitions that use nokeypad specifiers can usually be converted into VAXTPU procedures or into LEARN sequences. The following examples show an EDT key definition using nokeypad mode and the corresponding VAXTPU procedure and key definition. In each case, you define COMMA on the numeric keypad to transpose or swap the current and previous character. Note that -C in EDT nokeypad statements can be translated as MOVE_HORIZONTAL (-1) in VAXTPU procedures.

- In EDT:


```
DEFINE KEY 19 AS "-c dlc +c undc."
```
- In VAXTPU:


```
PROCEDURE user_transpose
  LOCAL swap_this;

  swap_this := ERASE_CHARACTER (1);

  MOVE_HORIZONTAL (-1);
  EVE$INSERT_TEXT (swap_this);
  RETURN (TRUE);
ENDPROCEDURE;

EVE$DEFINE_KEY ("user_transpose", COMMA, , EVE$X_USER_KEYS);
```

Use the WPS Keypad Ruler Key to Adjust Tab Stops

Setting the EDT keypad does not define keys for EDT-style tab adjustment. However, you can get similar effects by defining a key for the WPS keypad Ruler key (GOLD-R) and then using the ruler to add or delete tab stops.

For example, the following command defines F20 as the WPS Ruler key (without having to enable the WPS keypad):

Command: DEFINE KEY= F20 WPS GOLD-R

Then, to add or delete tab stops, do the following:

1. Press whatever key you have defined as the Ruler key.
EVE displays a ruler at the bottom of the current window (just above the status line for the window). The cursor appears in the ruler. Tab stops are marked with a *T*.
2. Put the cursor where you want to add or delete a tab stop. For example, you can press the left and right arrow keys to move to a particular column in the ruler, or press the TAB key to move to the next tab stop (*T*) in the ruler.
3. Type a *T* or *t* at that location to set the tab stop or, if there is already a tab there, to delete it. The new tab stops are immediately applied to the buffer you were editing.
4. Repeat steps 2 and 3 to add or delete other tab stops.
5. To exit from the ruler and resume editing, press Return or GOLD-RETURN.

6.9 EVE Command Summary

Section 6.9 shows EVE commands and the keys that are predefined by default for those commands.

Table 6-3: EVE Commands and Default Predefined Keys

Command	Key	What It Does
BOTTOM	GOLD-↓	Moves the cursor to the end of the current buffer.
BUFFER	None	Puts a specified buffer in the current window and moves the cursor to the last place it occupied in the buffer. (Buffers are storage areas that exist only during an editing session.) If the specified buffer does not exist, creates a new buffer and moves the cursor to the start of the buffer.
CAPITALIZE WORD	None	Capitalizes a single word or each word in the text highlighted by FIND or SELECT.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
CENTER LINE	None	Centers the current line between the left and right margins. The cursor moves with the line, remaining on the same character as the line moves.
CHANGE DIRECTION	None	Changes the direction of the current buffer. The direction of the buffer is shown in the status line.
CHANGE MODE	CTRL/A. Also, F14 on VT300-series and VT200-series terminals; Enter on VT100-series terminals	Changes the current editing mode as displayed on the highlighted status line. In insert mode, EVE inserts text at the current character position, moving existing text to accommodate the insertion. In overstrike mode, EVE overwrites text at the current position.
COPY or STORE TEXT	GOLD-Remove	Copies text that was marked with SELECT or FIND, putting it in the Insert Here buffer. Text that is copied is not removed from its original position.
CUT		Same as REMOVE
DELETE	⌫ or Delete	Erases the character to the left of the cursor. In insert mode, the rest of the line moves left one character to close the space. In overstrike mode, the erased character is replaced by a space. At the start of a line, DELETE erases the carriage return for the previous line (regardless of mode) and the current line moves up.
DELETE BUFFER	None	Deletes a buffer you specify by name.
DELETE WINDOW	None	Deletes the window the cursor is in, if you are using more than one window.
END OF LINE	CTRL/E or GOLD-->	Moves the cursor to the end of the current line.
ENLARGE WINDOW	None	Enlarges the window the cursor is in by a specified number of lines. For example, ENLARGE WINDOW 5 enlarges the window by five lines. The adjacent window shrinks accordingly.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
ERASE CHARACTER	None	Erases the character the cursor is on. In insert mode, the rest of the line moves to the left one character to close the space. In overstrike mode, the erased character is replaced by a space. If the cursor is at the end of the line, the carriage return is erased—regardless of the mode—and the next line moves up.
ERASE LINE	None	Erases from the current character to the end of the line, appending the next line to the end of the current line. If the cursor is at the end of the line, only the carriage return is erased and the next line moves up.
ERASE PREVIOUS WORD	None	Erases the previous word or the word the cursor is on. If the cursor is between words or on the first character of a word, the previous word is erased. If the cursor is in the middle of a word, all of that word is erased (same as ERASE WORD). If the cursor is at the start of a line, the carriage return at the end of the previous line is erased and the current line moves up.
ERASE START OF LINE	CTRL/U	Erases characters left of the cursor to the start of the line.
ERASE WORD	CTRL/J. Also, F13 on VT300-series and VT200-series terminals; COMMA (on the keypad) on VT100-series terminals	Erases the current word or, if the cursor is between words, erases the next word. If the cursor is at the end of the line, only the carriage return is erased and the next line moves up.
FILL	None	Reformats the current paragraph, select range, or found range, according to the margins of the buffer, so the maximum number of words fits on a line.
FILL PARAGRAPH	None	Reformats the paragraph the cursor is in, according to the margins set for the buffer.
FILL RANGE	None	Reformats the current select range or found range, according to the current margin settings.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
FIND	Find on VT300-series and VT200-series terminals	Searches the current buffer for the text string you specify and highlights the found text. The text that is highlighted is called the found range .
FIND NEXT	Find-Find on VT300-series and VT200-series terminals	Searches for the string of text you last specified with the FIND, REPLACE, or WILDCARD FIND command.
FIND SELECTED	Find-Insert Here on VT300-series and VT200-series terminals	Searches for a string of text you have selected, rather than for a typed string.
FORWARD	None	Sets the direction of the current buffer to forward (that is, to the right and down). The direction of the buffer is shown in the status line. FORWARD is the default setting when you enter EVE.
GET FILE or OPEN	None	Puts the specified file into the current window and puts the cursor at the beginning of the buffer. If the file does not exist, EVE puts an empty buffer in the current window.
GO TO	None	Returns the cursor to the location labeled by the MARK command. If the labeled location is found in another buffer, EVE moves the cursor to the other buffer and puts that buffer into the current window.
INCLUDE FILE	None	Inserts the contents of the specified file into the current buffer at the line above the cursor location.
INSERT HERE or PASTE	Insert Here on the minikeypad on VT300-series and VT200-series terminals; KP9 on VT100-series terminals	Inserts the text you copied or removed.
INSERT PAGE BREAK	CTRL/L	Inserts a form-feed character at the current position to mark the beginning of a new page. A page break appears as a small double F (F _F) and is always on a line by itself.
LINE	None	Moves the cursor to the beginning of the specified line number in the current buffer.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
LOWERCASE WORD	None	Changes the current word, select range, or found range to lowercase.
MARK	None	Puts an invisible mark at the current cursor location. The mark exists for the rest of an editing session or until you change it but is not saved when you exit.
MOVE BY LINE	F12 on VT300-series and VT200-series terminals; MINUS (on the keypad) on VT100-series	In forward direction: moves the cursor to the end of the current line or, if the cursor is already at the end of a line, to the end of the next line. In reverse direction: moves the cursor to the beginning of the current line or, if the cursor is already at the beginning of a line, to the beginning of the previous line.
MOVE BY PAGE	None	Moves the cursor to the next or previous page break (form feed), depending on the current direction. If there is no page break in the current direction, the cursor moves to the bottom or top of the buffer.
MOVE BY WORD	None	In forward direction: moves the cursor to the beginning of the next word or, if the cursor is already at the end of a line, to the beginning of the next line. In reverse direction: moves the cursor to the beginning of the previous word or, if the cursor is already at the beginning of a line, to the end of the previous line.
MOVE DOWN	↓. Also, KP2 on VT100-series terminals	Moves the cursor down one line.
MOVE LEFT	←. Also, KP1 on VT100-series terminals	Moves the cursor one character or column to the left.
MOVE RIGHT	→. Also, KP3 on VT100-series terminals	Moves the cursor one character or column to the right.
MOVE UP	↑. Also, KP5 on VT100-series terminals	Moves the cursor up one line.
NEW	None	Creates a new buffer, putting it in the current EVE window, and moves the cursor to the top of the new buffer. The new buffer is named MAIN. If a buffer named MAIN already exists, EVE prompts you for a buffer name.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
NEXT BUFFER	None	Puts your next buffer (if there is one) into the current window, returning the cursor to your last position in that buffer. This lets you move between buffers without typing buffer names.
NEXT SCREEN	E6 on VT300-series and VT200-series terminals; KP0 on VT100-series terminals	Scrolls forward in the current buffer by the number of lines in the current window minus one. For example, if the current window is 12 lines long, the NEXT SCREEN command scrolls the cursor forward 11 lines.
NEXT WINDOW		Same as OTHER WINDOW
ONE WINDOW	None	Restores the window the cursor is in as a single, large window. EVE deletes all other windows from the screen. However, the buffers associated with those windows are not deleted.
OPEN		Same as GET FILE
OPEN SELECTED	None	Opens a file whose name you have selected or found. This command is the same as using the GET FILE or OPEN command without having to type the file name.
OTHER WINDOW	GOLD-Next Screen	Moves the cursor to the next window on your screen, if there is one. The cursor appears in the last location it occupied in that window.
PAGINATE	None	<p>Inserts a "soft" page break for a 54-line page. A soft page break appears as a form feed followed by the null character - (F_F N_L).</p> <p>When the PAGINATE command is entered, EVE moves back to the previous page break (if any) then checks ahead for page breaks within the next 54 lines. If any soft breaks are found within those 54 lines, EVE removes them. EVE then moves down 54 lines, inserts a soft break, and puts the cursor on the next line. The soft break is inserted on a line by itself. If a hard page break is found within the 54 lines, EVE stops on the line after the hard break, in case you want to erase the break.</p>

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
PREVIOUS SCREEN	Prev Screen on VT300-series and VT200-series terminals; PERIOD (on the keypad) on VT100-series terminals	Scrolls backward in the current buffer by the number of lines in the current window minus one. For example, if the current window is 12 lines long, the PREVIOUS SCREEN command scrolls the cursor backward 11 lines.
PREVIOUS WINDOW	None	Puts the cursor in the previous (or other) window.
QUOTE	CTRL/V	Lets you insert nonprinting characters or control codes.
REMOVE or CUT	Remove on the minikeypad on VT200-series and VT300-series terminals; KP8 on VT100-series terminals	Removes the text that was marked with SELECT or highlighted by FIND, and places it in the Insert Here buffer.
RESET	GOLD-Select	<p>Cancels any of the following and resets the direction of the buffer to forward:</p> <ul style="list-style-type: none"> • Highlighting of a select or found range • A press of the GOLD key (or GOLD-number combination for a repeat count) • An incomplete or recalled command line, or Choices buffer display • The output of SHOW, SHOW DEFAULTS BUFFER, SHOW SUMMARY, or SHOW WILDCARDS, thereby returning you to the buffer you were working in
RESTORE	GOLD-Insert Here	Reinserts, at the current cursor position, the word, or line that you erased most recently with an EVE command or editing key. RESTORE does not restore single characters.
RESTORE CHARACTER	None	Reinserts, at the current cursor position, the character you have erased most recently with an EVE command or editing key. In overstrike mode, the restored character replaces the character the cursor is on. In insert mode, the restored character is inserted at the cursor position and existing text moves to accommodate it.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
RESTORE LINE	None	Reinserts, at the current cursor position, the line that you have erased most recently with an EVE command or editing key.
RESTORE SELECTION	None	Reinserts the text erased with a pending delete operation.
RESTORE WORD	GOLD-F13 on VT200-series and VT300-series terminals; none on VT-100 series terminals	Reinserts, at the current cursor position, the word that you have erased most recently with an EVE command or editing key.
REVERSE	None	Sets the direction of the current buffer to reverse, that is, to the left and up. The direction of the buffer is shown in the status line.
SAVE FILE	None	Writes the contents of the current buffer to the file associated with the buffer without ending the editing session. If you do not specify a file name with the SAVE FILE command, EVE prompts you for an output file specification. Similar to WRITE FILE.
SAVE FILE AS	None	Writes the contents of the current buffer to the file you specify without ending the editing session. Thus, if you are editing a file named FIRST.DAT you can save it as SECOND.TXT. This command does not change the name of the buffer. It does, however, associate the buffer with the file you name so any subsequent SAVE FILE or WRITE FILE commands or an EXIT command write the buffer to the file you named. This command requires you to supply a file specification.
SELECT	Select on VT200-series and VT300-series terminals; KP7 on VT100-series terminals	Marks text (highlighting it in reverse video) from the initial cursor location to wherever you move the cursor. The text that is highlighted is called the select range . To cancel the selection, enter the SELECT command again or use RESET.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
SELECT ALL	None	Marks all text (highlighting it in reverse video) in the current buffer regardless of the cursor position. The text that is highlighted is called the select range . To cancel the selection, enter the SELECT command or use RESET. Pending delete is temporarily disabled when the SELECT ALL command is used to avoid accidentally erasing all of the buffer.
SET BUFFER	None	Lets you specify the editing status of the buffer: whether the buffer can be modified or can be written to a file when you exit from EVE.
SET CURSOR BOUND	None	Makes the cursor follow the flow of text. The cursor cannot move into an unused portion of the buffer. Similar to cursor behavior in EDT, WPS, and other editors.
SET CURSOR FREE	None	Default setting. Allows the cursor to be put anywhere in the buffer and text can be entered there.
SET FIND NOWHITESPACE	None	Default setting. Sets FIND and WILDCARD FIND commands to match tabs and spaces exactly as you specify in the search string, and to search for strings that are entirely on one line.
SET FIND WHITESPACE	None	Sets FIND and WILDCARD FIND commands to treat spaces, tabs, and up to one line break as "white space" so you can search for strings of two or more words regardless of how they are separated.
SET LEFT MARGIN	None	Sets the left margin in the current buffer. The left margin must be greater than 0 but less than the right margin. By default, the left margin is 1 (leftmost column).
SET NOPENDING DELETE	None	Default setting. Disables deletion of selected text when you use DELETE or type new text. If you select text in the buffer, typing new text adds characters to the select range and using DELETE erases only the character to the left of the cursor.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
SET NOWRAP	None	Disables word wrapping at the right margin of the buffer. You must start new lines by pressing Return or by using the FILL command.
SET PENDING DELETE	None	Enables pending delete, which lets you quickly erase blocks of text. First, enable pending delete, then use the SELECT command to choose the text you want to erase. Erase the text by pressing the Delete key (or any other typing key). To reinsert what you deleted, move the cursor where you want the text to be and enter the RESTORE SELECTION command. The default is SET NOPENDING DELETE.
SET RIGHT MARGIN	None	Sets the right margin for the current buffer. The right margin must be greater than the left margin. By default, the right margin is one less than the width. The width is typically 80, so the default margin is typically 79.
SET PARAGRAPH INDENT	None	Specifies the number of spaces to be added to or subtracted from the first line of paragraphs you create or reformat. The default is 0 (no indent).
SET TABS AT	None	Sets tab stops at the columns that you specify. The column numbers must be in ascending order and separated by spaces. By default, tab stops are set every eight columns. The command does not affect the hardware tab settings of your terminal.
SET TABS EVERY	None	Sets tab stops at the specified interval. By default, tab stops are set every eight columns. The command does not affect the hardware tab settings of your terminal.
SET TABS INSERT	None	Default setting. Changes the tab mode so that EVE inserts a tab character at the current column when you press the Tab key. The cursor and text move to the next tab stop.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
SET TABS SPACES	None	Changes the tab mode to insert an appropriate number of spaces, rather than a tab character, when the Tab key is pressed. Previously existing tab characters are not affected.
SET TABS MOVEMENT	None	Changes the tab mode so the Tab key becomes a cursor-movement key. Pressing the Tab key moves the cursor to the next tab stop but does not insert a tab character.
SET TABS VISIBLE	None	Displays a tab character as a visible character on the screen, appearing as a small \overline{H}_T (horizontal tab).
SET TABS INVISIBLE	None	Default setting. Makes a tab character invisible on the screen, appearing as white space.
SET WILDCARD ULTRIX	None	Enables ULTRIX patterns for WILDCARD FIND.
SET WILDCARD VMS	None	Default setting. Enables VMS patterns for WILDCARD FIND.
SET WIDTH	None	Sets the width of lines displayed on the screen. Specify width as a positive integer. By default, the screen width is your terminal setting. (It is typically 80 columns.) If the width is set greater than 80, EVE sets the terminal to 132-column mode for the current editing session. When you exit from EVE, the terminal is restored to the default setting. Setting the width changes the display of text in all windows.
SET WRAP	None	Default setting. Enables word wrapping at the right margin of the buffer. EVE starts new lines without your pressing Return or using the FILL command.
SHIFT LEFT	None	Moves the window the cursor is in to the left a specified number of columns. The SHIFT LEFT command can be used only to reverse the effect of the SHIFT RIGHT command.

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
SHIFT RIGHT	None	Moves the window the cursor is in to the right a specified number of columns, allowing you to view columns of characters that do not currently appear on the terminal screen.
SHRINK WINDOW	None	Shrinks the window the cursor is in by a specified number of lines. For example, SHRINK WINDOW 5 shrinks the window by five lines. The adjacent window expands accordingly.
SHOW	None	Displays information about the buffers you have created during the editing session. If more than one buffer is active in your editing session, the SHOW command displays information about the buffer you are currently editing. For information about the other active buffers, press the Do key. To resume editing, press any other key.
SHOW BUFFERS	None	Lists the buffers you have created during an editing session. You can move the cursor through the list and specify a particular buffer for viewing by using the Select key.
SHOW SYSTEM BUFFERS	None	Lists the system buffers created by EVE, such as the Message buffer, Help buffer, Insert Here buffer, and \$RESTORE\$ buffer. You can move the cursor through the list and specify a buffer for viewing by using the Select key.
SHOW WILDCARDS	None	Lists the wildcard patterns you can use with WILDCARD FIND, either VMS or ULTRIX.
SPLIT WINDOW	None	Splits the window the cursor is in, forming two smaller windows. You can divide the window into more than two parts by specifying a number with the command. For example, SPLIT WINDOW 3 splits the window into three windows.
START OF LINE	CTRL/H. Also, GOLD-←	Moves the cursor to the beginning of the current line.
STORE TEXT		Same as COPY.
TOP	GOLD-↑	Moves the cursor to the beginning of the current buffer (upper left corner).

(continued on next page)

Table 6-3 (Cont.): EVE Commands and Default Predefined Keys

Command	Key	What It Does
TWO WINDOWS	None	Creates two windows; equivalent to SPLIT WINDOW 2 command.
UPPERCASE WORD	None	Changes the current word, select range, or found range to uppercase.
WILDCARD FIND	None	Searches for a pattern of text, using either VMS or ULTRIX wildcards, depending on your setting.
WRITE FILE	None	Writes the contents of the current buffer to the file associated with the buffer or to the file you specify on the command line without ending the editing session. If the current buffer does not have a file specification associated with it, EVE prompts you for an output file specification.

Chapter 7

Editing Text Files: Using EDT

EDT is an interactive text editor. With EDT you can create a new file, insert text into it, and modify that text. You can also edit text in existing files.

EDT provides both line and keypad editing. In line editing, you type the editing command and the range of text you want the command to affect. In keypad editing, you move the cursor directly to the text you want to change and press keypad keys to enter the editing commands.

EDT provides many predefined keys that let you enter commands quickly, as described in Section 7.2. In addition, you can define your own keys for EDT, as described in Section 7.7.

7.1 Invoking and Ending an EDT Session

An editing session begins when you invoke EDT with the DCL command EDIT. In an editing session, you can create and edit a new file, or you can edit an existing file. The session ends when you enter the EXIT or QUIT command.

7.1.1 Invoking EDT

To invoke EDT, type the DCL command EDIT and specify as a parameter the file you want to edit. If the specified file already exists, EDT saves the existing versions and places a copy of the latest version in your buffer. (A buffer is the temporary storage area in which you edit text.) The existing versions of the file remain unchanged. For example, to edit an existing file named MEMO.TXT, enter the following command line:

```
$ EDIT MEMO.TXT
```

```
Once the weather turns cold, mice may find a crack in your  
foundation and enter your house. They're looking for food and  
shelter from the harsh weather ahead.
```

```
[EOB]
```

The first few lines of the latest version of the file appear on the screen. The cursor is positioned at the top of the screen, and EDT is ready to receive a keypad-editing command.

7-2 Editing Text Files: Using EDT

If you invoke EDT to create a file, the following message appears:

```
$ EDIT NEWFILE.TXT
```

```
[EOB]
```

```
Input file does not exist
```

Only the EDT message and the end-of-buffer symbol, [EOB], appear on the screen, and EDT is ready to receive keypad-editing commands. See Section 7.2.1 for a description of EDT line commands.

NOTE: In the previous examples, you enter EDT in keypad (change) mode because a startup command file (SYS\$LOGIN:EDTINI.EDT) containing the SET MODE CHANGE command has been executed. If this command is not executed in an EDT startup command file, you will enter EDT in line mode.

7.1.2 Ending an EDT Session

To terminate an EDT session, press CTRL/Z. This puts you into line-editing mode. You can type EXIT or QUIT at the asterisk (*) prompt. QUIT terminates the editing session and does not save your edits. EXIT saves your edits in a new version of the file. (Note that the existing versions of a file remain unchanged regardless of how the editing session is terminated.)

To save your edited text, use the line-editing command EXIT to terminate EDT. When you enter the EXIT command, EDT creates an output file containing the edited version of the input file. By default, the output file has the same name and type as the input file, with the version number incremented by 1.

For example, if you enter the EXIT command after editing a file named MEMO.TXT;3, EDT creates a higher version named MEMO.TXT;4 as follows:

```
*EXIT
DISK1:[USER]MEMO.TXT;4  2 lines
$
```

To override the default output file name, enter the EXIT command with a new file specification as the parameter. For example, if you end the same editing session with EXIT MICE.TXT, EDT names the output file MICE.TXT;1, provided no other file named MICE.TXT exists.

```
*EXIT MICE.TXT
DISK1:[USER]MICE.TXT;1  2 lines
$
```

To terminate EDT without saving your edits, use the line-editing command QUIT. All edits you have made to the text are ignored, and no output file is created.

```
*QUIT
$
```


The QUIT command is a useful way to terminate EDT when you have opened a file by mistake. No new file version is created.

7.2 Entering EDT Commands

Enter most keypad-editing commands by pressing a keypad key. Enter line-editing commands by typing them after the line-editing prompt and pressing RETURN.

7.2.1 Entering EDT Line Commands

EDT prompts for line-editing commands with an asterisk. Line-editing commands usually operate on a range of one or more lines of text that you specify as a parameter for the command. For example, to display an entire file on your screen, enter the TYPE command and specify WHOLE as the parameter as follows:

```
*TYPE WHOLE
```

You can abbreviate EDT line-editing commands. For clarity, the examples in this chapter show complete line-editing commands.

7.2.2 Entering Keypad Commands

In keypad editing, the screen displays editing changes as you make them. You type text from the main keyboard and enter keypad-editing commands from the numeric keypad. (To initiate keypad editing, you must first enter the line-editing command CHANGE or have SET MODE CHANGE in your EDT startup file. See Section 7.4.2 for information on the CHANGE command.)

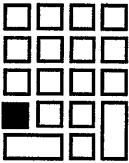
(See the description of EDT line-editing commands in the Reference Section for more information about keypad editing keys.)

Each key in the keypad performs at least one editing command; most perform two. Pressing a key invokes the regular, or upper, function. To invoke the alternate, or lower, function of a key, press the GOLD key (labeled PF1) first, followed by the desired key. In the examples that follow, a small diagram of the keypad highlights the key or keys that perform the command being described. The text associated with the keypad illustrates the effect of that editing command.

For example, keypad key 1 performs both the WORD and the CHNGCASE functions. To invoke the WORD command, press WORD: the cursor moves to the beginning of the next word.

7-4 Editing Text Files: Using EDT

WORD



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

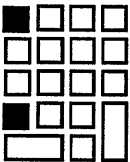
[EOB]

To invoke the **CHNGCASE** command, press the **GOLD** key first and then **CHNGCASE**. The character at the cursor or the characters highlighted with the select key changes from lowercase to uppercase or from uppercase to lowercase.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

CHNGCASE



Once The weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

The supplemental editing keys on the VT200 keypad perform the same functions as some of the EDT keypad keys. (See the description of EDT line-editing commands in the Reference Section for more information about these supplemental editing keys.)

7.2.3 Canceling EDT Commands

Use CTRL/C to cancel the currently executing EDT command without affecting previous edits. For example, to stop the display of a long file, press CTRL/C.

```
*TYPE WHOLE
```

```
.
```

```
.
```

```
CTRL/C
```

```
CANCEL
```

```
Aborted by CTRL/C
```

```
*
```

The display stops and the CTRL/C message appears.

7.3 Getting HELP in EDT

EDT provides a help facility for each of the EDT editing modes.

7.3.1 Getting HELP with Keypad-Editing Commands

To display a diagram of the keypad keys and their functions, enter change mode (assuming you are in line-editing mode) and then press the HELP key (labeled PF2). (On VT200-series terminals, you can also use the HELP key on the supplemental editing keypad.) To display information about a particular keypad command, first press the HELP key and then press the keypad key.

7.3.2 Getting HELP with Line-Editing Commands

To display a list of EDT topics on which information is available, type HELP and press RETURN. To display information about a particular command or topic, type HELP followed by the name of the topic and press RETURN. EDT responds with a display of information about the topic and a list of related topics about which information is available. To display information about the use of a particular command qualifier, type HELP plus the command and that qualifier and press RETURN. For example, to display information on the use of /QUERY with the COPY command, enter the following command line:

```
*HELP COPY /QUERY
```

7.4 Changing Editing Modes

You can easily switch back and forth between line and keypad editing; you can also enter line-editing commands from keypad mode. Before using keypad commands, be sure that your terminal type is set properly. (Use SHOW TERMINAL to display the setting and SET TERMINAL/INQUIRE to set the terminal type.)

7.4.1 Changing from Keypad to Line Editing

To change from keypad editing to line editing, press CTRL/Z. The asterisk prompt appears at the bottom of your screen, indicating EDT is ready to accept line-editing commands.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]
CTRL/Z

*

7.4.2 Changing from Line to Keypad Editing

To change from line editing to keypad editing, enter the CHANGE command:

*CHANGE

The first 22 lines of the file are displayed on your screen. If the file has fewer than 22 lines, the [EOB] symbol appears below the last line of the file.

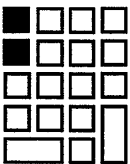
7.4.3 Entering Line-Editing Commands from Keypad Mode

The keypad COMMAND function allows you to enter line-editing commands without leaving keypad mode. First, enter COMMAND (by pressing GOLD and then COMMAND) to invoke the *Command:* prompt, then type the line-editing command and press ENTER. (If you press RETURN by mistake, ^M appears; delete the ^M by pressing the DELETE key on the main keyboard, and press ENTER.) The following example enters the line-editing command SET QUIET, which suppresses the sound made when EDT issues an error message:

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

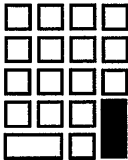
COMMAND



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]
Command: SET QUIET

ENTER



7.5 Recovering from Interruptions

You can recover from interruptions to your editing session in the following ways:

- **Deleting extraneous characters**—Pressing CTRL/W removes extraneous characters (such as a broadcast message or a message indicating that you have received electronic mail) from the screen and restores the previous display. Use CTRL/W to ensure that the cursor is in the correct position.
- **Resuming an interrupted editing session**—The DCL command CONTINUE resumes an editing session that was interrupted by pressing CTRL/Y, so long as only built-in DCL commands were entered after pressing CTRL/Y. For example, you could press CTRL/Y, enter the command SHOW TIME, and return to your editing session with the CONTINUE command.

(Press CTRL/W to refresh the screen display. The text of your editing session is once again displayed.)

- **Recovering a lost session**—By default, EDT keeps a journal file with the same file name as the input file and a file type of JOURNAL. If the editing session ends without interruption, the journal file is deleted when you terminate the session. If the editing session is aborted (for example, during a system failure, in response to pressing CTRL/Y, or entering the QUIT/SAVE command), you can recover your edits (with the exception of those commands entered just prior to the interruption). Enter the same command line you used to begin the editing session, adding the /RECOVER qualifier. For example:

```
§ EDIT/RECOVER MEMO.TXT
```

EDT will reproduce the editing session, reading the commands from the journal file and executing them on the screen.

7.6 EDT Keypad Editing

While line editing allows you to manipulate large portions of text easily, keypad editing provides easy manipulation of small units of text. Several EDT keypad commands enable you to find, insert, delete, substitute, and move text in a file. The cursor can be moved through a file in a variety of ways, and the position of the cursor in a file determines how text will be affected by EDT commands.

7.6.1 Manipulating the Cursor

You can manipulate the cursor with commands that move it unit by unit through the text or with commands that move it directly to a particular location. Several commands that move the cursor are controlled by the ADVANCE and BACKUP commands, which set the cursor's direction forward and backward. Unless otherwise stated, this chapter assumes the default direction of the cursor to be ADVANCE.

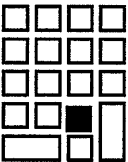
You can move the cursor by character, word, and line units. Use one of the following keys to move the cursor by character:

- **RIGHT ARROW** — Moves the cursor one character to the right.
- **LEFT ARROW** — Moves the cursor one character to the left.
- **CHAR** — Moves the cursor one character in the current direction.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house. They're looking for food and
shelter from the harsh weather ahead.
```

[EOB]

CHAR



```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house. They're looking for food and
shelter from the harsh weather ahead.
```

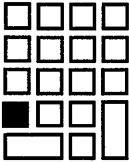
[EOB]

The **WORD** command moves the cursor to the beginning of the next or previous word.

```
Once the weather turns cold, mice may find a crack in your
foundation and enter your house. They're looking for food and
shelter from the harsh weather ahead.
```

[EOB]

WORD



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

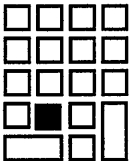
The following keys move the cursor by line:

- UP ARROW—Moves the cursor up one line.
- DOWN ARROW—Moves the cursor down one line.
- EOL—Moves the cursor to the end of the current or previous line.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

EOL



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

- F12 (the BACKSPACE key on VT100-series terminals)—Moves the cursor to the beginning of the previous line.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. **F12** (**BACKSPACE**)

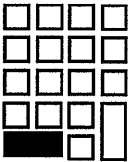
Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

7-10 Editing Text Files: Using EDT

- **LINE**—Moves the cursor to the beginning of the next line or previous line.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

LINE

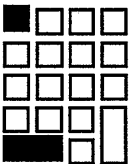


Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

The **OPEN LINE** command terminates a line without moving the cursor. (The **RETURN** key also terminates a line, but moves the cursor to the next line.) The **OPEN LINE** command is useful when you want to insert a blank line or a new line of text. When the cursor is placed at the beginning of a line and the **OPEN LINE** command is entered, the text on that line is moved down so that the cursor is at the beginning of a blank line as follows:

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

OPEN LINE



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

To move the cursor by large units, use the **SECT** and **PAGE** commands. The **SECT** and **PAGE** commands allow you to scan several lines of text at a time. The direction in which EDT moves depends upon whether **ADVANCE** or **BACKUP** is set.

- **SECT**—Moves the cursor across a 16-line section of text in EDT's current direction. If there are fewer than 16 lines, SECT moves the cursor across the existing lines.

(On the VT200-series terminals, the supplemental editing keypad key Next Screen moves the cursor 16 lines forward, regardless of EDT's current direction. The supplemental editing keypad key Prev Screen moves the cursor 16 lines backward, regardless of EDT's current direction.)

- **PAGE**—Moves the cursor to the next or previous page boundary (form feed) or to the end or top of the buffer if there is no boundary. To insert form feeds in your text, use CTRL/L.

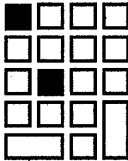
The TOP and BOTTOM commands allow you to move directly to the beginning or end of a buffer. (See Section 7.6.8 for more information about buffers.)

- **TOP**—Moves the cursor to the beginning, or top, of the buffer.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

TOP



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

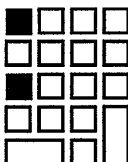
[EOB]

- **BOTTOM**—Moves the cursor to the end, or bottom, of the buffer.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

BOTTOM



7-12 Editing Text Files: Using EDT

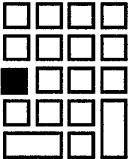
Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

The **ADVANCE** and **BACKUP** commands control the cursor's direction for the following EDT keypad-editing commands: **CHAR**, **CHNGCASE**, **EOL**, **FIND**, **FNDNXT**, **LINE**, **PAGE**, **SECT**, **SUBS**, and **WORD**. Each of the directional commands remains in effect until you set the cursor in the opposite direction with the other command.

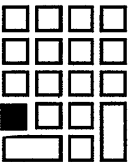
- **ADVANCE**—Sets the cursor's direction forward so that subsequent commands move the cursor in the forward direction. For example, if you enter the **WORD** command after using **ADVANCE**, the cursor moves forward one word.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

ADVANCE



WORD

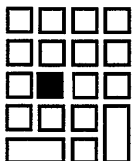


Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

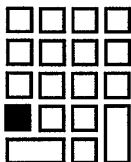
- **BACKUP**—Sets the cursor's direction in the backward direction so that subsequent commands move the cursor toward the top of the buffer. For example, if you enter the **WORD** command after using **BACKUP**, the cursor moves backward one word.

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

BACKUP



WORD



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

The cursor remains set in the backward direction until you press ADVANCE. For example, if you enter a second WORD command in the preceding example you receive a message indicating that the command requests EDT to back up past the top of the buffer.

The ADVANCE and BACKUP commands are particularly important in string searches; see Section 7.6.4 for more information on searches.

7.6.2 Inserting Text

To insert text in EDT keypad editing, position the cursor where you want the text to be inserted and begin typing; the cursor remains one position to the right of the last character inserted. Inserting text in the middle of a line moves both the cursor and the remainder of the line one position to the right for each character inserted. When the line exceeds 80 characters, the text you type will either wrap to the following line or disappear off your screen, depending on the status of the SET SCREEN, SET [NO]TRUNCATE, and SET [NO]WRAP commands. (See Section 7.8.1 for information about screen formatting commands.)

7.6.3 Deleting and Restoring Text

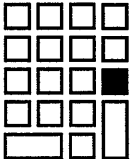
The delete commands work like the cursor movement commands. In EDT keypad editing, you can delete by character using the Delete key (<X) (DELETE on VT100-series terminals) and DEL C; by word using F13 (LINEFEED on VT100-series terminals) and DEL W; and by line using DEL L, DEL EOL, and CTRL/U.

The deleted text is stored in a buffer so that you can also restore the character (UND C), word (UND W), or line (UND L) most recently deleted wherever and as many times as you need. Note that the undelete commands restore only the corresponding units of text that were most recently deleted. For example, if you have deleted two lines of text with the DEL L (delete line) command, the UND L (undelete line) command will restore only one line, the line most recently deleted.

The <X> key on the main keyboard (the DELETE key on VT100-series terminals) deletes the character immediately to the left of the cursor. The EDT keypad-editing command DEL C deletes the character directly at the cursor. The UND C command restores the last character deleted with either the <X> (DELETE) key or the DEL C command. For example:

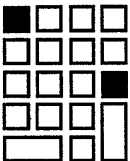
Once the weather turns cold, mice may find a crack in your
 foundation and enter your house. They're looking for food and
 shelter from the harsh weather ahead.
 [EOB]

DEL C



nce the weather turns cold, mice may find a crack in your
 foundation and enter your house. They're looking for food and
 shelter from the harsh weather ahead.
 [EOB]

UND C

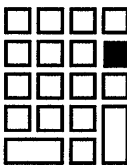


Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

The F13 key on the main keyboard (the LINEFEED key on VT100-series terminals) deletes to the beginning of the current or preceding word. The DEL W command deletes to the end of the current word. Blank spaces are considered part of the word they follow, while all other word delimiters are considered to be separate words. The UND W command restores the last word deleted with either the F13 (LINEFEED) key or the DEL W command. For example:

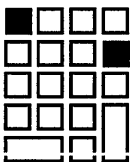
Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

DEL W



Once the weather turns cold, may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

UND W



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.
[EOB]

The following commands delete a line (or part of a line) of text:

- **DEL L**—Deletes from the cursor to the end of the line, including the line terminator. If the cursor is at the beginning of the line, the entire line is deleted, and the cursor is positioned at the beginning of the next line.

7-16 Editing Text Files: Using EDT

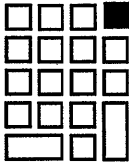
- **DEL EOL**—Deletes from the cursor to the end of the line (excluding the line terminator), leaving the cursor at the end of the truncated line.
- **CTRL/U**—Deletes from the cursor to the next previous beginning of line, leaving the cursor at the beginning of the previous line. (If CTRL/U is used when the cursor is at the beginning of the line, the previous line is deleted.)

The **UND L** command restores the last line (or part of a line) that was deleted with the **DEL L**, **DEL EOL**, or **CTRL/U** command. For example:

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

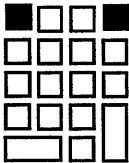
DEL L



Once the weather foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

UND L



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead.

[EOB]

The EDT line-editing command **DELETE** is useful for deleting large sections of text. Generally, you use line numbers to specify a range for a line-editing command. For example, to delete lines 306 through 860, enter the following:

```
*DELETE 306 THRU 860
```

```
555 lines deleted
```

```
861 Rodents have had a profound effect on human civilization.
```

*

Note that the EDT line-editing command SET NUMBERS (the default) must be in effect for line numbers to be displayed in EDT line editing.

You can also use certain keywords (such as WHOLE, REST, BEFORE) as range specifiers. For example, if you are in the middle of a long buffer and want to delete from the cursor to the end of the buffer, enter the following:

```
*DELETE REST
43 lines deleted
[EOB]
*
```

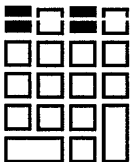
(You can also specify range by using the EDT keypad-editing command SELECT. See Section 7.6.7 for information on SELECT.)

7.6.4 Locating Text

You can move the cursor to a character string you specify with the FIND and FNDNXT EDT keypad-editing commands. The FIND command searches for the specified character string between the current position of the cursor and the beginning or end of the buffer (depending on whether the ADVANCE or the BACKUP command is in control). EDT does not distinguish between uppercase and lowercase letters unless you use the SET SEARCH EXACT line-editing command. When EDT finds the string, it positions the cursor at the first character in the string (unless the SET SEARCH END command is in effect, and the cursor is positioned at the last character in the string). In a long file, the message "Working" may flash on the screen while EDT searches for the string.

For example, to delete a comma after the word "house" in the following text, you can use the FIND command to move the cursor to the string "house." First, enter the EDT keypad command FIND by pressing the GOLD key and then the FIND key (on the VT200-series terminal you can also use the FIND key located on the supplemental editing keypad). Next, type the string you want to locate (the search string) after the Search for: prompt.

FIND



7-18 Editing Text Files: Using EDT

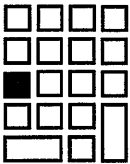
Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. Because mice reproduce so quickly, what started as one or two mice can quickly become an invasion. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.

[EOB]

Search for: house

To search in the forward direction, use the **ADVANCE** command to enter the search string.

ADVANCE

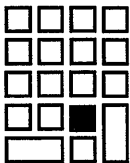


Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. Because mice reproduce so quickly, what started as one or two mice can quickly become an invasion. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.

[EOB]

Use the **CHAR** command to move the cursor to the comma after the word "house." Then use the **DEL C** command to delete the comma.

CHAR



Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. Because mice reproduce so quickly, what started as one or two mice can quickly become an invasion. If you seal the cracks and holes on the exterior of your house you can prevent these rodents from ever getting in.

[EOB]

To find the next occurrence of the string located with the FIND command, use the FNDNXT (find next) command. If there is no other occurrence of the string (as in the example above), EDT issues the message "String was not found."

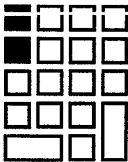
NOTE: The directional setting of the cursor determines the direction of the search. After you press FIND, you can press either ADVANCE or BACKUP (depending on the direction in which you want to search) to enter the search string. You can also use the ENTER command, which applies the current direction to the search.

7.6.5 Substituting Text

To substitute one character string for another, you can use the SUBS keypad-editing command or the SUBSTITUTE line-editing command. The EDT line-editing command can make global substitutions; that is, it can replace every occurrence of one character string in the specified range with another string using only one EDT line-editing command. In contrast, you must use the keypad SUBS command (press the GOLD key followed by the SUBS key) for each substitution you make. (If you do not specify a range, the line-editing command SUBSTITUTE replaces only the first occurrence of the search string in the current line with the substitute string.)

For example, to substitute the string "mice" for "elephants" throughout a buffer, enter the line-editing command SUBSTITUTE, the old string, and the new string, separating all three with the same delimiter. You can use any nonalphanumeric character (except the percent sign and underscore) as a delimiter for the SUBSTITUTE command, as long as the delimiting character is not part of either string. To apply the command to the entire buffer in a global substitution, specify WHOLE as the parameter. When the operation has been completed, EDT displays each occurrence of the substitution and the total number of substitutions. The following example substitutes the string "mice" for each occurrence of the string "elephants" in the following text:

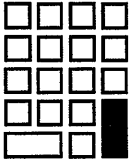
COMMAND



7-20 Editing Text Files: Using EDT

Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.
[EOB]

ENTER



1 Once the weather turns cold, elephants may find a crack
4 in your electrical wires and raid your food. Because elephants reproduce
5 so quickly, what started as one or two elephants can quickly become an
3 substitutions
Press return to continue

Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.
[EOB]

Note that a global substitution replaces all occurrences of the string, regardless of case or surrounding characters. If you want EDT to search for exact comparisons of case, use the SET SEARCH EXACT command. If the search string occurs in the middle of a longer string, the substitution will still be made. For instance, a global substitution of "IN" for "AT" would change all words containing the string "AT." ("LATER" would become "LINER", "THAT" would become "THIN", "SAT" would become "SIN", and so on.)

To get EDT to prompt you before each substitution, use the /QUERY qualifier with the SUBSTITUTE command.

Command: SUBSTITUTE\AT\IN\WHOLE/QUERY

EDT prompts you with a question mark (?) to verify each substitution. You can respond with one of the following:

- Y Yes, do the substitution.
- N No, do not do the substitution.

- Q Quit, terminate the command.
 A All, do the rest of the substitutions without query.

7.6.6 Moving Text

Both EDT keypad and line commands can move text; however, only line-editing commands transfer text between buffers and files.

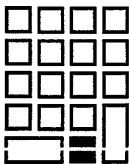
7.6.7 Moving Text Within the File

The EDT keypad-editing command CUT deletes a selected range of text and the PASTE command inserts it at the cursor's current position. (On the VT200-series terminals, the supplemental editing keys Remove and Insert Here perform the same functions as the EDT keypad commands CUT and PASTE.) For instance, to move the first sentence in the second paragraph of the example to the end of that paragraph, move the cursor to the beginning of the sentence and press SELECT. (On the VT200-series terminals, the supplemental editing key SELECT performs the same function as the EDT keypad command SELECT.) This marks the beginning of the selected range. (You can cancel the SELECT command with the RESET command.)

Once the weather turns cold, mice may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. Because mice reproduce so quickly, what started as one or two mice can quickly become an invasion. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.

[EOB]

SELECT



To mark the end of the selected range, move the cursor to the end of the sentence. The terminal highlights a selected range in reverse video. (The selected range includes the text up to the character preceding the cursor.)

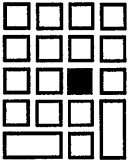
Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.

[EOB]

7-22 Editing Text Files: Using EDT

Press CUT to delete the selected text.

CUT



Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.

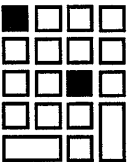
[EOB]

Deleted text remains in the PASTE buffer until you perform another CUT operation. To restore the text, move the cursor to the appropriate position and enter the PASTE command. (The text will be inserted directly in front of the cursor.)

Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in.

[EOB]

PASTE



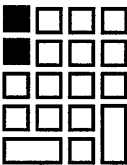
Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.

[EOB]

Because the selected text is held in the PASTE buffer until you perform another CUT operation (or give the line-editing command CLEAR PASTE), you can paste the text contained in the PASTE buffer as many times as you want. You can also enter the PASTE buffer to edit the text it contains. (See Section 7.6.8 for information on using multiple buffers.)

After moving the text, you may want to use the FILL command to reorganize selected text so that the maximum number of whole words are fitted within the current line width. The default line width is 80 characters, but you can use the SET WRAP command to use another line length for filling text. For example, you can set the line length to 71 characters with the EDT line-editing command SET WRAP and then fill a selected range of text.

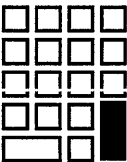
COMMAND



Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.

[EOB]
Command: SET WRAP 71

ENTER

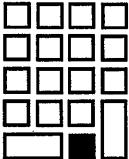


Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.

[EOB]

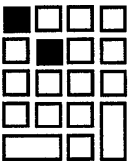
EDT will now wrap lines of inserted text and fill lines of selected text at a line width of 71 characters. Use the **SELECT** command to mark the text you want to affect and then enter the EDT keypad command **FILL**.

SELECT



Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.
[EOB]

FILL



Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.
[EOB]

There are several EDT line-editing commands that move text. For example, the **MOVE** and **COPY** commands each perform a function similar to those of the keypad **CUT** and **PASTE** operations. **MOVE** deletes text from one location and inserts it in another; **COPY** inserts a copy of the text where specified without deleting any text. The EDT line-editing commands **INCLUDE** and **WRITE** perform tasks not possible with EDT keypad-editing commands:

- **INCLUDE**—Copies a file into the buffer you are currently editing or the buffer you specify. Follow the VMS conventions for file specifications when

specifying the file to be copied to the buffer. For example, the following command copies the file named MEM.DAT to the buffer named BUF1:

Command: INCLUDE MEM.DAT =BUF1

- **WRITE**—Copies a specified range of text from a buffer (the current buffer by default) to a specified file. If you do not specify a range, the WRITE command copies the entire contents of the current buffer. For example, the following command copies the contents of the current buffer to the file ANIMALS.TXT:

Command: WRITE ANIMALS.TXT
 \$DISK1:[USER]ANIMALS.TXT;1 11 lines

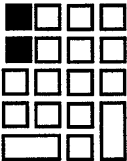
The message displays the new file specification and length.

7.6.8 Using Multiple Buffers

When you begin editing a file with EDT, you are working on a copy of the file in a buffer called MAIN. (EDT also uses a buffer called PASTE to store the text that you delete with the CUT and APPEND commands; you can edit this buffer just as you can edit other text buffers.) You can create other buffers to store pieces of text during your EDT editing session. You can enter and edit these buffers; you can copy text to and from them; and you can write their contents to specified files.

To create a buffer, press the COMMAND key. Type the line-editing command FIND followed by the equal sign and the name you are giving the buffer, then press the ENTER key. For example, the following command creates a buffer named BUF1:

COMMAND



Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.

[EOB]

Command: FIND=BUF1

7-26 Editing Text Files: Using EDT

When you enter this command, the system responds by displaying only the [EOB] symbol, which indicates that the current buffer, BUF1, is empty. You can now insert and edit text just as you would in the MAIN buffer. To return to the MAIN buffer, follow the same procedure, typing FIND=MAIN rather than FIND=BUF1. To return to your previous position in the MAIN buffer, include a period after the buffer's name as follows:

Command: FIND=MAIN.

The buffer named BUF1 remains intact until you exit from EDT, regardless of whether you enter the EXIT or QUIT command. That is, you can enter, edit, and exit from a buffer as necessary. However, when you exit from EDT, only the buffer MAIN is saved.

The SHOW BUFFER command displays the number of lines contained in each buffer and indicates (with an equal sign) the current buffer. The following example indicates that there are three buffers (including MAIN and PASTE, which always exist) and that MAIN is the current buffer:

COMMAND

```
■ □ □ □
■ □ □ □
□ □ □ □
□ □ □ □
□ □ □ □
□ □ □ □
```

Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.

[EOB]

Command: SHOW BUFFER

=MAIN 11 lines

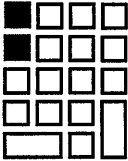
PASTE 3 lines

BUF1 2 lines

Press return to continue

Pressing the RETURN key returns the cursor to its previous position in the buffer.

You can further manipulate the contents of a buffer by specifying the buffer's name in an EDT line-editing command. For example, if you are in the MAIN buffer and want to save the contents of BUF1 in a file named RODENT.TXT before exiting from EDT, enter the following command:

COMMAND

Once the weather turns cold, elephants may find a crack in your foundation and enter your house. They're looking for food and shelter from the harsh weather ahead. Once inside, they may gnaw through electrical wires and raid your food. If you seal the cracks and holes on the exterior of your house, you can prevent these rodents from ever getting in. Because elephants reproduce so quickly, what started as one or two elephants can quickly become an invasion.

[EOB]

Command: WRITE RODENT.TXT =BUF1

\$DISK1:[USER]RODENT.TXT;1 2 lines

EDT returns a message indicating that the file has been created, and the cursor is returned to its previous location in the buffer.

7.7 Saving Time and Keystrokes—Defining Keys in EDT

If you have a series of keystrokes that you repeat frequently, then you can save time and keystrokes by using a feature of EDT that lets you associate a set of keystrokes with a particular key sequence. With this feature, you can define a key to output a string of text, to execute a series of EDT functions, or to combine one or more text strings with one or more EDT functions.

The easiest way to define keys is while you are already in the EDT editor. However, any key definitions that you make during an EDT editing session will be canceled when you exit from EDT. You can also make permanent key definitions that will apply whenever you use the EDT editor. This section describes how to define keys in EDT.

7.7.1 Defining Keys While in EDT

To define a key while you are already in an EDT editing session, you always follow the same general process:

- ❶ Signal EDT that you want to define a key by pressing `[CTRL]K`.
- ❷ Select the key (either a CTRL key or GOLD- sequence) that you want to define.

- ③ Begin the key definition with an open parenthesis.
- ④ Describe the text that you want to insert and/or the EDT functions that you want to execute.
- ⑤ End the key definition by typing a closed parenthesis and a period, and then pressing **ENTER**.

When you define a key while already in the EDT editor, the key definition will end when you exit from EDT. (See Section 7.7.3 for information about making key definitions that apply every time you use the EDT editor.)

Defining a Key to Insert Text

For example, suppose that you often type the words *International Development Organization*. To include this expression in your text simply by pressing **CTRL/A**, use the following steps:

1. Press **CTRL/K**. This tells EDT that you want to define a key.

When you press **CTRL/K**, the message "Press the key you wish to define" is displayed on your terminal.

2. Press **CTRL/A**. This tells EDT that it is the **CTRL/A** key that you want to define.

When you press **CTRL/A**, the message "Now enter the definition terminated by **ENTER**" is displayed on your terminal.

3. Enter the following text, exactly as it is shown below:

(iInternational Development Organization**CTRL/Z**).

(Note that when you press **CTRL/Z** in the context of defining a key, the symbol **^Z** is displayed).

4. Press **ENTER**.

Now press **CTRL/A**; the words *International Development Organization* are inserted in your text.

When you entered the definition of the key, the entire expression was enclosed in parentheses. After the first (open) parenthesis, the first character in the key definition was the letter *i*. This signifies that everything following the letter *i* will be text, until **CTRL/Z** is pressed. When you pressed **CTRL/Z**, it signified the end of

the text. The closed parenthesis and the final period signified the end of the key definition; the key definition is terminated when you pressed `[ENTER]`.

You can also define a key that includes more than one line of text, simply by using the `[RETURN]` key in your definition. For example, suppose that you were using text formatting software in which you type the following text from time to time:

```
<list>(unnumbered)
<le>
```

You could define the `[CTRL/E]` key to do this for you, as follows:

1. Press `[CTRL/K]`
2. Press `[CTRL/E]`
3. Type the following, exactly as shown:

```
(i<list>(unnumbered) [RETURN]<le> [CTRL/Z] . [ENTER]
```

As in the previous example, you type the letter *i* to signify the beginning of the text that is to be included in the key definition, and that you press `[CTRL/Z]` to signify the end of the text. Also note that when you press `[RETURN]` in the context of defining a key, the symbol `^M` is displayed on the terminal.

4. Press `[CTRL/E]` and see that the text is included in your file.

Defining a Key to Use EDT Functions

You can also define a key that performs one or more consecutive EDT functions. For example, suppose that you are editing the following text file, which has four columns of data. In this example, you want to eliminate the last two columns ("Price" and "Total") in each row:

Item	Quantity	Price	Total
Apples	20	1.00	20.00
Bananas	40	1.50	60.00
Beets	25	2.00	50.00
Carrots	30	2.00	60.00
Oranges	20	4.00	80.00
Peaches	10	3.00	30.00
Pears	5	6.00	30.00
Potatoes	50	1.00	50.00_

[EOB]

You could move the cursor to the first line ("Item"), press `[F1]` twice to move the cursor two words (to "Price"), press `[GOLD-KF2]` to delete the text from the cursor to the end of the line, press `[F0]` to move the cursor to the beginning of the next line, and then start the process all over again. Alternatively, you could define a key to do most of the work for you.

7-30 Editing Text Files: Using EDT

To define the `[CTRL/D]` key to do the work, use the following steps:

1. Press `[CTRL/K]`. This tells EDT that you want to define a key.

When you press `[CTRL/K]`, the message "Press the key you wish to define" is displayed on your terminal.

2. Press `[CTRL/D]`. This tells EDT that it is the `[CTRL/D]` key that you want to define.

When you press `[CTRL/D]`, the message "Now enter the definition terminated by ENTER" is displayed on your terminal.

3. Type these keys, exactly in the order shown:

1. (
2. `[KP1]`
3. `[KP1]`
4. `[GOLD-KP2]`
5. `[KP0]`
6. `).``[ENTER]`

The bottom of the screen will now look like this:

Item	Quantity	Price	Total
Apples	20	1.00	20.00
Bananas	40	1.50	60.00
Beets	25	2.00	50.00
Carrots	30	2.00	60.00
Oranges	20	4.00	80.00
Peaches	10	3.00	30.00
Pears	5	6.00	30.00
Potatoes	50	1.00	50.00

[EOB]

Now enter the definition terminated by ENTER
(GOLD+ELL)

This representational text is associated with the EDT functions that you select for the key definition. Notice that the key definition begins with an open parenthesis, and it ends with a closed parenthesis followed by a period. When you press `[ENTER]` to complete the key definition, the text is removed from the screen.

When you are in the EDT editor and define a key to be a series of EDT functions, you can either press the function keys or actually type the representational text (in this case, `WWD+ELL`).

Keys Available for Definitions

The only keys that can be defined in EDT are control keys and two-key sequences beginning with the GOLD (PF1) key.

7.7.2 Advanced Key Definitions

In addition to straightforward text insertions and series of functions, you can combine functions with text insertions. You use the same principles as when you define keys for text insertion only or functions only:

- Always begin your key definition with an open parenthesis.
- When you want to include text in your key definition, signal the beginning of the text by typing *i*, and signal the end of the text by pressing **[CTRLZ]**.
- When you want to include EDT functions in your definition, simply use the function keys from the keypad.
- End the key definition with a closed parenthesis and a period, and then press **[ENTER]**.

For example, suppose that you are editing the following command procedure:

```
$ COPY [MONTHLY]REGION-1.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-2.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-3.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-4.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-5.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-6.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-7.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-8.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-9.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-10.DAT [FINANCE]*.*
[EOB]
```

Due to a change on your system, it is now necessary to modify the command procedure to specify the devices on which the files reside. For each filespec, the directory **[MONTHLY]** must be preceded by the device name **DISK1:**, and the directory **[FINANCE]** must be preceded by the device name **DISK2:**.

To expedite the editing process, you could define a key in EDT that edits each line automatically. The key that you define would move the cursor to **[MONTHLY]**, insert the text **DISK1:**, move the cursor to **[FINANCE]**, insert the text **DISK2:**, and finally move the cursor to the beginning of the next line.

To define the key sequence **[GOLD-F]** to replicate this series of keystrokes, do the following:

1. Press **[CTRLK]** to signal a key definition.
2. Press **[GOLD-F]** to signify the key that you are defining.
3. Type an open parenthesis to begin the key definition.

7-32 Editing Text Files: Using EDT

4. Press **[KP1]** two times, to represent the cursor moving ahead two words.

5. Type the following, to insert the appropriate text:

```
iDISK1: [CTRL/Z]
```

6. Press **[KP1]** once, to represent the cursor moving ahead one word.

7. Type the following, to insert the appropriate text:

```
iDISK2: [CTRL/Z]
```

8. Press **[KP0]** to move the cursor to the beginning of the next line.

9. Type a closed parenthesis followed by a period. The bottom line of your terminal now looks like this:

```
$ COPY [MONTHLY]REGION-1.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-2.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-3.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-4.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-5.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-6.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-7.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-8.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-9.DAT [FINANCE]*.*
$ COPY [MONTHLY]REGION-10.DAT [FINANCE]*.*
[EOB]
```

```
Now enter the definition terminated by ENTER
[[[WIDISHI: SWIRISHI: DLV ]
```

10. End the key definition by pressing **[ENTER]**.

With the cursor at the beginning of the first line of your command procedure, press **[GOLD-F]**. The edits are automatically inserted on the first line, and the cursor is at the beginning of the second line.

You could complete the edits by pressing **[GOLD-F]** 9 more times, or you could further expedite the process by using the REPEAT function in EDT, with the following sequence:

1. Press the **[GOLD]** key
2. Type *10* (from the keyboard, not the keypad)
3. Press the sequence **[GOLD-F]**

This executes the **[GOLD-F]** sequence ten times.

Using the Find Function

To use the EDT find function in a key definition, you must use quotation marks around the text for which you are searching. That is, the expression "**subroutine**" in a key definition means "search for the text string *subroutine* and move the cursor to the beginning of that text string." For example, suppose you have a source program that includes several instances of the following text, with the name of various subroutines in place of *subroutine-name*:

```
.SUBROUTINE subroutine-name
```

You now find that some additional code is needed after each subroutine, in order to resolve potential errors, and you want to add the following line after each subroutine call:

```
.GO TO ERROR-CHECKING
```

A key defined for this would first search for the text string *.SUBROUTINE* and move the cursor to the beginning of the next line. You would then use the OPEN LINE function (to make sure there is a blank line), and then insert the text *.GO TO ERROR-CHECKING*. To define the `[CTRLB]` key to do this, you would use the following steps:

1. Press `[CTRLK]`
2. Press `[CTRLB]`
3. Type an open parenthesis to begin the key definition
4. Type the following to search for the string *.SUBROUTINE*:
".SUBROUTINE"
5. Press `[KPO]` to represent the cursor moving to the beginning of the next line, then press the sequence `[GOLD-KPO]` to represent the OPEN LINE function
6. Type the following to include the appropriate text:
i.GO TO ERROR-CHECKING`[CTRLZ]`

The bottom lines of your screen now look like this:

```
[EOB]
Now enter the definition terminated by ENTER
".SUBROUTINE"([M-C])i.GO TO ERROR-CHECKING"
```

7. Type a closed parenthesis and a period, then press `[ENTER]` to complete the key definition.

7.7.3 Permanent Key Definitions

The previous section explained how to define a key while in EDT. This is easy and quick to do, but any keys that you define during an editing session will apply only during that editing session. You can also define keys that will apply every time that you use the EDT editor. It is a more complex process than defining keys during an editing session, but it might save you time in the long run.

To define keys that will apply whenever you use the EDT editor, you use the following process:

- Create a file in which your keys are defined. This file is called an **EDT initialization file**, and it is a text file that can contain key definitions and other set-up information about your EDT session.
- Specify the initialization file that you will use by using the `/COMMAND=` qualifier in your `EDIT/EDT` command line. For example, if your EDT initialization file is `[THOMAS]EDTINI.EDT`, then you would use the following DCL-level command to edit a file named `REPORT.TEXT` using your initialization file:

```
$ EDIT /EDT /COMMAND=[THOMAS]EDTINI.EDT REPORT.TEXT
```

You can of course define a symbol in your login command file that would reduce the number of keystrokes that you need; for example:

```
$ EDT ::= EDIT /EDT /COMMAND=[THOMAS]EDTINI.EDT
```

The EDT Initialization File

The EDT initialization file is a text file that you can create to define keys and set-up parameters for your EDT editing session. Set-up parameters include any `SET` command that EDT allows, as described in the EDT reference documentation.

Defining Keys in the EDT Initialization File

To define a key in an EDT initialization file, use the following format:

```
DEFINE KEY key-name AS 'key-definition.'
```

The *key-name* can be either a CTRL key or a sequence using the `[GOLD]` key. For CTRL keys, use the word *CONTROL* followed by a single space and the CTRL character; for `[GOLD]` sequences, use the word *GOLD* followed by a single space and the appropriate character. For example, the following lines would begin key definitions for the `[CTRL/A]`, `[CTRL/B]`, `[GOLD-L]`, and `[GOLD-3]` keys:

```
DEFINE KEY CONTROL A AS '.....'
DEFINE KEY CONTROL B AS '.....'
DEFINE KEY GOLD L AS '.....'
DEFINE KEY GOLD 3 AS '.....'
```

Each key definition in the EDT initialization file must be on a single line, and the line can include up to 255 characters.

The next section describes the syntax that you use for the key definitions.

Key Definitions in the EDT Initialization File — Text

Key definitions in an EDT initialization file are similar to the key definitions made during an editing session. The letter *i* in a key definition signifies the beginning of text, and the text continues until a `[CTRLZ]` is reached.

When defining a key in the EDT editor, you could press `[CTRLZ]` and the symbol (^Z) would appear in the right place. However, if you are creating your EDT initialization file pressing `[CTRLZ]` will have a different meaning. Therefore, you have to use the SPECIAL INSERT function to insert the `[CTRLZ]` into your initialization file.

To insert a `[CTRLZ]` into your file, you type the following sequence:

```
GOLD
26 [from the keyboard]
GOLD
3 [from the keypad]
```

If you are using EDT, the symbol ^Z is displayed on your terminal.

So to define the `[CTRLA]` key to insert the text *International Development Organization*, you would include the following line in your initialization file:

```
DEFINE KEY CONTROL A AS 'iInternational Development Organization^Z.'
```

Remember that the ^Z is a symbol for `[CTRLZ]` that is inserted into your file with the following sequence:

```
GOLD
26
GOLD
3
```

Also remember to complete the key definition with a period, and then to enclose the key definition in quotation marks.

If you want to insert multiple lines of text in your key definition, then you must use the symbols for carriage-returns and line-feeds in the text that you insert. In Section 7.7.1, you learned how to define a key that would include the following text in a file:

```
<list> (unnumbered)
<le>
```

When defining a key in the EDT editor, you could insert a carriage-return simply by pressing the `[RETURN]` key. However, when you are editing your initialization file, you must insert the symbol for the `[RETURN]` key in a similar way as you inserted the symbol for the `[CTRLZ]` key. The `[RETURN]` key is considered to be text, so the symbol for it must also be preceded by *i* and followed by ^Z.

In EDT, the symbol for the `RETURN` key is represented by `<CR>`. To insert the symbol for `RETURN`, use the following sequence:

```
GOLD
13 (Keyboard)
GOLD
3 (Keypad)
```

When you do this, the symbol `<CR>` is displayed on your terminal.

So, to define key `CTRL/E` to output this text, you would use the following line in your EDT initialization file:

```
DEFINE KEY CONTROL E AS 'i<LIST>(UNNUMBERED)<CR><LE>^Z.'
```

In this example, the symbol `<CR>` is inserted using the sequence GOLD-13-GOLD-KP3, and the symbol `^Z` is inserted using the sequence GOLD-26-GOLD-KP3.

Key Definitions in the EDT Initialization File — EDT Function

To indicate EDT functions, specific symbols are used in your EDT initialization file. Table 7-1 lists these symbols; they are the same symbols displayed on your terminal when you define keys while in an EDT session.

Table 7-1: Symbols for EDT Functions

EDT Function	Key	Symbol	Explanation
FIND	PF3	" "1	Search for a string of text
DELETE LINE	PF4	D+NL	Delete Line
UNDELETE LINE	GOLD-PF4	UNDL	Insert contents of delete line buffer
SECTION	KP8	(16L)	Moves cursor one section (16 lines)
FILL	GOLD-PF8	FILLSR	Fills a selected range of text
PAGE	KP7	PAGETOP	Moves cursor to right of next page marker

¹Enclose the string in quotes

Table 7-1 (Cont.): Symbols for EDT Functions

EDT Function	Key	Symbol	Explanation
APPEND	KP9	APPENDSR	Removes contents of selected range from the current buffer and appends it to the contents of the paste buffer
REPLACE	GOLD-KP9	CUTSR=DELETE PASTE	Deletes text in selected range and replaces it with the contents of the paste buffer
DELETE WORD	MINUS (-)	DEW	Deletes a word
UNDELETE WORD	GOLD-MINUS	UNDW	Inserts the contents of the Delete word buffer
ADVANCE	KP4	ADV	Sets cursor direction forward
BOTTOM	GOLD-KP4	ER	Moves cursor to end of the buffer
BACKUP	KP5	BACK	Sets cursor direction backward
TOP	GOLD-KP5	BR	Moves cursor to first character at the beginning of the buffer
CUT	KP6	CUTSR	Replaces contents of paste buffer with the selected range; the selected range is removed from the current buffer
PASTE	GOLD-KP6	PASTE	Inserts contents of the paste buffer into the current buffer
DELETE CHARACTER	COMMA (KP)	D+C	Deletes a single character
UNDELETE CHARACTER	GOLD-COMMA	UNDC	Inserts the contents of the delete character buffer
WORD	KP1	W	Move ahead one word

(continued on next page)

Table 7-1 (Cont.): Symbols for EDT Functions

EDT Function	Key	Symbol	Explanation
CHANGE CASE	GOLD-KP1	CHGCSR	Change the case of current character, or entire select range if one is active
END OF LINE	KP2	EL	Move cursor to end of line
DELETE TO END OF LINE	GOLD-KP2	D+EL	Delete text from cursor to end of the line
CHARACTER	KP3	+C	Move ahead one character
LINE	KP0	L	Move cursor to beginning of next line
OPEN LINE	GOLD-KP0	(<CR>-C)	Insert open line
SELECT	PERIOD (KP)	SEL	Begin a select range
RIGHT	Right arrow	+C	Move right one character
LEFT	Left arrow	-C	Move left one character
UP	Up arrow	-V	Move up one line
DOWN	Down arrow	+V	Move down one line

To use the EDT functions in a key definition, use the symbols shown in the table. For example, consider the example shown earlier, where a key was defined to delete the last two columns of a list such as this:

Item	Quantity	Price	Total
----	-----	-----	-----
Apples	20	1.00	20.00
Bananas	40	1.50	60.00
Beets	25	2.00	50.00
Carrots	30	2.00	60.00
Peaches	10	3.00	30.00
Pears	5	6.00	30.00
Potatoes	50	1.00	50.00
Oranges	20	4.00	80.00

The desired series of steps was as follows:

1. Move the cursor ahead two words.
2. Delete the text from the cursor to the end of the line.
3. Move the cursor to the beginning of the next line.

As shown in Table 7-1, the symbols for these functions are as follows:

1. WW [To move the cursor ahead two words]
2. D+EL [To delete the text from the cursor to the end of the line]
3. L [To move the cursor to the beginning of the next line]

So, to define the `CTRL+E` key to move the cursor ahead two words, then delete to the end of the line, and then move to the next line, you would include the following line in your EDT initialization file:

```
DEFINE KEY CONTROL E AS 'WWD+EL.'
```

As you see, the symbols for the various EDT functions are simply listed one right after the other, with no intervening punctuation.

Here is another example of a useful key definition, for people who sometimes type letters in not quite the proper order:

```
DEFINE KEY CONTROL D AS '-C-CD+C+CUNDC+C.'
```

This key definition transposes the last two letters that you just typed. For example, suppose you type the following:

```
To be or not to be, thta
```

If you have defined `CTRL/D` as shown above, you could simply press `CTRL/D` and the typographical error will be corrected.

Sample EDT Initialization File

The following example shows an EDT initialization file that you could use or adapt to meet your needs. After this initialization file is created, you specify its use with the `/COMMAND=` qualifier in the `EDIT/EDT` command line.

In this example, the symbols `^Z` and `<CR>` represent `CTRL/Z` and `RETURN`, respectively, as explained earlier in this section.

```
set entity word '<>)]' ①
set mode change ②
set quiet ③
set wrap 70 ④
define key gold a as 'i<p><cr><list>(unnumbered)<cr><le>^z.' ⑤
define key gold b as 'i<CR><endlist><p><CR>^Z.' ⑥
define key gold d as '"(+csel)"i<CR>^Zappendsrl.' ⑦
define key gold e as 'i<CR>^Zext delete . thru end.' ⑧
define key control e as '-c-cd+c+cundc+c.' ⑨
define key gold f as 'brext delete . thru end.' ⑩
define key control f as 'brselerfillsrbr.' ⑪ fills entire buffer
define key control l as 'i<CR><le>^z.' ⑫
define key control z as 'ext ex.' ⑬
define key gold m as 'ext =main..' ⑭
define key gold w as 'ext set screen 132.' ⑮
define key gold x as 'ext =x..' ⑯
```

7-40 Editing Text Files: Using EDT

The first four lines of this sample initialization file have the following effect:

- ① The WORD function stops at the symbols listed, in addition to the existing defaults, each time you use EDT.
- ② Puts EDT in keypad mode when you enter the editor.
- ③ Turns off the terminal bell that would otherwise go *BRAP!* when a message is displayed.
- ④ Sets the right margin to 70 when you use EDT.

The following table lists the keys that are defined in this initialization file:

Ref.	Key or Sequence	Action Taken When Key or Sequence Is Pressed
	<code>GOLD-A</code>	A carriage return and the following text are inserted in the editing buffer: <pre><p> <list> (unnumbered) <le></pre>
	<code>GOLD-B</code>	A carriage return and the following text are inserted in the editing buffer: <pre><endlist> <p></pre>
	<code>GOLD-D</code>	The text within the next set of parentheses is removed from the current buffer and appended (with a carriage return) to the paste buffer. Using this key, you can extract data that is enclosed in parentheses, and create a list of the data elements in the paste buffer (which can then be written to a file).
	<code>GOLD-E</code>	Deletes all of the text between the cursor position and the end of the buffer.
	<code>CTRL/E</code>	The two characters that immediately precede the cursor are transposed. For example, to change <i>teh</i> to <i>the</i> with a single keystroke, you could use this <code>CTRL/E</code> key definition.
	<code>GOLD-F</code>	Deletes the entire contents of the buffer (from beginning to end).
	<code>CTRL/F</code>	The entire buffer is placed in a select range and filled.

Ref.	Key or Sequence	Action Taken When Key or Sequence Is Pressed
	<code>CTRL/L</code>	A carriage return and the following text are inserted in the editing buffer: <le>
	<code>CTRL/Z</code>	The edits you have made are saved, the file is written, and you exit from EDT. This is the same as pressing <code>CTRL/Z</code> and then typing EXIT at the asterisk (*) prompt.
	<code>GOLD-M</code>	EDT switches to the default buffer (MAIN), at the same cursor position as when you were last in that buffer.
	<code>GOLD-W</code>	The screen width on your terminal is set to to 132 columns.
	<code>GOLD-X</code>	EDT switches to a buffer named X. If buffer X does not exist, this command creates it; if buffer X does exist, the cursor position is the same as when you were last in that buffer.

7.7.4 Summary

- You can define `CTRL` keys and `GOLD` key sequences in EDT. Key definitions can either insert text, reproduce EDT functions, or combine text insertions with EDT functions.
- You can define keys during an EDT editing session, or you can define keys in an EDT initialization file. Keys defined during an editing session do not apply after you exit from EDT, but keys defined in an initialization file apply whenever the initialization file is specified with the `/COMMAND=` qualifier in the `EDIT/EDT` command line.
- To define a key during an editing session, press `CTRL/K`, then press the key that you want to define. Begin the key definition with an open parenthesis. When you want to insert text, signal the beginning of the text by typing the letter *i*, and signal the end of the text by pressing `CTRL/Z`. When you want to include EDT functions in your key definition, just press the appropriate keypad function key. To indicate that you want to search for a text string in your key definition, enclose the text in quotes. Complete the key definition by typing a closed parenthesis and a period, and then by pressing `ENTER`.
- To define keys in an initialization file, put the definition for each key on a single line, and begin each line with the syntax:

```
DEFINE KEY key-name AS key-definition
```

The key definition should begin with a single quote, and it should end with a period followed by a single quote.

To include text in a key definition, indicate the start of text with the letter *i*, and indicate the end of the text by inserting the symbol for `CTRLZ`. You can insert the `CTRLZ` symbol in your initialization file using the sequence `GOLD-28-GOLD-KP3`.

To include EDT functions in your key definitions, use the function symbols listed in Table 7-1. To search for text, use quotation marks to enclose the text that is the search string.

Troubleshooting

If you are having trouble defining a key, be sure of the following:

- All text strings begin with *i* and end with `CTRLZ`
- Use quotation marks to search for text strings
- When defining keys in an initialization file, use the proper symbols for EDT functions (as shown in Table 7-1)

7.8 Controlling EDT Sessions

You can control some of the characteristics of an EDT editing session with the SET commands. You can also define a macro (a sequence of line-editing commands) and define keys in EDT. You can enter these control commands interactively, or you can include them in an EDT startup command file.

7.8.1 Controlling Screen Format with SET Commands

Several EDT commands control the format of a screen display. Some are listed below. See the EDT commands in the Reference Section for a comprehensive list of the SET commands.

- **SET LINES *n***—Controls the number of lines that EDT displays on the screen. This number, which can be set from 1 to 22, defaults to 22. To set the screen to 15 lines, for example, type:

```
Command: SET LINES 15
```

Note that if you are editing at slow baud rates, setting the number of lines low will increase your editing speed.

- **SET SCREEN width**—Controls the maximum length of the line EDT displays; the default width is 80 characters. (When there are more characters than the SET SCREEN command specifies, EDT displays a diamond at the end of the line.)

```
Command: SET SCREEN 132
```

If you use the SET SCREEN command to make the screen wider than 80 on either a VT100- or VT200-series terminal, EDT changes the terminal's screen width to 132.

- **SET [NO]TRUNCATE**—Controls whether the characters that exceed the **SET SCREEN** width are displayed on the next line. The default is **SET TRUNCATE**, which ends the display of a line at the value of **SET SCREEN**.

Command: SET [NO]TRUNCATE

- **SET [NO]WRAP n**—Specifies *n* character positions as the point at which text will be moved to the beginning of the next line. When you are inserting text in EDT keypad mode and the cursor position exceeds the value of *n*, EDT wraps the next full word to the next line. (However, when you insert text in the middle of a line, that line does not always wrap.) The default is **NOWRAP**. To wrap the text exceeding 75 characters, for example, type:

Command: SET WRAP 75

The **SET** commands have corresponding **SHOW** commands; see the **EDT** commands in the Reference Section for a list of **SHOW** commands.

7.8.2 Controlling Editing Functions with SET Commands

Several commands control EDT's responses during an editing session, as follows. (See the **EDT** commands in the Reference Section for a comprehensive list of the **SET** commands.)

- **SET ENTITY**—Defines boundaries for the **WORD**, **SENTENCE**, **PARAGRAPH**, and **PAGE** entities. (The **SENTENCE** and **PARAGRAPH** entities are not used by any default key definitions; consequently, they are useful only in the key definitions you create with the **DEFINE KEY** command.) For example, the default boundaries for the **WORD** entity are a line feed, tab, form feed, line terminator, and space. To make the period and comma the only delimiters of the word entity, enter the following **SET ENTITY** command:

Command: SET ENTITY WORD ',.'

- **SET MODE**—Controls the EDT editing mode to be entered when the processing of the **EDTINI.EDT** file is completed (either line or change mode, which is keypad mode). For example, to enter change mode instead of line mode at the beginning of editing sessions, insert the following command at the end of your **EDT** startup command file:

```
SET MODE CHANGE
```
- **SET QUIET**—Suppresses the sound made when EDT issues an error message in keypad mode. The default is **NOQUIET**.

7.8.3 Defining EDT Macros

An EDT macro allows you to execute a sequence of EDT line-editing commands whenever you invoke the macro. To define a macro, use the EDT line-editing command `DEFINE MACRO` to define the name of a buffer as the macro name. Then create and enter a buffer with the same name as the macro. (See Section 7.6.8 for information about using multiple buffers.) Once in the buffer, type the EDT line-editing commands in the desired sequence, one command per line. For example, the following macro inserts a four-line heading:

```
INSERT;NAME:  
INSERT;DEPT:  
INSERT;DATE:  
INSERT;SUBJ:  
[EOB]
```

Then exit from the buffer. To invoke the macro, enter its name as an EDT line-editing command. The lines of the heading are inserted at the cursor position:

```
NAME:  
DEPT:  
DATE:  
SUBJ:
```

To make a macro available during other editing sessions, you can place the `DEFINE MACRO` command and the macro command sequence in an EDT startup command file. When you include a macro definition in a startup command file, be sure the command sequence contains the commands for entering the macro buffer (`FIND=buffer-name.`) and returning to the MAIN buffer (`FIND=MAIN.`). Note that you must precede each command in the sequence with the `INSERT` command. For more information about macro definitions, see Section 7.8.3.

Chapter 8

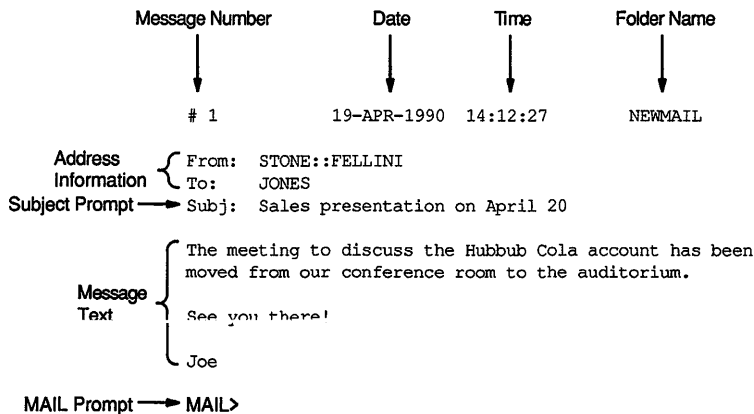
MAIL: Communicating with Other Users

MAIL lets you send messages to other users on your system or on any other computer that is connected to your system with the DECnet-VAX network. This chapter describes the routine tasks you can perform using MAIL and how you can customize MAIL to fit your needs.

For more information about MAIL commands and qualifiers, see the MAIL description in the Reference Section or type HELP at the MAIL> prompt.

Figure 8-1 shows a sample mail message and its components.

Figure 8-1: Sample Mail Message



ZK-0980A-GE

8.1 Invoking and Exiting MAIL

To perform MAIL tasks, you invoke MAIL and enter MAIL commands at the MAIL> prompt.

When the MAIL> prompt has been displayed, you can enter the appropriate MAIL commands to perform the following tasks:

- Read a mail message
- Send a mail message
- Reply to a mail message
- Forward a mail message
- Organize mail messages

The remaining sections in this chapter describe these tasks and provide examples for performing them. The Reference Section lists and describes the MAIL commands and their qualifiers.

Invoking MAIL

To invoke MAIL, enter the following command at the DCL prompt:

```
$ MAIL
```

MAIL displays the following prompt:

```
MAIL>
```

Exiting from MAIL

To exit from MAIL, enter the EXIT command at the MAIL prompt.

```
MAIL> EXIT
```

You can also exit from MAIL by pressing CTRL/Z or using the QUIT command.

NOTE: If you have entered the text of a message, pressing CTRL/Z will send the message. To cancel a send operation without exiting from MAIL, press CTRL/C.

8.2 Reading Messages

Invoke MAIL to read an old or new mail message. Messages you receive are stored in mail files, which have a default file type of MAI. In this file, by default, MAIL provides two **folders** that store old and new messages. New messages are automatically placed in a folder called NEWMAIL; old messages are placed in a folder called MAIL. After you read a new message, the message is automatically moved from the NEWMAIL folder to the MAIL folder. You can move between

these folders to read old or new mail messages by using the **SELECT** command. For information about reading old messages, see Section 8.2.2.

8.2.1 Reading a New Message

When you are logged in to your account and receive a mail message, **MAIL** notifies you. For example, notification of a message sent by user **FELLINI** is displayed as follows:

```
New mail on node DOODAH from STONE::FELLINI      (10:02:23)
```

To read a new message, use the following procedure:

1. Invoke **MAIL**.
2. Press **RETURN** at the **MAIL>** prompt.

```
MAIL> RET
```

If you have more than one new message, press **RETURN** at the **MAIL>** prompt to read the other messages. When you have read all your new messages, **MAIL** issues the message “%MAIL-E-NOMOREMSG, no more messages,” and continues to prompt for commands until you exit **MAIL**.

If you receive a mail message while you are in **MAIL**, enter the **READ/NEW** command to read the new message.

8.2.2 Reading Old Messages

To reread old mail messages in your default **MAIL** folder, use the following procedure:

1. Enter the **SELECT** command at the **MAIL>** prompt:

```
MAIL> SELECT MAIL
```

MAIL places you in the **MAIL** folder.

2. To read the first message in your default **MAIL** folder, press **RETURN** at the **MAIL>** prompt or enter the **READ** command.

MAIL displays the first message (1) in your default mail file on the screen.

3. To display the next message, press **RETURN**.
4. If the message is too long to display on one screen, press **RETURN** to display the next part of the message.
5. To skip part of a message and display the next message, enter **NEXT**.

8-4 MAIL: Communicating with Other Users

To read a particular message in your default MAIL folder, use the following procedure:

1. Enter the DIRECTORY command at the MAIL> prompt:

```
MAIL> DIRECTORY
```

MAIL displays a list like the following:

```
MAIL> DIRECTORY
```

			MAIL
#	From	Date	Subject
1	DOLCE::FELLINI	19-APR-1990	Sales presentation on April 20
2	DOODAH::JONES	19-APR-1990	Status

```
MAIL>
```

2. Enter the number of the message you want to read at the MAIL> prompt.

```
MAIL> 2
```

MAIL displays the message that you selected.

If you have many messages, you can locate a particular message by using the SEARCH command to find a specified string. To search for a string, specify that string as a parameter to the SEARCH command, as shown in the following example:

```
MAIL> SEARCH "appointment"
```

The SEARCH command selects and displays the first message in the current folder that contains the specified string.

To search for a new string, specify the string as a parameter to the SEARCH command. Each time you specify a new string, the SEARCH command starts the search at message number 1. To continue searching the folder for messages that contain the specified string, use the SEARCH command without specifying a parameter.

8.3 Sending a Message

To send a mail message to any user on your system, do the following:

1. Enter SEND at the MAIL> prompt:

```
MAIL> SEND
```

MAIL prompts you for the name of the user you want to receive the message.

```
To:
```

2. Type the name of the user receiving the message and press RETURN.

```
To: THOMPSON
```

MAIL prompts you for the subject of the message.

```
Subj:
```

3. Enter the subject of the message and press RETURN. Entering this information is optional.

Subj: Meeting on April 20

MAIL prompts you for the text of the message.

Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:

4. Enter the text of a message, or just press RETURN. Entering this information is optional.

I have some new ideas about the Hubbub Cola account. Let me know
when you're available to talk about them.

--Jeff

5. Press CTRL/Z to send the message. If you decide not to send the message, press CTRL/C, which cancels the send operation without exiting from MAIL.

8.3.1 Sending MAIL over the Network

If your computer system is part of a network, you can send mail to any other user on the network. If you are sending mail to someone on a different node, you must enter the user's node name and user name at the To: prompt using the following format:

nodename::username

For example, to send a message to user HIGGINS on node CHEETA, enter the following command and user name:

```
MAIL> SEND
To: CHEETA::HIGGINS
```

MAIL displays a message if the network connection to the remote node is not available. Wait a while and try to send the message later.

8.3.2 Sending a Message to More Than One User

You can send mail to several users at the same time in two ways: using individual user names at the To: prompt or using a distribution list.

To send the same message to several users using their user names, enter the user names at the To: prompt and separate them with commas. For example, to send a message to Thompson, Jones, and Barney, enter the following:

```
MAIL> SEND
To: THOMPSON, JONES, BARNEY
Subj: Meeting on January 9
```

Creating a Distribution List

A distribution list is a file that contains a list of users and their node names. You must use a text editor to create distribution lists; distribution lists are not created within MAIL.

To create a distribution list, use the following procedure:

1. Create a file, with the file type DIS, using a text editor.
2. Enter one user name per line in the file.
3. To include the names of other distribution lists in the file, specify an at sign (@) followed by the name of the distribution list.
4. To include comments in the file, enter an exclamation point (!).

The following example shows a distribution list file:

```
! ALLBUDGET.DIS
!  
! Budget Committee Members  
@BUDGET          ! listed in BUDGET.DIS.  
! Staff  
  Thompson  
  BRUTUS::JONES  
  PORTIA::BARNEY
```

In the preceding example, if the file BUDGET.DIS is not in the same directory as the new distribution list file you are creating (ALLBUDGET.DIS), include the file specification for BUDGET.DIS in the new distribution file. Depending on where you create ALLBUDGET.DIS, you might have to specify the device and directory in which BUDGET.DIS is located. (See Chapter 1 for more information about file specifications.)

Sending a Message to a Distribution List

To send mail to several users using a distribution list, use the following procedure:

1. Use a text editor to create a distribution list file.
2. Invoke MAIL.
3. Type SEND at the MAIL> prompt and press RETURN:
MAIL> SEND [RET]
4. Type an at sign (@) and the file name at the To: prompt. Press RETURN.
To: @ALLBUDGET [RET]
5. Type the subject of the message at the Subj: prompt and press RETURN.
Subj: Tomorrow's Meeting [RET]

6. Enter the text of the message at the text prompt.

Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:

The meeting about the Hubbub Cola account is tomorrow at 2:00.

--Jeff

By default, the system looks for a distribution list file with the file type DIS. If the file containing your distribution list has a different file type, you must specify the file name and file type at the To: prompt. If you invoke MAIL while in one directory and the file containing the distribution list is in another, enter the distribution list's full directory name at the To: prompt.

8.3.3 Sending a File

You can send a file to other users from within MAIL or from DCL level. Use the following procedure to send a file from within MAIL:

1. At the MAIL> prompt, enter SEND and the name of the file you want to send.

```
MAIL> SEND MEMO.TXT
```

2. At the To: prompt, enter the user name of the person you want to receive the file.

```
To: EDGELL
```

3. At the Subj: prompt, enter the subject of the file.

```
Subj: Another memo
```

4. To send the file, press RETURN; to cancel the send operation, press CTRL/C or CTRL/Y. CTRL/C keeps you within MAIL; CTRL/Y returns you to DCL level.

When you send a file from DCL level, MAIL is invoked, but you do not enter an interactive session, nor do you see the MAIL> prompt. When the file is sent, you are automatically returned to DCL level. When you are sending a file from DCL level, the argument to the (optional) /SUBJECT qualifier must be enclosed in quotation marks if it contains any spaces or nonalphanumeric characters.

For example, to send the file MEMO.TXT to user EDGELL on node CHEETA at DCL level, use the following procedure:

1. At the DCL prompt, enter the following command:

```
$ MAIL/SUBJECT="Another memo" MEMO.TXT CHEETA:EDGELL
```

2. Press RETURN to send the file; press CTRL/C to cancel the send operation.

8.3.4 Creating a File from a Message

To create a text file from a message, enter the **EXTRACT** command and the file name at the **MAIL>** prompt while you are reading the message. For example, to create a file named **JAN_MEETINGS.TXT** from the following mail message, enter the following command:

```
#1                19-APR-1990  14:12:27          NEWMAIL
From:  STONE::FELLINI
To:    Thompson
Subj:  Dates for January sales meetings

Sales meetings in January will be held on the following dates:
    Wednesday Jan.  3, 1990
    Tuesday   Jan.  9, 1990
    Monday    Jan. 15, 1990
    Thursday  Jan. 25, 1990
MAIL>EXTRACT JAN_MEETINGS.TXT
```

MAIL displays a message like the following one:

```
%MAIL-CREATED, DISK:[THOMPSON]JAN_MEETINGS.TXT.
```

When you exit from **MAIL**, the file is listed in your current directory (unless you specify another directory).

The mail header is composed of the **From:**, **To:**, and **Subj:** lines. To create a file that does not include header information, specify the **/NOHEADER** qualifier to the **MAIL** command.

The following example shows how to create a file named **JANUARY_MEETINGS.TXT** containing the text of message number 3:

```
MAIL> READ 3
.
.
.
MAIL> EXTRACT/NOHEADER JANUARY_MEETINGS.TXT
%MAIL-I-CREATED, DISK1:[JONES]JANUARY_MEETINGS.TXT;1 created
MAIL>
```

If the message has more than one header (for example, a forwarded message), only the topmost header is deleted.

Use the **/APPEND** qualifier to the **EXTRACT** command to copy a message to the end of an existing file. Use the **/ALL** qualifier to copy all the files in the current folder to an existing file.

8.4 Replying to a Message

To reply to a message you have received, use the following procedure:

1. Type **REPLY** at the **MAIL>** prompt and press **RETURN**.

MAIL displays the following header information:

```
To: STONE::THOMPSON
Subj: RE: Budget Meeting
Enter your message below. Press CTRL/Z when complete. CTRL/C to quit:
```

2. Type your message and press **CTRL/Z** to send the message; press **CTRL/C** to quit.

8.5 Forwarding a Message

To forward a mail message to other users, enter the **FORWARD** command at the **MAIL>** prompt after you have read the message.

8.6 Organizing Your Messages

To organize your mail messages, you can create your own mail folders and files. Each folder and file can contain any number of messages. The name of the current folder is displayed in the top right corner of the screen each time you enter a **READ** or **DIRECTORY** command. You can work only with messages that are in your current folder.

Like the default mail folders (**NEWMAIL**, **MAIL**, **WASTEBASKET**), the folders you create are normally stored in the mail file **MAIL.MAI**.

8.6.1 Creating and Modifying Folders

The following MAIL commands allow you to create and modify folders:

- **FILE** or **MOVE**—Files the current message in the folder you specify. If the folder does not exist, MAIL displays a message asking if you want to create it. After being filed, the message is automatically deleted from the current folder.
- **COPY**—Places a copy of the current message into the folder you specify. If the folder does not exist, MAIL displays a message asking if you want to create it. The following commands copy all messages containing the word **MEETING** from the current folder to a folder named **SCHEDULE**. After the commands

8-10 MAIL: Communicating with Other Users

are executed, you have two copies of each message, one in the current folder and one in the folder SCHEDULE. The first command selects and displays the first message containing the word *meeting*:

```
MAIL> SEARCH MEETING

MAIL> COPY SCHEDULE
Folder SCHEDULE does not exist.
Do you want to create it (Y/N, default is N)?Y
%MAIL-I-NEWFOLDER, folder SCHEDULE created
```

This command selects and displays the next message containing *meeting*:

```
MAIL> SEARCH

MAIL> COPY SCHEDULE
MAIL> SEARCH
%MAIL-E-NOTFOUND, no messages containing 'MEETING' found
```

8.7 Selecting Folders

To display a list of the folders in your current mail file, enter the DIRECTORY/FOLDER command, as shown in the following example:

```
MAIL> DIRECTORY/FOLDER
Listing of folders in SYS$LOGIN:[JONES]MAIL.MAI;1
  Press CTRL/C to cancel listing
MAIL      MEETING_MINUTES
MEMOS     PROJECT_NOTES
STAFF
```

To select a new folder as your current folder, use one of the following commands:

- **SELECT**—Selects the specified folder as the current folder.
- **DIRECTORY**—Selects the specified folder as the current folder and lists the messages in the folder.
- **READ**—Selects the specified folder as the current folder and displays the specified message (by default, the first message in the folder).

Deleting Folders

To delete a mail folder, delete all the messages in the folder or move them to another folder. For example, to delete the messages in the MUSIC folder, enter the following commands:

```
MAIL> SELECT MUSIC
%MAIL-I-SELECTED, 2 messages selected
MAIL> DELETE/ALL
```

Creating and Accessing Mail Files

To create a mail file, move a message into the file by entering the COPY, MOVE, or FILE command as you would to create a folder. When MAIL prompts you for the name of the folder, specify the name of the mail file after the name of the folder.

For example, to create the mail file ACCOUNTS.MAI, move the current message into a folder named FEED in the file ACCOUNTS.MAI, and delete the message from its current folder and file, enter the following commands:

```
MAIL> MOVE
 Folder: FEED [RET]
 File: ACCOUNTS [RET]
```

To work within a mail file other than the default mail file, use the MAIL command SET FILE to specify the alternate file. (The MAIL command SHOW FILE displays the name of the current mail file.) When you change mail files, the WASTEBASKET folder of the current mail file is emptied and deleted, and the mail file is closed.

8.8 Deleting Messages

To delete a mail message from the current folder, either enter the DELETE command while you are reading the message or enter the DELETE command followed by the number (or range of numbers) of the message you want to delete. For example, to delete messages 4, 5, 6, 11, 12, 14, 15, 16, and 17, enter the following at the MAIL> prompt and press RETURN:

```
MAIL> DELETE 4-6,11,12,14:17
```

You can use either the hyphen (-) or the colon (:) to define the range of messages to be deleted.

Recovering Deleted Messages

When you delete a message, the message is moved to a folder called WASTEBASKET. Deleted messages collect in the WASTEBASKET folder until you exit from the current mail file (either by exiting from MAIL or by specifying a different mail file). When you exit from the current mail file, WASTEBASKET is emptied and the folder itself is deleted. During your interactive MAIL session, you can recover any deleted message by moving the message out of the wastebasket folder.

8.9 Customizing Your MAIL Environment

This section describes the following tasks that can help you use MAIL more efficiently. These tasks are as follows:

- Creating a mail subdirectory
- Using a text editor in MAIL
- Using the MAIL keypad

8.9.1 Creating a Mail Subdirectory

When you receive mail messages, they are by default written to files named MAIL\$xxxxxxxxx.MAI located in your top level directory. (Note that the X's represent a long, random file specification.) Your default mail file, MAIL.MAI, is created in your top level directory the first time you receive a mail message. To avoid the display of MAI files in your top level directory, use the MAIL command SET MAIL_DIRECTORY. This command creates a mail subdirectory and moves all your MAI files to that subdirectory. To move the MAI files from a subdirectory back to your top level directory, use the SET NOMAIL_DIRECTORY command.

TIP: To display the name of the subdirectory that contains all your MAI files, enter SHOW MAIL_DIRECTORY at the MAIL> prompt.

```
MAIL> SHOW MAIL_DIRECTORY
```

MAIL displays the following message:

```
Your mail file directory is DISK$:[FELLINI.MAIL]
```

8.9.2 Using the Mail Keypad

You can use the numeric keypad on your keyboard to execute commands in MAIL. Most keypad keys can execute two commands. To enter the top command for each key shown in the following diagram, press the appropriate key. To enter the bottom command shown in the following diagram, press the PF1 key before you press the key.

PF1 GOLD	PF2 HELP DIR/FOLDER	PF3 EXT/MAIL EXTRACT	PF4 ERASE SEL/MAIL
7 SEND SEND/EDIT	8 REPLY REP/ED/EXT	9 FORWARD FORWD/EDIT	— READ/NEW SHOW/NEW
4 CURRENT CURRENT/EDIT	5 FIRST FIRST/EDIT	6 LAST LAST/EDIT	, DIR/NEW DIR MAIL
1 BACK BACK/EDIT	2 PRINT PRINT/PR/NOT	3 DIR DIR/ST=99999	ENTER SELECT
0 NEXT NEXT/EDIT	. FILE DELETE		

ZK-1744-GE

For example, to execute the MAIL command SEND, press the keypad key 7. To execute the MAIL command SEND/EDIT, press the PF1 key first and then press keypad key 7. (For more information about mail keypad commands, see MAIL in the Reference Section.)

You can redefine the keypad keys to execute MAIL commands when you are in MAIL. Defining keypad keys in MAIL is similar to defining keypad keys to execute DCL commands; see the DEFINE/KEY command in the MAIL part of the Reference Section for more information.

8.9.3 Using a Text Editor in MAIL

You can use a VMS text editor to write your message before you send it. To do so, specify the /EDIT qualifier with the SEND command as shown in the following example:

```
MAIL> SEND/EDIT
```

After you respond to the To: and Subj: prompts, MAIL invokes the text editor. By default, MAIL invokes the EDT editor.

8-14 MAIL: Communicating with Other Users

If you see an asterisk (*) after you enter the subject line and press RETURN, press the C key to enter the screen editor. To send the message, press CTRL/Z and enter the EXIT command; to cancel the send operation, press CTRL/Z and enter the QUIT command.

TIP: By specifying /EDIT when you invoke MAIL, you can use the editor for send, reply, and forward operations during the ensuing mail session.

Setting the Default Editor

By default, MAIL invokes the EDT editor when you specify the MAIL command SEND/EDIT. By entering the TPU parameter to the MAIL command SET EDITOR, you can specify that the Text Processing Editor be invoked instead. (EVE is the default TPU editor.) The TPU editor remains your default MAIL editor (even if you log out of the system and log back in) until you enter the SET EDITOR EDT command.

For example, to set the default MAIL editor to TPU, enter the following command at the MAIL> prompt:

```
MAIL> SET EDITOR TPU
```

In the following example, the default MAIL editor has been set to TPU, and the MAIL command SEND/EDIT has been entered at the MAIL> prompt. MAIL displays the following screen:

```
Buffer  MAIN
```

```
| Insert | Forward
```

Enter the text of your message using EVE commands to move around in the **buffer**. A buffer is a temporary storage area that exists only during an editing session. To send the message, press CTRL/Z.

To display the name of the default MAIL editor, enter the MAIL command SHOW EDITOR.

```
MAIL> SHOW EDITOR
```

MAIL displays the default MAIL editor as follows:

```
Your editor is TPU.
```


Chapter 9

VMS SORT/MERGE: Sorting and Merging Files

This chapter describes how to use VMS SORT/MERGE to perform the following tasks:

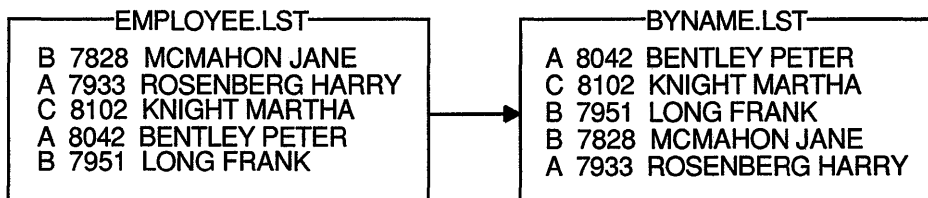
- Sort records from one or more input files according to the fields you select and generate one reordered output file.
- Merge up to 10 input files that have been previously sorted according to the same key fields and generate one output file.

9.1 Sorting Records

A file record is similar to a line of text in a file. Record sorting, the default sort operation, keeps records intact and produces an output file consisting of complete records. Records can be subdivided into fields, which describe individual segments of the record. A field is specified by the starting position of its first character in the record and the length, in characters, of the field. You can sort records based on the contents of certain fields by specifying the field as a sort key.

The following example illustrates an ascending (the default) record sort based on that portion of each record starting at character position 8 and extending to the end of the record (the name):

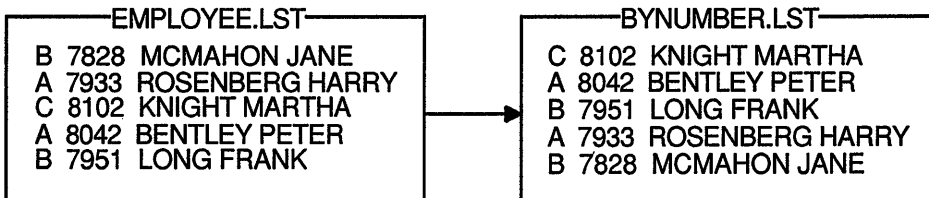
```
§ SORT/KEY=(POSITION=6,SIZE=15) EMPLOYEE.LST BYNAME.LST
```



9-2 VMS SORT/MERGE: Sorting and Merging Files

The following example sorts the same file in descending order using the field in character positions 3 through 6 (the number) as the sort key:

```
§ SORT/KEY=(POSITION=3,SIZE=4,DESCENDING) EMPLOYEE.LST BYNUMBER.LST
```



ZK-1749-GE

The first parameter of the SORT command names the file or files to be sorted. Multiple files are treated as one large file for sorting purposes. The second parameter provides a name for the ordered output file that the sort will create. The following example sorts the records in two files, EMPLOYEE.LST and EMPLOYER.LST, and creates the ordered output file BYNAME.LST:

```
§ SORT EMPLOYEE.LST,EMPLOYER.LST BYNAME.LST
```

Single Key

By default, the SORT command assumes that a key field in a record has the following characteristics:

- Begins in the first position of a record
- Includes the entire record
- Contains character data
- Will be sorted in ascending order

Use the /KEY qualifier to specify characteristics of the key field other than those assumed by default.

In the following example, the /KEY qualifier specifies that the key field starts in position 8 and is 15 characters long:

```
§ SORT/KEY=(POSITION=8,SIZE=15) EMPLOYEE.LST BYNAME.LST
```

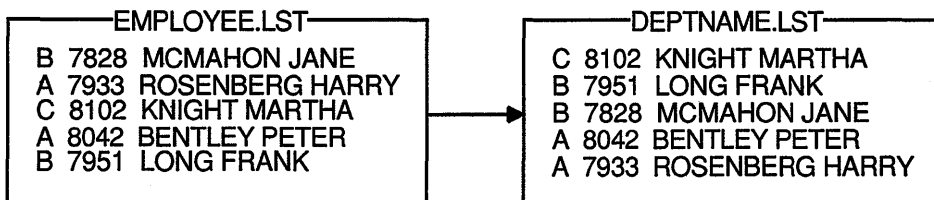
(If an actual key would have to extend beyond the end of the record to meet the size specification—for example, if the key is the last item in a variable-length format—the missing characters are treated as null characters.)

Multiple Keys

You can specify more than one key field, up to a limit of 255 characters. Each key can be ascending or descending. Specify multiple keys in the order of their priority in the sort. For example, the following command sorts records first on the value of position 1 in descending order, then on the value of positions 8 through 27 (or the end of the record) in ascending order:

```
$ SORT/KEY=(POSITION=1,SIZE=1,DESCENDING) -
  _$/KEY=(POSITION=8,SIZE=15) -
  _$ EMPLOYEE.LST DEPTNAME.LST
```

The results of the sort specified in the preceding example are as follows:



ZK-1764-GE

By default, records with identical keys are kept but not sorted predictably. To retain identical keys and arrange them according to the input file order, specify the /STABLE qualifier. To eliminate duplicate keys, specify the /NODUPLICATES qualifier.

9.2 Sorting Character Data Files

The SORT command assumes by default that the files to be sorted contain character data. Characters are sorted according to a collating sequence, which describes the order in which characters are arranged (A, B, C, and so on).

ASCII is the default collating sequence for character data. In general, ASCII orders numbers (0 through 9) first, then uppercase letters (A through Z), and then lowercase letters (a through z).

You can specify the EBCDIC collating sequence to generate an output file that is ordered in EBCDIC sequence (although it remains in ASCII representation). To use the EBCDIC collating sequence, specify the /COLLATING_SEQUENCE=EBCDIC qualifier.

The multinational collating sequence collates characters according to the international character set defined by Digital (see the Reference Section). The multinational collating sequence compares for different characters first, then for different diacritical forms of the same character (formed by using diacritical marks as part of "compose sequences" on VT200-series terminals), and then for different cases (uppercase or lowercase) of the same character.

9-4 VMS SORT/MERGE: Sorting and Merging Files

To use the multinational collating sequence, specify the /COLLATING_SEQUENCE=MULTINATIONAL qualifier.

NOTE: Use caution when using the multinational collating sequence to sort or merge files for further processing. Sequence-checking procedures in most programming languages compare numeric characters. Because the multinational sequence is based on actual graphic characters (and not the codes representing those characters), normal sequence checking will not work.

9.3 Sorting Noncharacter Data Files

If you sort files containing items other than character data, you must specify the data type of each key. Also, you must take care in calculating starting positions and sizes, because the items being compared may occupy more than 1 byte. For example, if you are sorting a file that contains 20 characters followed by 3 floating-point numbers in F_floating format, and the key is the last floating-point number, you must make the following specification:

```
$ SORT/KEY=(POSITION=29,F_FLOATING) STATS.RAW STATS.SOR
```

In the example, the character data occupies positions 1 through 20 (20 characters), the first F_floating-point number occupies position 21 through 24, the second F_floating-point number occupies positions 25 through 28, and the third F_floating-point number occupies positions 29 through 32. The size of the floating-point number is not specified (because it is fixed at 4 bytes).

9.4 Entering Records from a Terminal

The records to be sorted or merged need not be in a file. You can enter the records directly from the terminal as you enter the SORT or MERGE command.

To enter the input records for a sort or merge operation from your terminal, specify SYS\$INPUT as the input file parameter, qualifying it with the size of the longest record (in bytes) and the approximate size of the input file (in blocks). After you enter the command, enter the input records on successive terminal lines. End each record by pressing RETURN. End the file by pressing CTRL/Z.

The following example demonstrates a sort operation in which the input records to be sorted are entered directly from the terminal:

```
$ SORT/KEY=(POSITION=8,SIZE=15) -  
_ $ SYS$INPUT/FORMAT=(RECORD_SIZE=22,FILE_SIZE=10) BYNAME.LST  
B 7828 MCAHON JANE [RET]  
A 7933 ROSENBERG HARRY [RET]  
C 8102 KNIGHT MARTHA [RET]  
A 8042 BENTLEY PETER [RET]  
B 7951 LONG FRANK [RET]  
[CTRLZ]
```

9.5 Submitting Batch Jobs

If you are sorting large files, you should consider submitting the sort operation as a batch job, because the sort will require some time. Batch jobs are programs or DCL command procedures that run independently of your current session. See Section 10.4 and Section 13 for more information about batch jobs and command procedures, respectively.

If the records to be sorted are in a file, the command procedure you submit as a batch job must contain the SORT command and explicitly set your default directory or include the directory in the command file specifications. The following example submits the DCL command procedure SORTJOB.COM as a batch job. The text of the command procedure is shown following the command line:

```
$ SUBMIT SORTJOB
! SORTJOB.COM
!
$ SET DEFAULT [USER.PER] ! Set default to location of input files
$ SORT/KEY=(POSITION=8,SIZE=15) EMPLOYEE.LST BYNAME.LST
```

You can include the input records in the batch job by placing them after the SORT command, one record per line, as shown in the following example. As with terminal input of records, you specify the input file parameter as SYS\$INPUT and qualify it with the record size (in bytes) and the approximate file size (in blocks):

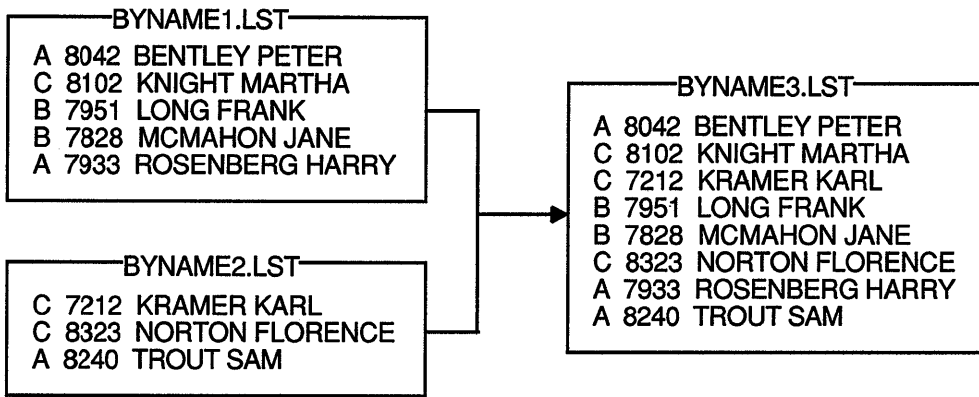
```
$ SUBMIT SORTJOB
! SORTJOB.COM
!
$ SET DEFAULT [USER.PER]
$ SORT/KEY=(POSITION=8,SIZE=15) -
SYS$INPUT-
/FORMAT=(RECORD_SIZE=22,FILE_SIZE=10) -
BYNAME.LST
B 7828 MCMAHON JANE
A 7933 ROSENBERG HARRY
C 8102 KNIGHT MARTHA
A 8042 BENTLEY PETER
B 7951 LONG FRANK
```

9.6 Merging Files

The MERGE command combines up to 10 sorted files into one ordered output file. The input files must all have the same format, and all must have been sorted on the same key fields.

The following example demonstrates the merging of two files based on the field in each record starting at position 8 and extending to the end of the record (the name field):

```
$ MERGE/KEY=(POSITION=8,SIZE=15) BYNAME1.LST,BYNAME2.LST BYNAME3.LST
```



ZK-1771-GE

By default, MERGE does sequence checking to ensure that the input files are in order. The sequence check stops the merge and reports an error if a record is found to be out of order. To prevent sequence checking during the merge, specify the /NOCHECK_SEQUENCE qualifier.

Chapter 10

Processes: Using the VMS Environment

A process is an environment created by the system that lets you interact with the VMS operating system.

The system creates a process for you when you perform one of the following tasks:

- Log in—The system creates a process for each interactive user.
- Submit a batch job—The system creates a process for each batch job. When the batch job is completed, the system deletes the process.
- Spawn a subprocess—The system creates a process when you use the SPAWN command.
- Run a program using either the /DETACHED qualifier or the /UIC=uic qualifier.

This chapter describes how and when to use the following processes:

- Subprocesses
- Programs
- Batch jobs

10.1 Interpreting Your Process Context

Characteristics that a process uses, such as privileges, symbols, and logical names form a **process context**. To display the process context for your current process, enter the SHOW PROCESS/ALL command. The following example shows a sample process context:

```
19-APR-1990 13:30:37.12 ①           User: CLEAVER ②
Pid: 24E003DC ③           Proc. name: CLEAVER_1 ④       UIC: [DOC,CLEAVER] ⑤
Priority: 4 ⑥           Default file spec: DISK1:[CLEAVER] ⑦
```

10-2 Processes: Using the VMS Environment

Process Quotas: ⑧

Account name: DOC			
CPU limit:	Infinite	Direct I/O limit:	18
Buffered I/O byte count quota:	31808	Buffered I/O limit:	25
Timer queue entry quota:	10	Open file quota:	57
Paging file quota:	22276	Subprocess quota:	4
Default page fault cluster:	64	AST quota:	38
Enqueue quota:	600	Shared file limit:	0
Max detached processes:	0	Max active jobs:	0

Accounting information: ⑨

Buffered I/O count:	140	Peak working set size:	383
Direct I/O count:	7	Peak virtual size:	2336
Page faults:	304	Mounted volumes:	0
Images activated:	1		
Elapsed CPU time:	0 00:00:00.55		
Connect time:	0 00:00:22.76		

Process privileges: ⑩

GROUP	may affect other processes in same group
TMPMBX	may create temporary mailbox
OPER	operator privilege
NETMBX	may create network device

Process rights identifiers: ⑪

INTERACTIVE
LOCAL
SYS\$NODE_ZEUS

Process Dynamic Memory Area ⑫

Current Size (bytes)	25600	Current Total Size (pages)	50
Free Space (bytes)	19592	Space in Use (bytes)	6008
Size of Largest Block	19520	Size of Smallest Block	24
Number of Free Blocks	3	Free Blocks LEQU 32 Bytes	1

Processes in this tree: ⑬

CLEAVER
CLEAVER_1 (*)

- ① Current date and time—The date and time when the SHOW PROCESS/ALL command is executed.
- ② User name—The user name assigned to the account that is associated with the process.
- ③ Process identification number (PID)—A unique number assigned to the process by the system. The SHOW PROCESS command displays the PID as a hexadecimal number.
- ④ Process name—The name assigned to the process. Since process names are unique, the first process logged in under an account is assigned the user name, and subsequent processes logged in under the same account are assigned the terminal name. You can change your process name with the DCL command SET PROCESS/NAME.

- ⑤ **User identification code (UIC)**—The group and member numbers (or letters) assigned to the account that is associated with the process (for example, [PERSONNEL,RODGERS]). Part of your UIC identifies the group to which you belong. Within a group, users are allowed to share files or system resources more freely than between groups.
- ⑥ **Priority**—The current priority of the process.
- ⑦ **Default file specification**—The current device and directory. Change your current defaults with the DCL command SET DEFAULT.
- ⑧ **Process quotas**—The quotas (limits) associated with the process. Examine these quotas with the /QUOTAS or /ALL qualifiers of the SHOW PROCESS command.
- ⑨ **Accounting information**—The continuously updated account of the process's use of memory and CPU time. Examine this information with the /ACCOUNTING or /ALL qualifiers of the SHOW PROCESS command.
- ⑩ **Process privileges**—The privileges granted to your processes. Privileges restrict the performance of certain system activities to certain users. Examine your privileges with the /PRIVILEGES or /ALL qualifiers of the SHOW PROCESS command.
- ⑪ **Process rights identifiers**—System-defined identifiers that are used in conjunction with access control list protection. Identifiers provide the means of specifying the users in an access control list. An access control list is a security tool that defines the kinds of access to be granted or denied to users of an object, such as a file, device, or mailbox.
- ⑫ **Process dynamic memory area**—The process's current use of dynamic memory. Dynamic memory is allocated by the system to an image when that image is executing. When that memory is no longer needed by one process, the system allocates it to another process. Examine this information with the /MEMORY or /ALL qualifiers of the SHOW PROCESS command.
- ⑬ **Processes in this tree**—A list of subprocesses belonging to the parent process. An asterisk appears after the current process. Examine this with the DCL SHOW PROCESS/SUBPROCESSES or /ALL command.

10.2 Using Subprocesses

The SPAWN command enables you to create a subprocess of your current process. Within this subprocess, you can interact with the system and log out of the subprocess to return to your parent process, or switch between your parent process and subprocesses. Only one of your processes is executing at any time.

10-4 Processes: Using the VMS Environment

By default, the subprocess assumes the name of the parent process followed by an underscore and a unique number. For example, if the parent process name is DOUGLASS, the subprocesses are named DOUGLASS_1, DOUGLASS_2, and so on.

Typically, you use a subprocess in one of the following two ways:

- To interrupt a task, perform a second task, then return to the original task— You can use CTRL/Y to interrupt one task, spawn a subprocess to perform a second task, exit from the subprocess, and then enter the CONTINUE command to return to the original task. By default, when you create a subprocess, the parent process hibernates, and you are given control at DCL level within the subprocess. Your default directory is the current directory of the parent process. (If you interrupt the EDT editor, enter the CONTINUE command and press CTRL/W to refresh the screen.)
- To perform a second task while continuing to work on your original task— You can create the subprocess with the SPAWN/NOWAIT command. SPAWN/NOWAIT generates a noninteractive, batch-like subprocess and is used to execute only commands that do not require input.

Because both the parent and the subprocess are executing concurrently, both attempt to control the terminal. To prevent conflicts, also specify the following:

- /OUTPUT qualifier—Indicates that the subprocess should write output to a specified file rather than to the terminal.
- SPAWN command parameter or /INPUT qualifier—Indicates that the subprocess should execute the specified commands rather than reading input from the terminal.

When you specify the /INPUT qualifier of the SPAWN command, the subprocess is created as a noninteractive process that exits upon encountering a severe error or an end-of-file indicator. At DCL level, CTRL/Z is treated as an end-of-file indicator.

10.2.1 Creating a Subprocess

In the following example, a user interrupts a command image (the TYPE command) with CTRL/Y, spawns a subprocess, exits from the subprocess, and returns to the original process:

```
$ TYPE MICE.TXT
Once the weather turns cold, mice may find a crack in your
foundation and enter your house. They're looking for food and
shelter from the harsh weather ahead.
```

```
.
.
.
CTRLV
$ SPAWN
%DCL-S-SPAWNED, process DOUGLASS_1 spawned
%DCL-S-ATTACHED, terminal now attached to process DOUGLASS_1
$ MAIL
MAIL>
.
.
.
MAIL> EXIT
$ LOGOUT
  Process DOUGLASS_1 logged out at 31-DEC-1988 12:42:12.46
%DCL-S-RETURNED, control returned to process DOUGLASS
$ CONTINUE
Once inside, they may gnaw through electrical wires and raid
your food. Because mice reproduce so quickly, what started
as one or two mice can quickly become an invasion. If you seal
the cracks and holes on the exterior of your foundation, you can
prevent these rodents from ever getting in.
```

Because each process you create is unique, commands executed in one process do not usually affect any other process. However, because control of the terminal passes between processes, commands that affect the terminal characteristics (for example, SET TERMINAL) affect any process controlling that terminal. For example, if one process inhibits echoing and exits without restoring it, echoing remains inhibited for the next process that gains control of the terminal. Reset any altered terminal characteristics with the SET TERMINAL command.

10.2.2 Exiting from a Subprocess

To exit from a subprocess created by SPAWN, use one of the following commands:

- **LOGOUT**—When you exit from a subprocess with the LOGOUT command, the subprocess is deleted (along with any subprocesses that it created), and you are returned to the parent process.
- **ATTACH**—When you exit from a subprocess with the ATTACH command, the subprocess hibernates, and control of your terminal is transferred to the specified process. (You must specify either a process name as a parameter to the ATTACH command or a process identification number (PID) as a value of the /IDENTIFIER qualifier of the ATTACH command.) The following example shows how to exit from the subprocess DOUGLASS_1 and attach to the process DOUGLASS:

10-6 Processes: Using the VMS Environment

```
$ ATTACH DOUGLASS

%DCL-S-RETURNED, control returned to process DOUGLASS

$ SHOW PROCESS

19-APR-1990 10:34:58.50   VTA303           User: DOUGLASS
Pid: 25C002B4   Proc. name: DOUGLASS       UIC: [200,200]
Priority: 4   Default file spec: SYS$SYSDEVICE:[DOUGLASS]

Devices allocated: $11$VTA303:
```

10.2.3 Looking at a Subprocess Context

By default, a subprocess inherits the following items from the parent process: defaults, privileges, symbols, logical names, control characters, message format, verification state, and key definitions. The environment that these items collectively create is called the **process context**. The following items, however, are not inherited from the parent process:

- Process identification number (PID)—The system assigns each created subprocess a unique process identification number.
- Process name—By default, the subprocess name consists of the name of the parent process followed by an underscore and an integer. Use the `/PROCESS` qualifier of the `SPAWN` command to specify a process name other than the default. A process name must be unique.
- Created commands—Commands that are defined by a parent process using the `SET COMMAND` command are not copied to a subprocess. To use a created command in a subprocess, you must use `SET COMMAND` to create that command for the subprocess.
- Authorize privileges—When you spawn to a subprocess, the process context contains the privileges of the parent process, not the privileges that you are authorized to enable. For example, if you spawn to a subprocess while in `MAIL` and want to perform a privileged operation, you need to have set the proper privilege in the parent process before you invoked `MAIL`.

You can use the following `SPAWN` qualifiers to prevent the subprocess from inheriting a number of these items:

Qualifier	Items Inhibited or Changed
<code>/CARRIAGE_CONTROL, /PROMPT</code>	DCL prompt
<code>/NOCLI</code>	CLI (command language interpreter; DCL by default)
<code>/NOKEYPAD</code>	Keypad definitions
<code>/NOLOGICAL_NAMES</code>	Logical names
<code>/NOSYMBOL</code>	Symbols

The `/SYMBOL` and `/LOGICAL_NAMES` qualifiers do not affect system-defined symbols (such as `$SEVERITY` and `$STATUS`) or system-defined logical names (such as `SYS$COMMAND` and `SYS$OUTPUT`). Symbols are described in Chapter 12, and logical names are described in Chapter 11.

Because copying logical names and symbols to a subprocess can be time-consuming (a few seconds), you may want to use the `/NOLOGICAL_NAMES` and `/NOSYMBOL` qualifiers to the `SPAWN` command unless you plan to use the logical names or symbols in the subprocess. If you use subprocesses frequently, the `ATTACH` command provides the most efficient way to enter and exit a subprocess. This method allows you to transfer control quickly between the parent process and subprocess rather than repeatedly waiting for the system to create a new subprocess for you.

10.3 Executing Programs Across the Network

Because of support provided by DECnet-VAX, programs can execute across the network as if they were executing locally. Because DECnet-VAX is integrated within the VMS operating system, it is easy to write programs that access remote files. To access a remote file in an application program, you need only include in your file specification the name of the remote node and any required access control information.

Task-to-task communications, a feature common to all DECnet implementations, allows two application programs running on the same or different operating systems to communicate with each other regardless of the programming languages used. Examples of network applications are distributed processing applications, transaction processing applications, and applications providing connection to servers.

10.4 Using Batch Jobs

A batch job is a noninteractive process. Because a batch job executes in a process of its own, you can have two or more processes doing different things at the same time. For example, you can perform a computer task interactively while the system executes a program or command procedure in batch mode.

The following sections describe how to use batch jobs to perform computing tasks.

10.4.1 Submitting a Batch Job

To run a job in batch mode, submit your job to a batch queue (a list of batch jobs waiting to execute) by entering the DCL command `SUBMIT`. When you submit a job, it is directed to the default batch queue, `SYS$BATCH`, where it is added to the end of the queue of jobs waiting to be executed. When the jobs preceding yours are completed, your job is executed. (On a VMS system, the number of batch jobs that can execute simultaneously is specified when the batch queue is created by the system manager.)

10-8 Processes: Using the VMS Environment

By default, the `SUBMIT` command uses a file type of `COM`. For example, the following command enters `JOB1.COM` into `SYS$BATCH`:

```
$ SUBMIT JOB1
Job JOB1 (queue SYS$BATCH, entry 651, started on SYS$BATCH)
```

The system displays the name of the job, the queue containing the job, and the entry number assigned to the job. You receive the `DCL` prompt once your job is submitted to the batch queue. If you need to reference your batch job in any `DCL` commands (`DELETE/ENTRY`, for example), do so by using the job entry number. (You can obtain the job entry number by using the `SHOW ENTRY` command.) Note that if multiple procedures are submitted in a batch job, the batch job terminates when any procedure exits with an error or fatal error status.

Your batch job does not necessarily have to start running at the time you submit it to the batch queue. To specify a different time, enter the `SUBMIT/AFTER` command. In the following example, the job is submitted after 11:30 p.m.:

```
$ SUBMIT/AFTER=23:30 JOB1.COM
```

10.4.2 Batch Job Output

By default, accumulated output from a batch job is written to a log file once each minute. (To specify a different time interval, include the `SET OUTPUT_RATE` command in your command procedure.) If you attempt to use the `EDT` editor to read the log file while the system is writing to it, you receive a message indicating that the file is locked by another user. Wait a few seconds and try again. The `EVE` editor, however, allows you to read the batch job's log file. By specifying `EDIT/TPU/READ_ONLY` and the name of the log file, you can use `EVE` commands to move around the log file and ensure that any changes you make to the file are not saved. If you omit the `/READ_ONLY` qualifier and modify the log file in any way, the batch job terminates.

Because your batch job is a process that logs in under your user name and executes your login command procedure, the output from a batch job includes the contents of your login command procedure. The output also includes everything written to the batch job log file (command procedure output, error messages, and so on) and the full logout message. To prevent your login command procedure from being written to the batch log file, add the following command to the beginning of your login command procedure:

```
$ IF F$MODE() .EQS. "BATCH" THEN SET NOVERIFY
```

By default, the log file name is the name under which you submitted the job. Also by default, the log file has a file type of `LOG` and assumes the device and directory specified by your login defaults. To specify a different log file name when you submit the job, use the `/LOG_NAME` qualifier to the `SUBMIT` command.

When the batch job completes, the log file is queued to the default system printer (`SYS$PRINT`), printed, and deleted. To save the log file after printing it, use the `/KEEP` qualifier to the `SUBMIT` command. To save the log file without printing it, use the `/NOPRINT` qualifier to the `SUBMIT` command.

10.4.3 Restarting a Batch Job

If the system fails while your batch job is executing, your job does not complete. When the system recovers and the queue is restarted, your job is aborted, and the next job in the queue is executed. However, by specifying the `/RESTART` qualifier when you submit a batch job, you indicate that the system should reexecute your job if the system crashes before the job is finished.

By default, a batch job is reexecuted beginning with the first line. See Chapter 13 for more information about symbols you can add to your command procedures to specify a different restarting point.

Chapter 11

Logical Names: Defining Names for Devices and Files

A logical name is a string of characters (for example, `WORK_DISK` or `PAY_FILE`) that is usually equated to a file name, device name, or other logical name. For example, when you equate the name `WORK_DISK` to a physical device `DRA1`, then `WORK_DISK` is a logical name and `DRA1` is an equivalence string.

Logical names can be defined by you or by the system. This chapter describes how you can create and use logical names to perform the following tasks:

- Reduce typing by using logical names as a short way of specifying files or directories you refer to frequently.
- Avoid confusion about the location of disk volumes.
- Keep your programs and command procedures independent of physical file specifications. (For example, if a command procedure references the logical name `ACCOUNTS`, you can equate `ACCOUNTS` to any file on any disk before executing the command procedure.)

This chapter also discusses the logical names created by the system.

11.1 Creating Logical Names

You can create your own logical names with either the `ASSIGN` or the `DEFINE` command. Usually, you define logical names in your login command procedure (`login.com`), so you can use the logical name whenever you are logged in. You can also create logical names interactively; however, you will be able to use these logical names only while your current process is active. Your system manager can also create logical names that can be used by anyone logged in to the system.

This section uses the `DEFINE` command to create logical names. (Note that the syntax for the `ASSIGN` command differs from the syntax for the `DEFINE` command. For information on using the `ASSIGN` command, see the Reference Section.

11-2 Logical Names: Defining Names for Devices and Files

The syntax for defining a logical name is as follows:

```
DEFINE logical-name equivalence-name[...]
```

For example, to associate the logical name *WORK_DISK* with the equivalence name *DRA1*, use the following command either at DCL level or in your login.com:

```
$ DEFINE WORK_DISK DRA1:
```

After you have defined this logical name, you can use the logical name (*WORK_DISK*) interchangeably with the equivalence name (*DRA1*).

11.1.1 Rules for Creating Logical Names

Observe the following rules when creating a logical name with the *DEFINE* command:

- A logical name and its equivalence name can each have a maximum of 255 characters. A logical name can contain alphanumeric characters, as well as the underscore (`_`), dollar sign (`$`), and hyphen (`-`).
- When specifying an equivalence name, you must include the punctuation marks (colons, brackets, periods) that would be required if it were part of a file specification. For example, a device name is terminated by a colon, a directory specification is enclosed in square brackets, and a file type is preceded by a period.
- You can optionally terminate a logical name with a colon. If you do this, the *ASSIGN* command removes the colon before placing the logical name in a logical name table. The *DEFINE* command does not remove the colon before placing the name in a logical name table.

In general, you should not specify a colon at the end of a logical name when you are creating it. However, if you do so and want to save the colon as part of the logical name, use the *DEFINE* command. (Note that when you delete a logical name ending with a colon, you need to specify two colons because the *DEASSIGN* command, like the *ASSIGN* command, removes one colon before it searches the logical name table for a match.)

If the logical name is part of a file specification, the logical name must be the leftmost component of the file specification and must be separated from the rest of the file specification by a colon. When you use a logical name to represent a complete file specification, the terminating colon is not needed. For example, the following commands display the file `DISK1:[SALES_STAFF]PAYROLL.DAT`:

```

$ DEFINE PAY_DISK1:[SALES_STAFF]PAYROLL.DAT
$ TYPE PAY

$ DEFINE PAY_FILE_DISK1:[SALES_STAFF]PAYROLL
$ TYPE PAY_FILE:.DAT

$ DEFINE PAY_DIR_DISK1:[SALES_STAFF]
$ TYPE PAY_DIR:PAYROLL.DAT

$ DEFINE PAY_DISK_DISK1:
$ TYPE PAY_DISK:[SALES_STAFF]PAYROLL.DAT

```

By default, the **DEFINE** command places logical names in your process logical name table, where the logical name is available only to your process and subprocesses. Section 11.4 describes logical name tables.

11.1.2 Equating More Than One Equivalence Name

You can equate more than one logical name with an equivalence name. For example, you can equate the logical names **\$TERMINAL** and **CONSOLE** to the physical name of a terminal so that both logical names translate to the same device. (If you equate a logical name to more than one equivalence string in a single command, you create a search list for the system to use to translate the names. See Section 11.8 for information about search list translation.)

If you equate a logical name to one equivalence string and then equate the same logical name to another equivalence string, the second definition supersedes the first. You can, however, equate the same logical name to different equivalence strings if the logical name definitions are in different tables (described in Section 11.4). You can equate the same logical name to different equivalence strings in the same table if they are defined in different access modes (described in Section 11.6).

If you cannot access a file and the command you are specifying and the file specification seem in order, check the left-hand component of the file specification (with **SHOW LOGICAL**) to be sure that it is not defined as a logical name.

11.2 Displaying Logical Names

You can show the equivalence name for a logical name with the **SHOW LOGICAL** command. For example, to display the equivalence name for the logical name **WORK_DISK**, enter the following command:

```
$ SHOW LOGICAL WORK_DISK
```

The system displays the following information:

```
"WORK_DISK" = "DRA1:" (LNM$PROCESS_TABLE)
```

11.3 Deleting Logical Names

To delete a logical name, use the DEASSIGN command. For example, to define the logical name STAFF to the subdirectory, [JONES.STAFF], enter the following command:

```
$ DEFINE STAFF [JONES.STAFF]
```

To delete this logical name, enter the following command:

```
$ DEASSIGN STAFF
```

Logical names in your process and job tables are automatically deleted when your process terminates. However, if you specify the /USER_MODE qualifier to the DEFINE command, you can place a logical name in the process logical name table and execute one command image before the logical name is deleted.

11.4 Understanding Logical Name Tables

The system stores logical names and their equivalence strings in tables called **logical name tables**. The system provides the following logical name tables:

- Process table
- Job table
- Group table
- The system table

Some logical name tables are available only to your process; these tables are called **process-private**. Other tables are **shareable**; that is, they are available to other users on the system.

When you enter a logical name as part of a command line, the system translates the logical name by searching the logical name tables in a certain order. Information about existing logical name tables and the order in which they are searched is stored in two **logical name directory tables**.

Identical logical names can exist in more than one table. The logical name that is used depends on the order in which the logical name tables are searched. For example, when the system attempts to translate a logical name in order to identify the location of a file, it uses the logical name LNM\$FILE_DEV to provide the list of tables in which to look for the name. The order in which the tables are listed is also the order in which they are searched. The precedence order defined by LNM\$FILE_DEV is (1) process table, (2) job table, (3) group table, and (4) system table. Therefore, if a logical name exists in both the process and the group logical name tables, the logical name within the process table is used. See Section 11.5.2 for more information about LNM\$FILE_DEV.

Within each table, the system defines some logical names for you. Each table and its system-defined logical names are described in the following sections.

11.4.1 The Process Table

Your process logical name table, named LNM\$PROCESS_TABLE, contains logical names that are available only to your process and any subsequent subprocesses. Use the logical name LNM\$PROCESS to refer to the process table.

Process logical names are recognized by the process they were created in and by any subsequent subprocesses. However, process logical names are not recognized by any parent process.

To display the logical names in your process table, use the following command:

```
$ SHOW LOGICAL/PROCESS
```

You can also specify the SHOW LOGICAL/TABLE=table_name command to display the contents of any logical name table.

By default, the DEFINE and DEASSIGN commands place names in and delete names from your process table.

Every process on the system has a process logical name table. When you log in, the system creates logical names for your process and places them in your process table. These names are listed in Table 11-1.

Table 11-1: Default Process Logical Names

Logical Name	Description
SYS\$COMMAND	The initial file (usually your terminal) from which DCL reads input. (A file from which DCL reads input is called an input stream .) The command interpreter uses SYS\$COMMAND to “remember” the original input stream.
SYS\$DISK	Default device established at login or changed by the SET DEFAULT command.
SYS\$ERROR	The default device or file to which DCL writes error messages generated by warnings, errors, and severe errors.
SYS\$INPUT	The default file from which DCL reads input.
SYS\$NET	The source process that invokes a target process in DECnet-VAX task-to-task communication. When opened by the target process, SYS\$NET represents the logical link over which that process can exchange data with its partner. SYS\$NET is defined only during task-to-task communication.
SYS\$OUTPUT	The default file (usually your terminal) to which DCL writes output. (A file to which DCL writes output is called an output stream .)
TT	Default device name for terminals.

Note that the logical names SYS\$INPUT, SYS\$OUTPUT, SYS\$ERROR, and SYS\$COMMAND refer to files that remain open for the life of the process.

11-6 Logical Names: Defining Names for Devices and Files

They are referred to as **process-permanent files**. For more information on process-permanent files, see Section 11.10.1.

11.4.2 The Job Table

Your job logical name table contains logical names that are available to all processes in your job tree, no matter what process or subprocess you are currently in. Your job table is named LNM\$JOB_xxx, where xxx is the Job Information Block address (defined by the system) for your job tree. Use the logical name LNM\$JOB to refer to your job table.

When you log in, the system creates certain logical names and places them in the job logical name table. These names are listed in Table 11-2. In addition, the logical names created for mounted disks and tapes and temporary mailboxes are also placed in the job logical name table.

Table 11-2: Default Job Logical Names

Logical Name	Description
SYS\$LOGIN	Your default device and directory when you log in.
SYS\$LOGIN_DEVICE	Your default device when you log in.
SYS\$REM_ID	For jobs initiated through a DECnet network connection, the identification of the process on the remote node from which the job was originated. On VMS operating systems, if proxy logins are enabled, this identification is the process's user name, or, if proxy logins are not enabled, this is the process identification number (PID). (Proxy logins to proxy accounts allow users to access files across the network without specifying an access control string.)
SYS\$REM_NODE	For jobs initiated through a DECnet network connection, the name of the remote node from which the job was originated.
SYS\$SCRATCH	Default device and directory to which temporary files are written.

There is one job table for each job tree in the system. All job tables are shareable so that all users may access them. However, to access a job logical name table other than your own, you must redefine LNM\$JOB in your process directory logical name table. For more information about LNM\$JOB, see Section 11.5.

11.4.3 The Group Table

The group logical name table contains logical names that are available to all users with the same user identification code (UIC) group number. The group table is named LNM\$GROUP_xxx, where xxx represents your UIC group number. Use the logical name LNM\$GROUP to refer to your group table. Every group on the system has a corresponding group logical name table.

To create or delete a name in your group table, you need GRPNAM, GRPPRV, or SYSPRV privilege. See the *VMS System Manager's Manual* for a description of user privileges.

11.4.4 The System Table

The system logical name table contains logical names that are available to all users on the system. The system table is named LNM\$SYSTEM_TABLE; use the logical name LNM\$SYSTEM to refer to it. To create or delete a name in the system table, you must have a UIC group number between 0 and 10, or SYSNAM or SYSPRV privilege.

There is only one system logical name table for the system. It contains the names shown in Table 11-3.

Table 11-3: Default System Logical Names

Logical Name	Description
DBG\$INPUT	Default input stream for the VMS Debugger; equated to SYS\$INPUT
DBG\$OUTPUT	Default output stream for the VMS Debugger; equated to SYS\$OUTPUT
SYS\$COMMON	Device and directory name for the common part of SYS\$SYSROOT
SYS\$ERRORLOG	Device and directory name of error log data files
SYS\$EXAMPLES	Device and directory name of system examples
SYS\$HELP	Device and directory name of system HELP files
SYS\$INSTRUCTION	Device and directory name of system instruction data files
SYS\$LIBRARY	Device and directory name of system libraries
SYS\$LOADABLE_IMAGES	Device and directory of operating system executive loadable images, device drivers, and other executive loaded code
SYS\$MAINTENANCE	Device and directory name of system maintenance files
SYS\$MANAGER	Device and directory name of system manager files
SYS\$MESSAGE	Device and directory name of system message files
SYS\$NODE	Network node name for the local system if DECnet-VAX is active on the system
SYS\$SHARE	Device and directory name of system shareable images
SYS\$SPECIFIC	Device and directory name for node-specific part of SYS\$SYSDEVICE
SYS\$STARTUP	Device and directory name of system startup files
SYS\$SYSDEVICE	VMS system disk containing system directories

(continued on next page)

Table 11-3 (Cont.): Default System Logical Names

Logical Name	Description
SYS\$SYSROOT	Device and root directory for system directories
SYS\$SYSTEM	Device and directory of operating system programs and procedures
SYS\$TEST	Device and directory name of User Environment Test Package (UETP) files
SYS\$UPDATE	Device and directory name of system update files

11.5 Directory Logical Name Tables

The system provides the following two directory tables to catalog your logical name tables:

- LNM\$PROCESS_DIRECTORY catalogs your process tables (LNM\$PROCESS and LNM\$JOB).
- LNM\$SYSTEM_DIRECTORY catalogs your shareable tables (LNM\$GROUP and LNM\$SYSTEM).

Both of these directories contain logical names that translate iteratively to table names. The name of a logical name table must be recorded in one of these directory tables in order for the system to find it.

To see the relationship of directory tables to logical name tables enter the SHOW LOGICAL/STRUCTURE command, as shown in the following example:

```
$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
  (LNM$PROCESS_TABLE)
(LNM$SYSTEM_DIRECTORY)
  (LNM$GROUP_000360)
  (LNM$JOB_806E98E0)
  (LNM$SYSTEM_TABLE)
```

11.5.1 The Process Directory Table

Each process on the system has its own process directory logical name table. When you log in, the VMS operating system places certain logical names in your process directory table. These names are listed in Table 11-4.

Table 11-4: Default Process Directory Logical Names

Logical Name	Description
LNМ\$GROUP	A logical name that is defined as LNМ\$GROUP_xxx, where xxx represents your group number. LNМ\$GROUP_xxx is the logical name table used by your UIC group. (The table LNМ\$GROUP_xxx is cataloged in the system directory table.) Therefore, LNМ\$GROUP is a logical name that translates iteratively to the name of your group logical name table.
LNМ\$JOB	A logical name that is defined as LNМ\$\$JOB_xxx, where xxx represents a number unique to your job tree. LNМ\$JOB_xxx is the logical name table used by your job. (The table LNМ\$JOB_xxx is cataloged in the system directory table.) Therefore, LNМ\$JOB is a logical name that translates iteratively to the name of your job logical name table.
LNМ\$PROCESS	A logical name that translates iteratively to LNМ\$PROCESS_TABLE, which is the name of your process logical name table.
LNМ\$PROCESS_DIRECTORY	The name of your process directory logical name table.
LNМ\$PROCESS_TABLE	The name of your process logical name table.

11.5.2 The System Directory Table

There is one system directory logical name table. The VMS operating system places certain logical names in the system directory table. These names are listed in Table 11-5.

Table 11-5: Default System Directory Logical Names

Logical Name	Description
LNМ\$DCL_LOGICAL	A logical name that is defined as LNМ\$FILE_DEV. This logical name iteratively translates into the list of logical name tables searched and displayed by the SHOW LOGICAL and SHOW TRANSLATION commands and the F\$TRNLNM lexical function. By default, these commands search and display the process, job, group, and system logical name tables, in that order.
LNМ\$DIRECTORIES	A logical name that is defined as LNМ\$PROCESS_DIRECTORY and LNМ\$SYSTEM_DIRECTORY.

(continued on next page)

Table 11-5 (Cont.): Default System Directory Logical Names

Logical Name	Description
LNМ\$FILE_DEV	A logical name that is defined as the list of logical name tables searched by the system when processing a file specification. By default, it is defined as LNМ\$PROCESS, LNМ\$JOB, LNМ\$GROUP, and LNМ\$SYSTEM. This means that the process, job, group, and system logical name tables are searched, in that order.
LNМ\$GROUP_XXX	The name of a group logical name table, where xxx is a group number. There is an LNМ\$GROUP_XXX logical name table for each group in the system.
LNМ\$JOB_XXX	The name of a job logical name table, where xxx is a number unique to this job tree. There is an LNМ\$JOB_XXX logical name table for each job in the system.
LNМ\$SYSTEM	A logical name that translates iteratively to LNМ\$SYSTEM_TABLE, which is the name of the system logical name table.
LNМ\$SYSTEM_DIRECTORY	The name of the system directory logical name table.
LNМ\$SYSTEM_TABLE	The name of the system logical name table.

Generally, you do not need to change the default logical name table definitions set up in the directory tables, LNМ\$PROCESS_DIRECTORY and LNМ\$SYSTEM_DIRECTORY. Two reasons for changing the entries in the directory tables are (1) to create another logical name table, and (2) to change the search order for file specification logical names by redefining LNМ\$FILE_DEV. See Section 11.7 for information about creating your own logical name table and changing the order in which the system searches the logical name tables.

Multiple tables with the same name may exist. For example, there may exist both a process-private and a shareable table called MY_TABLE. The process-private version always takes precedence over the shareable table in all logical name table processing. When a logical name, such as LNМ\$FILE_DEV, is used as a table name, the logical name is iteratively translated until a list of table names is formed. During this iterative translation, each name is first translated in the process directory. If this translation fails, it is then translated in the system directory. This order of precedence cannot be changed. As a consequence of this ordering, a logical name placed in the process directory table for use as a table name will always take precedence over any identical name residing in the system directory.

11.6 Logical Name Access Modes

The four access modes in the VMS operating system are as follows:

- User-mode (the outermost and least privileged mode)
- Supervisor-mode
- Executive-mode
- Kernel-mode (the innermost and most privileged mode)

When you create a logical name with DCL commands, it has an access mode of user, supervisor, or executive. By default, logical names are created in supervisor mode; you must have SYSNAM privilege to create an executive mode logical name. To see the access mode for a logical name, use the SHOW LOGICAL/FULL command, as follows:

```
$ SHOW LOGICAL/FULL PROJECT
"PROJECT" [super] = "DISK1:[JONES]" (LNM$PROCESS_TABLE)
```

This shows that the logical name PROJECT was created in supervisor mode.

You can equate the same logical name to different equivalence strings in the same logical name table by specifying different access modes for each definition. The following example equates the logical name ACCOUNTS to two different equivalence names in the process logical name table—one in supervisor-mode and one in executive-mode:

```
$ DEFINE ACCOUNTS DISK1:[ACCOUNTS]CURRENT.DAT
$ DEFINE/EXECUTIVE_MODE ACCOUNTS DISK1:[JANE.ACCOUNTS]OBSOLETE.DAT
```

Logical names created in user mode are temporary. Define a logical name in user mode when you want to define it only for the execution of the next image. In the following example, the logical name ADDRESSES is automatically deleted after the execution of the program PAYABLE:

```
$ DEFINE/USER_MODE ADDRESSES DISK1:[SAM.ACCOUNTS]OVERDUE.LIS
$ RUN PAYABLE
```

In looking up logical names, all privileged images and utilities, such as LOGINOUT and MAIL, bypass the user- and supervisor-mode portions of the system logical name table. Therefore, DIGITAL recommends that logical names for important system components (public disks and directories, for example) be defined in executive mode, using the DCL command DEFINE/SYSTEM/EXECUTIVE. (Only the operating system and privileged programs can create logical names in kernel-mode.) This operation requires either the SYSPRV or SYSNAM privilege.

11.7 Creating a Logical Name Table

The `CREATE/NAME_TABLE` command creates a logical name table and catalogs it in one of the directory logical name tables. (Logical names that identify logical name tables or that translate iteratively to logical name tables must always be entered into one of the directory logical name tables.) To create a logical name table that is private to your process, create the table in `LNМ$PROCESS_DIRECTORY` (the default). If you want the table to be shareable, specify `/PARENT_TABLE=LNМ$SYSTEM_DIRECTORY` with the `CREATE/NAME_TABLE` command. Creating shareable name tables requires `SYSPRV` privilege or `ENABLE` access to the parent table.

The following example creates a process-private logical name table named `TAX`, places the definition for the logical name `CREDIT` in the table, and verifies the table's creation. (You must specify the `/TABLE` qualifier with the `SHOW LOGICAL` command to display a logical name in any table other than `LNМ$SYSTEM` or `LNМ$PROCESS`.)

```
$ CREATE/NAME_TABLE TAX
$ DEFINE/TABLE=TAX CREDIT [ACCOUNTS.CURRENT]CREDIT.DAT
$ SHOW LOGICAL/TABLE=TAX CREDIT

"CREDIT" = "[ACCOUNTS.CURRENT]CREDIT.DAT" (TAX)
```

To make the system search a user-created logical name table automatically when processing file specifications, you must create a process-private version of the default search list (`LNМ$FILE_DEV`) in `LNМ$PROCESS_DIRECTORY`. To add the created table's name to the default search list, you can define `LNМ$FILE_DEV` as follows:

```
$ DEFINE/TABLE=LNМ$PROCESS_DIRECTORY LNМ$FILE_DEV -
_$ TAX, LNМ$PROCESS, LNМ$JOB, LNМ$GROUP, LNМ$SYSTEM
```

Placing the table's name first specifies that the system search that table first, and so on in the order of specification.

To delete a logical name table, specify the table that contains it (the system or process directory logical name table) and the name of the table. Deleting a shareable logical name table requires `DELETE` access to the table or `SYSPRV` privilege. For example, to delete the logical name table `TAX` of the preceding example, specify the following command line:

```
$ DEASSIGN/TABLE=LNМ$PROCESS_DIRECTORY TAX
```

Note that all logical names in descendant tables (and the descendant tables themselves) are deleted when a parent logical name table is deleted.

11.8 Using Search Lists

A search list is a logical name that has more than one equivalence name. You can use a search list in any place you can use a logical name. For example:

```
$ DEFINE GETTYSBURG [JONES.HISTORY],[JONES.WORKFILES]
$ SHOW LOGICAL GETTYSBURG

"GETTYSBURG" = "[JONES.HISTORY]" (LNM$PROCESS_TABLE)
              = "[JONES.WORKFILES]"
```

The logical name GETTYSBURG is a search list because it has more than one equivalence name.

When you use a logical name that is a search list, the system translates the logical name until it finds a match. The order in which you specify the equivalence strings determines the order in which the system translates the names. It uses each equivalence name listed in the definition until a match is found.

A search list is not a wildcard. It is a list of places to look. Once a file is found, the search is ended. For example:

```
$ TYPE GETTYSBURG:SPEECH.TXT

DISK1:[JONES.HISTORY]SPEECH.TXT;2

Fourscore and seven years ago, our fathers brought forth on
this continent a new nation, conceived in liberty, and
dedicated to the proposition that all men are created equal.
.
.
.
```

In the previous example, the TYPE command searches the equivalence names [JONES.HISTORY] and [JONES.WORKFILES] in the order they were listed when GETTYSBURG was defined. Once it finds a file named SPEECH.TXT, the search is halted and the file is displayed.

You can use a search list with a command that accepts wildcards. When you use wildcards, the system forms file specifications using each equivalence name in the search list. The command operates on each file specification that identifies an existing file.

For example, if you specify the DIRECTORY command with a wildcard character in the version field, it finds all versions of SPEECH.TXT in the search list defined by GETTYSBURG, as shown in the following example:

```
$ DIRECTORY GETTYSBURG:SPEECH.TXT;*

Directory DISK1:[JONES.HISTORY]
SPEECH.TXT;2      SPEECH.TXT;1

Total of 2 files.
```

11-14 Logical Names: Defining Names for Devices and Files

```
Directory DISK1:[JONES.WORKFILES]
SPEECH.TXT;1
Total of 1 file.
Grand total of 2 directories, 3 files.
```

The **DIRECTORY** command searches the equivalence names [JONES.HISTORY] and [JONES.WORKFILES] in the order they were listed when **GETTYSBURG** was defined. It finds a file named **SPEECH.TXT** in each directory. If **SPEECH.TXT** exists in only one of the directories, only one directory listing is displayed. If **SPEECH.TXT** does not exist in either directory, an error message is displayed indicating that the file was not found.

When you use a search list with a command that does not accept wildcards in a file specification, the system forms a file specification using each equivalence name in the search list until a file specification for an existing file is found. The command affects only the first file found. For example:

```
$ DEFINE DECEMBER DISK1:[FRED],WORK2:[BARNEY]
$ EDIT/EDT DECEMBER:QUOTAS.TXT
```

First, the system forms the file specification **DISK1:[FRED]QUOTAS.TXT** and searches for that file. If **QUOTAS.TXT** is found in **DISK1:[FRED]**, it is opened for editing. No other files are subsequently opened. If **QUOTAS.TXT** is not found in **DISK1:[FRED]**, the system searches for it in **WORK2:[BARNEY]**. If **QUOTAS.TXT** is found there, it is opened. If it is not found, an error message is displayed. The system displays an error message only after it checks all equivalence names in a search list. Then the system reports an error only on the last file it attempted to find.

The **RUN** command is an exception. When the **RUN** command is followed by a search list, the system forms file specifications as described previously. However, the system then checks to see whether any of the files in the list are installed images. It runs the first file in the search list that is an installed image. Then the **RUN** command terminates.

If none of the file specifications are installed images, the system repeats the process of forming file specifications. This time it looks for each file specification on the disk. It runs the first file it finds there. An error message is displayed if none of the specified files is found in either the known file list or on the disk.

11.9 Using Logical Node Names

A logical node name is a special type of logical name that can be used in place of a network node name or in place of a node name and an access control string. For example:

```
$ DEFINE BOS "BOSTON""ADAMS JOHN"::"
```

The logical name **BOS** is equated to the node name **BOSTON** and an access control string, where **ADAMS** is the user name and **JOHN** is the password.

Use the logical name BOS to avoid typing (and displaying) your user name and password on the terminal screen.

NOTE: Do not place a DEFINE command that includes a password in a file (your login command procedure, for example). If others read the file, they will see the password.

11.10 System-Created Logical Names

The system creates a number of logical names for you when you start the system and log in. By default, DCL creates and assigns logical names to four process-permanent files. When you redefine these logical names, only your process is affected. The system defines other logical names that you can reassign only with special privileges.

11.10.1 Process-Permanent Logical Names

Process-permanent logical names are created by DCL when you log in and remain defined for the life of your process. You cannot deassign these logical names. You can redefine them (by specifying the same name in a DEFINE command), but if the redefined name is later deassigned, the process-permanent name is reestablished. These process-permanent logical names, as follows, are available to each user of the system at the process level:

- **SYS\$INPUT**—Logical name that refers to the default input device or file
- **SYS\$OUTPUT**—Logical name that refers to the default output device or file
- **SYS\$ERROR**—Logical name that refers to the default device or file to which the system writes messages
- **SYS\$COMMAND**—Logical name that refers to the value of **SYS\$INPUT** when you log in

Table 11-6 shows what these logical names are equated to by default.

Table 11-6: Equivalence Names for Process-Permanent Logical Names

Logical Name	Interactive Mode	Batch Mode	Command Procedure
SYS\$COMMAND	Terminal ¹	Disk ²	Terminal
SYS\$INPUT	Terminal	Disk	Disk

¹Device name of your terminal

²Device name of the initial default device

Table 11-6 (Cont.): Equivalence Names for Process-Permanent Logical Names

Logical Name	Interactive Mode	Batch Mode	Command Procedure
SYS\$ERROR	Terminal	Log file ³	Terminal
SYS\$OUTPUT	Terminal	Log file	Terminal

³Batch job log file

The following sections describe how to use process-permanent logical names as file specifications.

11.10.1.1 Redefining SYS\$INPUT

You can redefine SYS\$INPUT so that a command procedure reads input from the terminal or another file. For example, to edit a file from a command procedure, include the following lines in the command procedure:

```
$ DEFINE/USER_MODE SYS$INPUT SYS$COMMAND
$ EDIT/TPU MYFILE.DAT
.
.
.
```

In the previous example, SYS\$INPUT is redefined as SYS\$COMMAND so that the editor obtains input from the terminal rather than from the command procedure file (the default). SYS\$COMMAND refers to the terminal, the initial input stream when you logged in. The /USER_MODE qualifier tells the command procedure that SYS\$INPUT is redefined only for the duration of the next image. In this example, the next image is the editor. When the editor is finished, SYS\$INPUT resumes its default value; in this case, the default value is the command procedure file.

Note that if you redefine SYS\$INPUT, DCL ignores your definition. DCL always obtains input from the default input stream. However, images, such as command procedures, can use your definition for SYS\$INPUT.

11.10.1.2 Redefining SYS\$OUTPUT

You can redefine SYS\$OUTPUT to redirect output from your default device to another file. When you redefine SYS\$OUTPUT, the system opens a file with the name you specify in the logical name assignment. When you define SYS\$OUTPUT, all subsequent output is directed to the new file.

In the following example, SYS\$OUTPUT is defined as MYFILE.LIS before the SHOW DEVICES command is entered. The display produced by SHOW DEVICES is directed to MYFILE.LIS in your current directory rather than to your terminal. You can manipulate this data as you would any other text file.

```
$ DEFINE SYS$OUTPUT MYFILE.LIS
$ SHOW DEVICES
```


Remember to deassign `SYS$OUTPUT`, or output will continue to be written to the file you specify. Note that you can redefine `SYS$OUTPUT` in user mode (with `DEFINE/USER_MODE`) to redirect output from an image. This definition is in effect only until the next command image is executed. Once the command image is executed (that is, the output is captured in a file), the logical name `SYS$OUTPUT` resumes its default value.

When you log in, the system creates two logical names called `SYS$OUTPUT`. One name is created in executive mode; the other name is created in supervisor mode. You can supersede the supervisor mode logical name by redefining `SYS$OUTPUT`. If you deassign the supervisor mode name, the system redefines `SYS$OUTPUT` in supervisor mode, using the executive mode equivalence name. You cannot deassign the executive mode name.

In the following example, `SYS$OUTPUT` is redefined to the file `TEMP.DAT`. When `SYS$OUTPUT` is redefined, output from `DCL` and from images is directed to the file `TEMP.DAT`. The output from the `SHOW LOGICAL` command and from the `SHOW TIME` command is also sent to `TEMP.DAT`. When you deassign `SYS$OUTPUT`, the system closes the file `TEMP.DAT` and redefines `SYS$OUTPUT` to your terminal. When you enter the `TYPE` command, the output collected in `TEMP.DAT` is displayed on your terminal.

```
$ DEFINE SYS$OUTPUT TEMP.DAT
$ SHOW LOGICAL SYS$OUTPUT
$ SHOW TIME
$ DEASSIGN SYS$OUTPUT
$ TYPE TEMP.DAT
  "SYS$OUTPUT" = "DISK1:" (LNM$PROCESS_TABLE)
31-DEC-JAN-1988 13:26:53
```

When you redefine `SYS$OUTPUT` to a file, the logical name contains only the device portion of the file specification, even though the output is directed to the file you specify. In the previous example, when `SYS$OUTPUT` was redefined, the equivalence name contained the device name `DISK1`, not the full file specification.

If the system cannot open the file you specify when you redefine `SYS$OUTPUT`, an error message is displayed.

After you redefine `SYS$OUTPUT`, most commands direct output to the existing version of the file. However, certain commands create a new version of the file before they write output.

11.10.1.3 Redefining `SYS$ERROR`

You can redefine `SYS$ERROR` to direct error messages to a specified file. However, if you redefine `SYS$ERROR` so it is different from `SYS$OUTPUT` (or if you redefine `SYS$OUTPUT` without also redefining `SYS$ERROR`), `DCL` commands send informational, warning, error, and severe error messages to both `SYS$ERROR` and `SYS$OUTPUT`. Therefore, you receive these messages twice—once in the file indicated by the definition of `SYS$ERROR` and once in the file indicated by `SYS$OUTPUT`. Success messages are sent only to the file indicated by `SYS$OUTPUT`.

11-18 Logical Names: Defining Names for Devices and Files

If you redefine `SYS$ERROR` and then run an image that references `SYS$ERROR`, the image sends error messages only to the file indicated by `SYS$ERROR` even if `SYS$ERROR` is different from `SYS$OUTPUT`. Only DCL commands and images using standard VMS error display mechanisms send error messages to both `SYS$ERROR` and `SYS$OUTPUT` when these files are different.

11.10.1.4 Redefining `SYS$COMMAND`

Although you can redefine `SYS$COMMAND`, DCL ignores your definition. DCL always uses the default definition for your initial input stream. However, if you execute an image that references `SYS$COMMAND`, the image can use your new definition.

11.10.2 System-Permanent Logical Names

The following table lists the logical names automatically defined when the system starts up. These names are available to all users of the system at the system level.

Logical Name	Equivalence Name
<code>DBG\$INPUT</code>	<code>SYS\$INPUT</code> at the process level
<code>DBG\$OUTPUT</code>	<code>SYS\$OUTPUT</code> at the process level
<code>SYS\$COMMON</code>	<code>SYS\$SYSDEVICE:[SYS<i>n</i>.SYSCOMMON.]</code> , where <i>n</i> is the root directory number of your processor
<code>SYS\$ERRORLOG</code>	<code>SYS\$SYSROOT:[SYSERR]</code>
<code>SYS\$EXAMPLES</code>	<code>SYS\$SYSROOT:[SYSHLP.EXAMPLES]</code>
<code>SYS\$HELP</code>	<code>SYS\$SYSROOT:[SYSHLP]</code>
<code>SYS\$INSTRUCTION</code>	<code>SYS\$SYSROOT:[SYSCBI]</code>
<code>SYS\$LIBRARY</code>	<code>SYS\$SYSROOT:[SYSLIB]</code>
<code>SYS\$LOADABLE_IMAGES</code>	<code>SYS\$SYSROOT:[SYS\$LDR]</code>
<code>SYS\$MAINTENANCE</code>	<code>SYS\$SYSROOT:[SYSMAINT]</code>
<code>SYS\$MANAGER</code>	<code>SYS\$SYSROOT:[SYSMGR]</code>
<code>SYS\$MESSAGE</code>	<code>SYS\$SYSROOT:[SYSMSG]</code>
<code>SYS\$NODE</code>	Name of your node if you are on a network
<code>SYS\$SHARE</code>	<code>SYS\$SYSROOT:[SYSLIB]</code>
<code>SYS\$SPECIFIC</code>	<code>SYS\$SYSDEVICE:[SYS<i>n</i>.]</code> , where <i>n</i> is the root directory number of your processor
<code>SYS\$STARTUP</code>	As a search list, points first to <code>SYS\$SYSROOT:[SYS\$STARTUP]</code> , then to <code>SYS\$MANAGER</code>
<code>SYS\$SYSDEVICE</code>	System disk (usually <code>SYS\$DISK</code>)

Logical Name	Equivalence Name
SYS\$SYSROOT	As a search list, points first to SYS\$SYSDEVICE:[SYS <i>n</i> .], where <i>n</i> is the root directory number of your processor; then to SYS\$COMMON
SYS\$SYSTEM	SYS\$SYSROOT:[SYSEXE]
SYS\$TEST	SYS\$SYSROOT:[SYSTEST]
SYS\$UPDATE	SYS\$SYSROOT:[SYSUPD]



Chapter 12

Symbols: Defining Commands and Expressions

Symbols are similar to logical names, because they equate a character-string expression to another expression. Whereas logical names were used to represent devices, files, or another logical names, **symbols** can represent DCL commands (for example, "MAIL"), character or numeric values (for example, "17" or "DOG"), or a logical value (such as "TRUE"). Symbols are useful shortcuts for entering DCL-level commands that you frequently use, and they can be essential aids in representing data in command procedures.

This chapter describes how to use symbols. Read this chapter to learn how to do the following:

- Define a symbol to represent a DCL-level command
- Use symbols to collect, store, and manipulate data

12.1 Using Symbols to Represent DCL Commands

You can define a symbol to represent a DCL command either in your login command file (LOGIN.COM) or interactively, at DCL level. Usually, it is a good idea to define symbols for frequently-used commands in your login command file. When you define the symbol in your login command file, you can use the symbol each time that you log in; when you define the symbol interactively, the symbol can be used only during the current process.

To define a symbol to represent a DCL command, use the following syntax:

```
$ symbol_name ::= DCL_command_line
```

For example, suppose you often use the DIRECTORY command with the qualifiers /NOTRAILING and /COLUMN=2. You could include the following line in your login command file, allowing you to enter this command just by entering the two-character symbol DI:

```
$ DI ::= DIRECTORY /NOTRAILING /COLUMN=2
```

12-2 Symbols: Defining Commands and Expressions

You can also use symbols to enter DCL command lines that execute images or command procedures.

The following sample section of a login command file gives some examples of using symbols for DCL command line substitution. Descriptions of the individual symbols follows the sample code section:

```
$ DEL      :==  DELETE /LOG /CONFIRM①
$ ED       :==  EDIT /TPU②
$ HOST     :==  SET HOST③
$ HR       :==  SET HOST RED④
$ M*ALL    :==  MAIL /EDIT=(SEND,REPLY)⑤
$ PROT     :==  SET PROT=(O:RWED,G:RWE,W:RWE) /LOG⑥
$ TIME     :==  @[JONES]TIME⑦
$ PRINTALL :==  $[ACCOUNTS]PRINTALL⑧
```

- ① Defines the symbol DEL to represent the command line DELETE /LOG /CONFIRM. When this line is in your login command file and you enter the command DEL, it is equivalent to entering the complete command DELETE /LOG /CONFIRM. For example, the following sequence could take place:

```
$ DEL OLD_ACCOUNTS.TXT;1
DELETE SYS$SYSDEVICE:[JONES]OLD_ACCOUNTS.TXT;1 ? [N]: Y
%DELETE-I-FILDEL, SYS$SYSDEVICE:[JONES]OLD_ACCOUNTS.TXT;1 deleted
(20 blocks)
```

- ② Defines the symbol ED to be equivalent to the DCL command EDIT /TPU. For example, if you have defined this symbol, you would edit the file MANAGERS.DIS with the TPU (EVE) editor by entering the following command:

```
$ ED MANAGERS.DIS
```
- ③ Defines the symbol HOST to be equivalent to the DCL command SET HOST. As with the SET HOST command, you must supply the name of the node to which you want to set host.
- ④ Defines the symbol HR to be equivalent to the DCL command line SET HOST RED. In this case, the argument to the SET HOST command is supplied as part of the symbol definition.
- ⑤ Defines a symbol to be equivalent to the DCL-level MAIL command. In this symbol, the asterisk (*) is used to indicate that any of the following symbols are equivalent to the command MAIL:

```
M
MA
MAI
MAIL
```
- ⑥ Defines the symbol TIME to be equivalent to the DCL command that executes the command procedure [JONES]TIME.COM.

- ⑦ Defines the symbol PRINTALL to execute the image WORK_DISK:[ACCOUNTS]PRINTALL.EXE. Note that when the dollar sign (\$) precedes a file specification at the beginning of a symbol definition (without any space between the dollar sign and the file specification), then the dollar sign has the meaning of "RUN".

TIP: When using symbols to represent DCL commands or collect, store, and manipulate data you can use several DCL commands. Two of the commands you can use are SHOW SYMBOL and DELETE SYMBOL.

The SHOW SYMBOL command displays symbol values. Specify the name of the symbol to display the value of a particular local symbol. Specify the name of the symbol and /GLOBAL to display the value of a particular global symbol. Specify /ALL to display all local symbols and /ALL/GLOBAL to display all global symbols.

The DELETE/SYMBOL command deletes a symbol. You must include the /GLOBAL qualifier to delete a global symbol. In the following example, the global symbol TEMP is deleted:

```
$ DELETE/SYMBOL/GLOBAL TEMP
```

12.2 Using Symbols to Collect, Store, and Manipulate Data

You can use symbols to store and manipulate a variety of values. This section describes the values that can be stored in symbols. It also describes how symbols can be combined in expressions to manipulate the values the symbols contain.

12.2.1 Defining Symbols as Character Strings

Defining a symbol as a character string allows you to insert that string in a command line by typing the symbol (with surrounding apostrophes to force substitution, as described in Section 12.2.4). In the following example, the symbol FILE is first defined as a complete file specification and then used in the TYPE command:

```
$ FILE == "DISK1:[JONES.TAXES]CORPORATE.DAT"
$ TYPE 'FILE'
```

The string can be a directory you often access. In the following example, whenever the symbol TAXES occurs in a command line, the literal value replaces the symbol before the line is executed.

```
$ TAXES == "[JONES.TAXES]"
$ COPY 'TAXES'OVERDUE.DAT OVERDUE.TMP
```

12-4 Symbols: Defining Commands and Expressions

Symbols can also be variables, which hold values that you calculate or that you assign as something other than a literal. For example, you might assign the value of a lexical function to a variable or read the value of a file record into a variable. As variables, symbols are most often used in command procedures (see Chapter 13).

12.2.2 Creating Symbols

To create a symbol, assign a value to a symbolic name using the following format:

```
symbol-name =[=] value
```

The symbol name can be 1 through 255 characters long and must begin with a letter, an underscore (`_`), or a dollar sign (`$`). In a symbol name, both lowercase and uppercase letters are treated as uppercase.

The value you assign to a symbol can be made either locally or globally available to the command interpreter:

- **Local**—A local symbol is available to the command level that defined it, to any command procedure executed from that level, and to lower command levels. (By convention, DCL level—command level 0—is the highest command level and command level 31 the lowest command level. Thus, when you move from command level 3 to command level 2, you are moving to the next higher command level. If you execute a command procedure interactively, the commands in the procedure are executed at command level 1. You can create a maximum of 31 command levels.)
- **Global**—A global symbol is available to any command level regardless of the level at which it was defined.

To create a local symbol, use a single equal sign in the assignment statement. To create a global symbol, use two equal signs. The following commands define the local symbol `FILE` as the character string `DISK2:[BOLIVAR]PRICES.CUR` and the global symbol `MAX_VALUE` as the number 24.

```
$ FILE = "DISK2:[BOLIVAR]PRICES.CUR"  
$ MAX_VALUE == 24
```

You can omit the quotation marks around character strings in assignment statements if you precede the equal sign or signs with a colon. Symbol assignments that omit quotation marks automatically change the character string to uppercase letters and compress multiple spaces and tabs to a single space. The following example again creates the local symbol `FILE`, this time omitting the quotation marks because of the included colon:

```
$ FILE := ACCOUNTS:[BOLIVAR]PRICES.84
```


The result of DCL's evaluation of a symbol is either a character string or an integer value. The data type (character or integer) of a symbol is determined by the data type of the value currently assigned. The type is not permanent: if the value changes type, as in the following example, the symbol changes type. In the following example, the local symbol NUM is first assigned a character value and then converted to an integer value when assigned an integer expression:

```
$ NUM = "ABC"
$ NUM = 2 + 5
```

Local symbols take precedence over global symbols with the same name. Symbols take precedence over identical command names. This means that if you define a symbol with the same name as a DCL command, your definition overrides the command name. For example, if you define the symbol HELP as the command TYPE HELP.LST, you can no longer invoke the system's HELP facility by typing HELP.

12.2.3 Understanding Symbol Tables

Symbols are stored in the following tables, which are maintained by the operating system:

- Local symbol table—DCL maintains a local symbol table for your main process and for every command level that you create when you execute a command procedure, use the CALL command, or submit a batch job. A local table is deleted when its associated command level terminates. (See Chapter 10 for more information about processes, command procedures, and batch jobs.)

In addition to the local symbols you create, a local symbol table contains eight symbols that are maintained by DCL. These symbols, named P1, P2, and so on through P8, are used for passing parameters to a command procedure. Parameters passed to a command procedure are regarded as character strings. Otherwise, P1 through P8 are defined as null character strings (""). They are stored in the local symbol table.

- Global symbol table—DCL maintains only one global symbol table for the duration of a process. In addition to the global symbols you create, the global symbol table contains the reserved global symbols described in the following table. These global symbols give you status information on your programs and command procedures as well as on system commands and utilities.

Reserved Global Symbols	Definition										
\$STATUS	The condition code returned by the most recently executed command. \$STATUS conforms to the format of a VMS message code. Applications programs can set the value of the global symbol \$STATUS by including a parameter value to the EXIT command. The system uses the value of \$STATUS to determine which message, if any, to display and whether to continue execution at the next-higher command level. The value of the lower three bits in \$STATUS is placed in the global symbol \$SEVERITY.										
\$SEVERITY	The severity level of the condition code returned by the most recently executed command. \$SEVERITY, which is equal to the lower three bits of \$STATUS, can have the following values: <table style="margin-left: 2em;"> <tr><td>0</td><td>Warning</td></tr> <tr><td>1</td><td>Success</td></tr> <tr><td>2</td><td>Error</td></tr> <tr><td>3</td><td>Information</td></tr> <tr><td>4</td><td>Severe (fatal) error</td></tr> </table>	0	Warning	1	Success	2	Error	3	Information	4	Severe (fatal) error
0	Warning										
1	Success										
2	Error										
3	Information										
4	Severe (fatal) error										
\$RESTART	Has the value TRUE if a batch job was restarted after it was interrupted by a system crash. Otherwise, \$RESTART has the value FALSE.										

12.2.4 Understanding Symbol Substitution

When a command line is executed, symbols in the following positions are automatically substituted:

- On the right side of an [:]= or [:]= assignment statement
- In a lexical function
- In the brackets on the left side of an assignment statement when you are performing substring substitution or number overlays (see Section 12.2.6.4)
- In a DEPOSIT, EXAMINE, IF, or WRITE command
- At the beginning of a command line

To force substitution of a symbol that is not in one of the positions listed, enclose the symbol with apostrophes as follows:

```
$ TYPE 'B'
```

To force substitution of a symbol within a quoted character string, precede that symbol with two apostrophes and follow it with a single apostrophe as follows:

```
$ T = "TYPE 'B'"
```

When processing a command line, DCL replaces symbols with their values in the following order:

- **Forced substitution**—From left to right, DCL replaces all strings delimited by apostrophes (or double apostrophes for strings within quotation marks). Symbols preceded by single apostrophes are translated iteratively; symbols preceded by double apostrophes are not.
- **Automatic substitution**—From left to right, DCL evaluates each value in the command line, executing it if it is a command and evaluating it if it is an expression. Symbols in expressions are replaced by their assigned values; this substitution is not iterative.

The following example demonstrates the effect of the order in which DCL substitutes symbols. Assume the following symbol definitions:

```
$ PN = "PRINT/NOTIFY"
$ FILE1 = "[BOLIVAR]TEST_CASE.TXT"
$ NUM = 1
```

Given the preceding symbol definitions, the following commands print the file named [BOLIVAR]TEST_CASE.TXT:

```
$ FILE = "'FILE'/'NUM'/"
$ PN 'FILE'
```

In the first command, forced substitution causes NUM to become 1, making FILE"NUM' become FILE1. If you enter the command SHOW SYMBOL FILE, you will see that FILE = "FILE1".

The second command performs two substitutions. First, 'FILE' is substituted with 'FILE1'. 'FILE1' also requires substitution because it is enclosed in single quotation marks. Automatic substitution causes FILE1 to become [BOLIVAR]TEST_CASE.TXT. This file name is then appended to the value of PN, which is PRINT/NOTIFY. The resulting string is as follows:

```
$ PRINT/NOTIFY [BOLIVAR]TEST_CASE.TXT
```

12.2.5 Using Symbol Values

A symbol can be defined with a character string, a number, a lexical function, a logical value, or another symbol. The following sections describe these values.

12.2.5.1 Character String Values

Characters fall into the following three main categories:

- **Alphanumeric characters**—The uppercase letters A through Z, the lowercase letters a through z, the digits 1 through 9, the dollar sign (\$), the underscore (_), and the hyphen (-).
- **Special characters**—All other characters that can be displayed or printed: the exclamation point (!), quotation marks ("), number sign (#), and so on.

12-8 Symbols: Defining Commands and Expressions

- **Nonprintable characters**—All characters that cannot be printed or displayed. In general, nonprintable characters are ignored for display and print purposes. However, several nonprintable characters serve control functions as follows:

Character	Function
HT	Starts printing or typing at the next horizontal tab
LF	Starts printing or typing on the next line
FF	Starts printing or typing at the top of the next page
CR	Starts printing or typing at the first space on the same line
ESC	Introduces a terminal escape sequence
SP	Inserts one space

You can define a character string by enclosing it in quotation marks. In this way, alphabetic case and spaces are preserved when the symbol assignment is made.

12.2.5.2 Numeric Values

A number can have the following values:

- **Decimal**—The ASCII characters 0 through 9
- **Hexadecimal**—The ASCII characters 0 through 9 and A through F
- **Octal**—The ASCII characters 0 through 7

The number you assign to a symbol must be in the range -2147483648 through 2147483647 (decimal). (An error is not reported if a number outside this range is specified or calculated, but an incorrect value results.)

At DCL command level and within command procedures, specify a number as follows:

- **Positive numbers**—Specify a positive number by typing the appropriate digits. The following example assigns the number 13 to the symbol DOG_COUNT:

```
$ DOG_COUNT = 13
$ SHOW SYMBOL DOG_COUNT
  DOG_COUNT = 13   Hex = 0000000D   Octal = 0000000015
```

- **Negative numbers**—Precede a negative number with a minus sign, as in the following example:

```
$ BALANCE = -15237
$ SHOW SYMBOL BALANCE
  BALANCE = -15237   Hex = FFFFC47B   Octal = 37777742173
```

- **Radix**—Specify a number in a radix other than decimal by preceding the number (but not the minus sign) with %X for hexadecimal numbers and %O for octal numbers. For example:

```
$ DOG_COUNT = %XD
$ SHOW SYMBOL DOG_COUNT
DOG_COUNT = 13   Hex = 0000000D   Octal = 0000000015

$ BALANCE = -%X3B85
$ SHOW SYMBOL BALANCE
BALANCE = -15237   Hex = FFFFC47B   Octal = 37777742173
```

- **Fractions**—A number cannot include a decimal point. In calculations, fractions are truncated; for example, 8 divided by 3 equals 2.

Numbers are stored internally as signed 4-byte integers, called longwords; positive numbers have values of 0 through 2147483647 and negative numbers have values of 4294967296 minus the absolute value of the number. The number -15237, for example, is stored as 4294952059. Negative numbers are converted back to minus-sign format for ASCII or decimal displays; however, they are not converted back for hexadecimal and octal displays. For example, the number -15237 appears in displays as hexadecimal FFFFC47B (decimal 4294952059) rather than hexadecimal -00003B85.

Numbers are stored in text files as a series of digits using ASCII conventions (for example, the digit 1 has a storage value of 49).

12.2.5.3 Values Returned by Lexical Functions

Typically used in command procedures, lexical functions provide users with a means to obtain information from the system, including information about system processes, batch and print queues, and user processes. You can also use lexical functions to manipulate character strings and translate logical names. When you assign a lexical function to a symbol, the symbol is equated to the information returned by the lexical function (for example, a number or character string). At DCL level, you can then display that information with the DCL command SHOW SYMBOL. In a command procedure, the information stored in the symbol can be used later in the procedure. See the description of the DCL commands and lexical functions in the Reference Section.

To use a lexical function, type the name of the lexical function (which always begins with F\$) and its argument list. Use the following syntax:

```
F$function-name(args[,...])
```

The argument list follows the function name with any number of intervening spaces and tabs. When using a lexical function, observe the following rules:

- Enclose the argument list in parentheses.
- Within the list, specify arguments in exact order and separate them with commas; even if you omit an optional argument, do not omit the comma.
- If no arguments are required, type an empty set of parentheses.

12-10 Symbols: Defining Commands and Expressions

- Do not enclose lexical functions in quotation marks.
- If an argument contains a character string, enclose the character string in quotation marks.
- If an argument contains an integer, a symbol, or another lexical function, do not enclose these values in quotation marks.

Use lexical functions the same way you would use character strings, integers, and symbols. The following example uses the `F$LENGTH` function. `F$LENGTH` returns an integer that specifies the length of the string. The returned value is assigned to the symbol `LEN`.

```
$ LEN = F$LENGTH("The cow jumped over the moon.")
$ SHOW SYMBOL LEN
  LEN = 29   Hex = 0000001D   Octal = 00000000035
```

You can use a lexical function in any position that you can use a symbol. In positions where symbol substitution must be forced by enclosing the symbol in apostrophes, lexical function evaluation must be forced by placing the lexical function within apostrophes. Lexical functions can also be used as argument values in other lexical functions.

The following example equates the length of the character symbol `LINE` to a numeric symbol named `L`:

```
$ L = F$LENGTH (LINE)
```

The following example strips the last two characters from the character string that is the value of the symbol `LINE`:

```
$ LINE = F$EXTRACT (0,F$LENGTH(LINE)-2,LINE)
```

12.2.5.4 Logical Values

Some operations interpret numbers and character strings as logical data with values as follows:

- **True**—A number has a logical value of true if it is odd (that is, the low-order bit is 1). A character string has a logical value of true if the first character is an uppercase or lowercase `T` or `Y`.
- **False**—A number has a logical value of false if it is even (that is, the low-order bit is 0). A character string has a logical value of false if the first character is **not** an uppercase or lowercase `T` or `Y`.

In both of the following examples, `DOG_COUNT` is assigned the value 13. `IF STATUS` means if the logical value of `STATUS` is true.

```
$ STATUS = 1
$ IF STATUS THEN DOG_COUNT = 13

$ STATUS = "TRUE"
$ IF STATUS THEN DOG_COUNT = 13
```

12.2.5.5 Using a Symbol as a Value for Another Symbol

After a symbol is defined, it can be used as a value for another symbol. A symbol can be interpreted as a character string or a number, depending on the context in which it is used. For example, suppose a symbol, COUNT, is assigned the integer value 3 as follows:

```
$ COUNT = 3
```

Then the value of COUNT can be used in other assignment statements. In the following example, the value of COUNT is added to 1:

```
$ TOTAL = COUNT + 1
```

The result, 4, is equated to the symbol TOTAL. You can confirm the assignment of the value to TOTAL by entering the SHOW SYMBOL command as follows:

```
$ SHOW SYMBOL TOTAL
TOTAL = 4   Hex = 00000004   Octal = 0000000004
```

You can include the symbol COUNT in a string assignment statement as follows:

```
$ BARK := P'COUNT'
```

COUNT is converted to a string value and appended to the character P. BARK now has the same value as P3.

To include a symbol in a string assignment, use either a colon and an equal sign (:=) or a colon and two equal signs (:=), and enclose the symbol in apostrophes. Otherwise, DCL will not recognize it as a symbol.

If you define a null character string for a symbol, that symbol has a value of 0, as shown in the following example:

```
$ A = ""
$ B = 2
$ C = A + B
$ SHOW SYMBOL C
C = 2   Hex = 00000002   Octal = 0000000002
```

12.2.6 Using Symbols in Expressions

An **expression** is a combination of values. Each value in an expression is connected to another value by an **operator**. Operators are denoted in the following ways:

- Special characters—Asterisk (*), slash (/), plus sign (+), and minus sign (-).
- Special names—.EQ., .EQS., .GE., .GES., .GT., .GTS., .LE., .LES., .LT., .LTS., .NE., .NES., .NOT., .AND., and .OR.; the names can be uppercase or lowercase.

12-12 Symbols: Defining Commands and Expressions

Data entities and operators can be adjacent or can be separated by any number of spaces or tabs. The values in the expression can be symbols or literal values (such as 3 or "DOG"). Expressions take the following two forms:

- **Operations**—An operation combines two data entities or alters a data entity. The following example combines the literal values 10 and 3 by adding them:

```
$ DOG_COUNT = 10 + 3
$ SHOW SYMBOL DOG_COUNT
DOG_COUNT = 13 Hex = 0000000D Octal = 0000000015
```

- **Comparisons**—A comparison evaluates a relationship between two entities as true or false. A true comparison evaluates to a numeric value of 1, and a false comparison evaluates to a numeric value of 0. The following example compares the value of the symbol DOG_COUNT with the literal value 13 and finds them to be equal:

```
$ DOG_CHECK = DOG_COUNT .EQ. 13
$ SHOW SYMBOL DOG_CHECK
DOG_CHECK = 1 Hex = 00000001 Octal = 0000000001
```

You can create character string expressions, numeric expressions, and logical expressions. These are described in the following sections.

12.2.6.1 Character String Expressions

A character string expression can contain character strings, lexical functions that evaluate to character strings, or symbols that have character string values. Attempting an operation or comparison between a character string and a number causes the character string to be converted to a number.

You can specify the following character string operations:

- **Concatenation**—The plus sign concatenates two character strings. For example:

```
$ COLOR = "light brown"
$ WEIGHT = "30 lbs."
$ DOG2 = "No tag, " + COLOR + ", " + WEIGHT
$ SHOW SYMBOL DOG2
DOG2 = "No tag, light brown, 30 lbs."
```

- **Reduction**—The minus sign removes the second character string from the first character string. For example:

```
$ SHOW SYMBOL DOG2
DOG2 = "No tag, light brown, 30 lbs."
$ DOG2 = DOG2 - ", 30 lbs."
$ SHOW SYMBOL DOG2
DOG2 = "No tag, light brown"
```

If the second character string occurs more than once in the first character string, only the first occurrence of the string is removed.

When you compare two character strings, the strings are compared character by character; strings of different lengths are not equal (for example, “dogs” is greater than “dog”).

The comparison criteria are the ASCII values of the characters. Under this criterion, the digits 0 through 9 are less than the letters A through Z, and the uppercase letters A through Z are less than the lowercase letters a through z. A character string comparison ends when either of the following conditions is true:

1. All the characters have been compared, in which case the strings are equal.
2. The first mismatch occurs.

You can specify the following varieties of string comparisons. In the examples, assume that the symbol `LAST_NAME` has the value “WHITFIELD.”

- **Equal to**—The operator `.EQS.` compares one character string to another for equality. The following comparison evaluates to 0 to indicate that the value of the symbol `LAST_NAME` does not equal the literal “NORMAN”:

```
$ TEST_NAME = LAST_NAME .EQS. "NORMAN"
$ SHOW SYMBOL TEST_NAME
TEST_NAME = 0 Hex = 00000000 Octal = 00000000000
```

- **Greater than or equal to**—The operator `.GES.` compares one character string to another for a greater or equal value in the first specified string. The following comparison evaluates to 1 to indicate that the value of the symbol `LAST_NAME` is greater than or equal to the literal “NORMAN”:

```
$ TEST_NAME = LAST_NAME .GES. "NORMAN"
$ SHOW SYMBOL TEST_NAME
TEST_NAME = 1 Hex = 00000001 Octal = 00000000001
```

- **Greater than**—The operator `.GTS.` compares one character string to another for a greater value in the first specified string. The following comparison evaluates to 1 to indicate that the value of the symbol `LAST_NAME` is greater than the literal “NORMAN”:

```
$ TEST_NAME = LAST_NAME .GTS. "NORMAN"
$ SHOW SYMBOL TEST_NAME
TEST_NAME = 1 Hex = 00000001 Octal = 00000000001
```

- **Less than or equal to**—The operator `.LES.` compares one character string to another for a lesser or equal value in the first specified string. The following comparison evaluates to 0 to indicate that the value of the symbol `LAST_NAME` is not less than or equal to the literal “NORMAN”:

```
$ TEST_NAME = LAST_NAME .LES. "NORMAN"
$ SHOW SYMBOL TEST_NAME
TEST_NAME = 0 Hex = 00000000 Octal = 00000000000
```

12-14 Symbols: Defining Commands and Expressions

- **Less than**—The operator `.LTS.` compares one character string to another for a lesser value in the first specified string. The following comparison evaluates to 0 to indicate that the value of the symbol `LAST_NAME` is not less than the literal `"NORMAN"`:

```
$ TEST_NAME = LAST_NAME .LTS. "NORMAN"
$ SHOW SYMBOL TEST_NAME
TEST_NAME = 0 Hex = 00000000 Octal = 00000000000
```

- **Not equal**—The operator `.NES.` compares one character string to another for inequality. The following comparison evaluates to 1 to indicate that the value of the symbol `LAST_NAME` does not equal the literal `"NORMAN"`:

```
$ TEST_NAME = LAST_NAME .NES. "NORMAN"
$ SHOW SYMBOL TEST_NAME
TEST_NAME = 1 Hex = 00000001 Octal = 00000000001
```

12.2.6.2 Numeric Expressions

In a numeric expression, the values involved must be literal numbers (such as 3) or symbols with numeric values. In addition, you can use a character string that represents a number (for example, `"23"` or `"-51"`). Attempting an operation or comparison between a number and a character string causes the character string to be converted to a number.

You can specify the following numeric operations:

- **Multiplication**—The asterisk multiplies two numbers. For example:

```
$ BALANCE = 142 * 14
$ SHOW SYMBOL BALANCE
BALANCE = 1988 Hex = 000007C4 Octal = 00000003704
```

- **Division**—The slash divides the first specified number by the second specified number. For example:

```
$ BALANCE = BALANCE / 14
$ SHOW SYMBOL BALANCE
BALANCE = 142 Hex = 0000008E Octal = 00000000216
```

If a number does not divide evenly, the remainder is lost. (No rounding takes place.)

- **Addition**—The plus sign adds two numbers. For example:

```
$ BALANCE = BALANCE + 37
$ SHOW SYMBOL BALANCE
BALANCE = 179 Hex = 000000B3 Octal = 00000000263
```

- **Subtraction**—The minus sign subtracts the second specified number from the first specified number. For example:

```
$ BALANCE = BALANCE - 15416
$ SHOW SYMBOL BALANCE
BALANCE = -15237 Hex = FFFFC47B Octal = 00000142173
```

- **Unary plus and minus**—The plus and minus signs change the sign of the number they precede. For example:

```
$ BALANCE = -(-142)
$ SHOW SYMBOL BALANCE
  BALANCE = 142   Hex = 0000008E   Octal = 00000000216
```

You can specify the following numeric comparisons:

- **Equal to**—The operator **.EQ.** compares one number to another for equality. The following comparison evaluates to 1 to indicate that **BALANCE** equals **-15237**:

```
$ TEST_BALANCE = BALANCE .EQ. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 1   Hex = 00000001   Octal = 00000000001
```

- **Greater than or equal to**—The operator **.GE.** compares one number to another for a greater or equal value in the first number. The following comparison evaluates to 1 to indicate that **BALANCE** is greater than or equal to **-15237**:

```
$ TEST_BALANCE = BALANCE .GE. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 1   Hex = 00000001   Octal = 00000000001
```

- **Greater than**—The operator **.GT.** compares one number to another for a greater value in the first number. The following comparison evaluates to 0 to indicate that **BALANCE** is not greater than **-15237**:

```
$ TEST_BALANCE = BALANCE .GT. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 0   Hex = 00000000   Octal = 00000000000
```

- **Less than or equal to**—The operator **.LE.** compares one number to another for a lesser or equal value in the first number. The following comparison evaluates to 1 to indicate that **BALANCE** is less than or equal to **-15237**:

```
$ TEST_BALANCE = BALANCE .LE. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 1   Hex = 00000001   Octal = 00000000001
```

- **Less than**—The operator **.LT.** compares one number to another for a lesser value in the first number. The following comparison evaluates to 0 to indicate that **BALANCE** is not less than **-15237**:

```
$ TEST_BALANCE = BALANCE .LT. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 0   Hex = 00000000   Octal = 00000000000
```

- **Not equal to**—The operator **.NE.** compares one number to another for inequality. The following comparison evaluates to 0 to indicate that **BALANCE** equals **-15237**:

```
$ TEST_BALANCE = BALANCE .NE. -15237
$ SHOW SYMBOL TEST_BALANCE
  TEST_BALANCE = 0   Hex = 00000000   Octal = 00000000000
```

12.2.6.3 Logical Expressions

A logical operation affects all the bits in the number being acted upon. The values for logical expressions are integers, and the result of the expression is an integer as well. If you specify a character string value in a logical expression, the string is converted to an integer before the expression is evaluated.

String and integer values are evaluated as follows:

- If the first character is T, t, Y, or y, a character string has a logical value of true (1).
- If the first character is not T, t, Y, or y, a character string has a logical value of false (0).
- If an integer is odd (the low-order bit is 1), it has a logical value of true (1).
- If an integer is even (the low-order bit is 0), it has a logical value of false (0).

Typically, you use logical expressions to evaluate the low-order bit of a logical value; that is, to determine whether the value is true or false. You can specify the following logical operations:

- **Not**—The operator `.NOT.` reverses the bit configuration of a logical value. A true value becomes false and a false value becomes true. The following example reverses a true value to false. The expression evaluates to `-2`; the value is even and is therefore false:

```
$ SHOW SYMBOL STATUS
  STATUS = 1   Hex = 00000001   Octal = 00000000001
$ STATUS = .NOT. STATUS
$ SHOW SYMBOL STATUS
  STATUS = -2  Hex = FFFFFFFE   Octal = 37777777776
```

- **And**—The operator `.AND.` combines two logical values as follows:

Bit Level	Entity Level
1 .AND. 1 = 1	true .AND. true = true
1 .AND. 0 = 0	true .AND. false = false
0 .AND. 1 = 0	false .AND. true = false
0 .AND. 0 = 0	false .AND. false = false

The following example combines a true value and a false value to produce a false value:

```
$ STAT1 = "TRUE"
$ STAT2 = "FALSE"
$ STATUS = STAT1 .AND. STAT2
$ SHOW SYMBOL STATUS
  STATUS = 0   Hex = 00000000   Octal = 00000000000
```

- Or—The operator `.OR.` combines two logical values as follows:

Bit Level	Entity Level
1 .OR. 1 = 1	true .OR. true = true
1 .OR. 0 = 1	true .OR. false = true
0 .OR. 1 = 1	false .OR. true = true
0 .OR. 0 = 0	false .OR. false = false

The following example combines a true value and a false value to produce a true value:

```
$ STAT1 = "TRUE"
$ STAT2 = "FALSE"
$ STATUS = STAT1 .OR. STAT2
$ SHOW SYMBOL STATUS
  STATUS = 1   Hex = 00000001   Octal = 0000000001
```

12.2.6.4 Substring Replacement and Numeric Overlays

You can replace a part of a character string with another character string. The assignment statement has the following format for local symbols:

`symbol-name[offset,size] := replacement-string`

The assignment statement has the following format for global symbols:

`symbol-name[offset,size] ::= replacement-string`

The fields are as follows:

- *Offset* is an integer that indicates the position of the replacement-string relative to the first character in the original string. An offset of 0 means the first character in the symbol, an offset of 1 means the second character, and so on.
- *Size* is an integer that indicates the length of the replacement-string.

To replace substrings, observe the following rules:

- The square brackets are required notation. No spaces are allowed between the symbol name and the left bracket.
- Integer values for size and offset values can be in the range of 0 through 768.
- The replacement-string must be a character string.

In the following example, the first assignment statement gives the symbol `A` the value `PACKRAT`. The second statement specifies that `MUSK` replace the first four characters in the value of `A`. The result is that the value of `A` becomes `MUSKRAT`.

12-18 Symbols: Defining Commands and Expressions

```
$ A := PACKRAT
$ A[0,4] := MUSK
$ SHOW SYMBOL A
  A = "MUSKRAT"
```

The symbol name you specify can be undefined initially. The assignment statement creates the symbol name and, if necessary, provides leading or trailing spaces in the symbol value. For example:

```
$ B[4,3] := RAT
```

If the symbol B does not have a previous value, it is given a value of four leading spaces followed by RAT. This format creates a blank line of any length. The following example gives the symbol LINE a value of 80 blank spaces:

```
$ LINE[0,80] := " "
```

Lining up records in columns makes a list easier to read and sort. You can use this format to specify how you want data to be stored. For example:

```
$ DATA[0,15] := 'NAME'
$ DATA[17,1] := 'GRADE'
```

The first statement fills in the first 15 columns of DATA with whatever value NAME has. The second statement fills in column 18 with whatever value GRADE has. Columns 16 and 17 contain blanks.

A special format of the assignment statement can also be used to perform binary (bit-level) overlays of the current symbol value. This format for local symbols is as follows:

```
$ symbol-name[bit-position,size] = replacement-expression
```

The format for global symbols is as follows:

```
$ symbol-name[bit-position,size] == replacement-expression
```

The fields are as follows:

- *Bit-position* is an integer that indicates the location relative to bit 0 at which the overlay is to occur.
- *Size* is an integer that indicates the number of bits to be overlaid.

When using numeric overlays, observe the following rules:

- The square brackets are required notation. No spaces are allowed between the symbol name and the left bracket.
- Literal values are assumed to be decimal.
- The maximum length for *size* is 32 bits.
- *Replacement-expression* must be a numeric expression.

- When *symbol-name* is either undefined or defined as a string, the result of the overlay is a string. Otherwise, the result is an integer.

The following example defines the symbol BELL as the value 7. The low-order byte of BELL has the binary value 00000111. By changing the 0 at offset 5 to 1 (beginning with 0, count bits from right to left), you create the binary value 00100111 (decimal value 39).

```
$ BELL = 7
$ BELL[5,1] = 1
$ SHOW SYMBOL BELL
  BELL = 39   Hex = 00000027   Octal = 0000000047
```

12.2.6.5 Order of Operations and the Results of Evaluations

An expression can contain any number of operations and comparisons. You can indicate precedence (the order in which operation and comparison should be evaluated) by placing operations to be performed first in parentheses. (Parentheses can be nested.) Otherwise, operations within an expression are evaluated in the following order:

1. Unary plus (+) and minus (-)
2. Multiplication and division
3. All other numeric and character operations
4. All numeric and character comparisons
5. Logical NOT operations
6. Logical AND operations
7. Logical OR operations

Operations and comparisons that have the same precedence are evaluated from left to right. The following examples illustrate precedence of operations in expressions:

```
$ BALANCE = 150 + 20 * 4
      (BALANCE = 150 + 80)
$ SHOW SYMBOL BALANCE
  BALANCE = 230   Hex = 000000E6   Octal = 00000000346

$ BALANCE = (150 + 20) * 4
      (BALANCE = 170 * 4)
$ SHOW SYMBOL BALANCE
  BALANCE = 680   Hex = 000002A8   Octal = 00000001250

$ STATUS = 150 * 4 .GT. 80 * 2
      (STATUS = 600 .GT. 160)
$ SHOW SYMBOL STATUS
  STATUS = 1   Hex = 00000001   Octal = 00000000001
```

12-20 Symbols: Defining Commands and Expressions

An expression has either an integer or a string value, depending on the types of values and the operators used. Table 12-1 summarizes how DCL evaluates expressions. The first column lists the different values and operators that an expression might contain. The second column tells, for each case, what the entire expression is equated to. Within the table *any value* stands for a string or an integer.

Table 12-1: Determining the Value of an Expression

Expression	Resulting Value Type
Integer value	Integer
String value	String
Integer lexical function	Integer
String lexical function	String
Integer symbol	Integer
String symbol	String
+, -, or .NOT. any value	Integer
Any value .AND. or .OR. any value	Integer
String + or—string	String
Integer + or—any value	Integer
Any value + or—integer	Integer
Any value * or / any value	Integer
Any value (string comparison) any value	Integer
Any value (numeric comparison) any value	Integer

Chapter 13

Command Procedures: Programming with DCL

A command procedure is a file that contains DCL commands and data lines used by the DCL commands. You can write both simple and complex command procedures. A simple command procedure executes a series of DCL commands in the order in which they are written. For example, the following command procedure sets your default directory and examines it:

```
$ ! PROCEDURES.COM
$ !
$ ! Enter [MAINT.PROCEDURES] and examine it
$ SET DEFAULT [MAINT.PROCEDURES]
$ DIRECTORY
```

A complex command procedure performs program-like functions. For example, the following command procedure asks for directory names and examines the directories:

```
$ ! DIRECTORY.COM
$ !
$ ! Examine directories
$START:
$ INQUIRE DIR_NAME "Directory name"
$ IF DIR_NAME .EQS. "" THEN GOTO END
$ DIRECTORY 'DIR_NAME'
$ GOTO START
$END:
```

This chapter describes how to create and use simple command procedures. For information about designing, coding, and testing complex command procedures, refer to the *Guide to Using VMS Command Procedures*.

13.1 Formatting a Command Procedure

Use the following rules to format a command procedure:

- Use a dollar sign (\$) to begin each line containing a command, comment, or label.
- Do not begin data lines with a dollar sign.
- Use comments to explain the command procedure to anyone who must maintain it. Place comments at the beginning of a procedure to describe the procedure and the parameters passed to it; place them at the beginning of each block of commands to describe that section of the procedure. The command interpreter ignores comments when the command procedure executes. Precede a comment with an exclamation point; the comment is all text to the right of an exclamation point. (To include a literal exclamation point in a command line, precede and follow it with quotation marks.)
- Use complete names for commands and qualifiers. Commands and qualifiers are usually self-explanatory when they are not abbreviated. Abbreviated commands and qualifiers may no longer be unique when new commands and qualifiers are added to the VMS operating system.
- Put labels on separate lines to make loops, subroutines, and conditional code easier to understand. (Labels mark the beginning of loops, subroutines, and conditional code.) You may choose to differentiate labels from commands by placing labels immediately after the dollar sign and by preceding commands with spaces. A label can have up to 255 characters, cannot contain embedded spaces, and must be ended with a colon. (The GOTO, GOSUB, and CALL commands transfer control to labels, which mark the beginning of a loop, a section of code, or a subroutine.)
- Separate command sequences with lines containing a dollar sign and an exclamation point (\$!). This makes it easier to see the outline of the command procedure. (If you insert blank lines, the command interpreter interprets them as data lines and produces a message warning you that the data lines were ignored.)

13.2 Executing a Command Procedure

You can execute command procedures as follows:

- Interactively from DCL level
- From within another command procedure
- On a remote node
- In batch mode

The following sections contain procedures for each of these methods.

Executing a Command Procedure Interactively

To execute a command procedure interactively, type an at sign (@) followed by the file specification of the procedure. The file type defaults to COM. For example, the following command executes the procedure SETD.COM in the directory [MAINT.PROCEDURES] on the disk WORKDISK:

```
$ @WORKDISK:[MAINT.PROCEDURES]SETD
```

To simplify the execution of a command procedure, create a global symbol or a logical name, and place the symbol or logical name definition in your login command procedure. (Section 13.3 describes how to create a login command procedure. Symbols are described in Chapter 12; logical names are described in Chapter 11.) Equating the command line to a global symbol allows you to invoke the command procedure from any directory by entering the global symbol name as shown in the following example:

```
$ SETD == "@WORKDISK:[MAINT.PROCEDURES]SETD"
$ SETD
```

Equating the file specification to a logical name allows you to invoke the command procedure from any directory by entering an at sign (@) followed by the logical name as shown in the following example:

```
$ DEFINE SETD WORKDISK:[MAINT.PROCEDURES]SETD.COM
$ @SETD
```

Executing a Command Procedure from Within Another Command Procedure

To execute a command procedure from within another command procedure, use the at sign (@) followed by the file specification of the procedure. For example, the following command procedure, WRITEDATE.COM, invokes the command procedure GETDATE.COM:

```
$! WRITEDATE.COM
$ INQUIRE TIME "What is the current time in hh:mm format?"
$ @GETDATE [JONES.COM]GETDATE.COM
```

Executing a Command Procedure on a Remote Node

To execute a command procedure interactively on a remote node, use the `TYPE` command. The `TYPE` command lets you execute command procedures to list the users logged on to the remote node or to display the status of services in the local cluster not provided clusterwide. The output of the command procedure is displayed on the user's terminal at the local node.

13-4 Command Procedures: Programming with DCL

To execute a command procedure in the top level directory of another account on the remote node, use an access control string in the command in the following format:

```
TYPE node_name"user_name password"::"TASK=command_procedure"
```

The variable *user_name* is the user name of the account on the remote node, *password* is the password of the account on the remote node, and *command_procedure* is the name of the command procedure.

For example, the following command procedure, SHOWUSERS.COM, displays the users logged in at the remote node on which the command procedure resides:

```
#! SHOWUSERS.COM
$ IF F$MODE() .EQS. "NETWORK" THEN DEFINE/USER SYS$OUTPUT SYS$NET
$ SHOW USERS
```

The following command executes the command procedure SHOWUSERS.COM and displays the output from this command procedure on the user's terminal. The command procedure resides in the top level directory of account BIRD on node ORIOLE.

```
$ TYPE ORIOLE"BIRD BOULDER"::"TASK=SHOWUSERS"
```

```
VAX/VMS Interactive Users
19-APR-1990 17:20:13.30
Total number of interactive users = 4
Username      Process Name   PID      Terminal
FLICKER       Freddie       00536278 TXA1:
ROBIN         Red           00892674 VTA2:
DOVE          Whitie        00847326 TXA3:
DUCK          Donna         02643859 RTA1:
```

Executing a Command Procedure as a Batch Job

You can also submit a command procedure to a batch queue to execute as a batch job. If your system is part of a network, you can submit a command procedure to execute as a batch job on a remote node. Within a command procedure, you can use DCL commands to open and close files on a remote node and read and write records in these files, using the same commands and qualifiers as for local files. Section 10.4 contains more information about batch jobs.

13.2.1 Changing Command Procedure Levels

A command procedure level is an input stream for the DCL command interpreter. You can create a maximum of 31 command levels. There are two ways to create new command levels. You can either use the CALL command to call a subroutine that exists within the command procedure, or you can nest command procedures by using an execute procedure (@) command inside one command procedure to invoke another command procedure. When you use the CALL command or nest a command procedure, the command level increases by 1.

When you invoke a command procedure, the command level increases by 1. For example, if you invoke procedure SUB from DCL command level (level 0), SUB executes at command level 1. If SUB then invokes SUB1, which invokes SUBSUB1, SUB1 executes at command level 2, and SUBSUB1 executes at command level 3.

By convention, DCL level (command level 0) is the highest command level and command level 31 the lowest command level. Thus, when you move from command level 3 to command level 2, you are moving to the next higher command level.

13.2.2 Exiting from a Command Procedure

A command procedure exits when it reaches the end of the procedure, an EXIT command, or a STOP command. If the exit is caused by the end of the procedure or an EXIT command, control returns to the next higher command level. If the exit is caused by the EXIT command, you can return a status value to the next higher command level by specifying the value as the parameter of the EXIT command. See Chapter 12 for more information about the global symbols \$STATUS and \$SEVERITY. For example, if you invoke SUB at DCL command level, and SUB calls SUB1, the following sequence of actions occurs:

1. Exiting from SUB1 returns you to SUB at the command line following the call to SUB1.
2. Exiting from SUB returns you to DCL command level.

If the exit is caused by the STOP command, control always returns to DCL command level, regardless of the command level in which the STOP command executes.

13.3 Designing a Login Command Procedure

You can create a command procedure, called a login command procedure, to execute the same commands each time you log in. Name your login command procedure LOGIN.COM, and place it in your top level directory, unless your system manager tells you otherwise.

13-6 Command Procedures: Programming with DCL

The following sample LOGIN.COM procedure illustrates some commands you might want to include in your login command procedure:

```
$! Sample LOGIN.COM for user MARCIA with
$! default disk of DISK3
$!
$! Exit if this is a batch job or another
$! type of noninteractive process
$!
$ IF F$MODE() .NES. "INTERACTIVE" THEN EXIT ①
$!
$! Tailor the default behavior of
$! certain DCL commands
$!
$ PUR*GE ::= PURGE/LOG
$ SUB*MIT ::= SUBMIT/NOLOG FILE/NOTIFY
$ M*AIL ::= MAIL/EDIT=(SEND, FORWARD, REPLY)
$!
$! Define global symbols
$!
$ DISPLAY ::= MONITOR PROCESSES/TOPCPU
$ GO ::= SET DEFAULT
$ LP ::= SHOW QUEUE/ALL SYS$PRINT
$ SS ::= SHOW SYMBOL
$ SQ ::= SHOW QUEUE/ALL
$ REM ::= @DISK3:[MARCIA.PROG]REMINDER
$ MAIN ::= SET DEFAULT DISK3:[MARCIA]
$!
$! Define logical names for:
$! Directories
$ DEFINE HOME DISK3:[MARCIA]
$ DEFINE REV DISK3:[MARCIA.REVIEWS]
$ DEFINE TOOLS DISK3:[MARCIA.TOOLS]
$! Files
$ DEFINE EQUIP DISK3:[MARCIA.LISTS]EQUIPMENT.DAT
$ DEFINE ACCOMP DISK3:[MARCIA]ACCOMPLISHMENTS.DAT
$! Users
$ DEFINE JON DAISY::HARRIS
$ DEFINE JANE DAISY::MOORE
$!
$! Define keys to execute commands
$!
$ DEFINE/KEY PF3 "SHOW USERS" /TERMINATE
$ DEFINE/KEY KP7 "SPAWN" /TERMINATE
$ DEFINE/KEY KP8 "ATTACH "
$ DEFINE/KEY KP4 "SET HOST "
$!
$! Change the prompt string to a three-character
$! abbreviation of the node name
$!
$ NODE = F$GETSYI("NODENAME") ②
$ PROMPT = F$EXTRACT(0,3,NODE)
$ SET PROMPT = "'PROMPT'> "
$!
$! Type the system notices ③
$!
```

```

$ ON ERROR THEN CONTINUE ④
$!
$ TYPE SYS$SYSTEM:NOTICE.TXT
$!
$! Run a program that displays today's appointments ⑤
$!
$ RUN DISK3:[MARCIA.PROG]REMINDER

```

- ① The F\$MODE lexical function returns the mode (interactive, batch, network, or other) that the process is in when the LOGIN.COM procedure is executing. This statement causes the procedure to exit unless you are using the system interactively. You should test the mode at the beginning of your LOGIN.COM procedure to ensure that commands used only in interactive mode are not executed in any other mode; in some cases, these commands can abort noninteractive processes.
- ② This group of commands changes the DCL prompt to the first three characters of the node name. The F\$GETSYI lexical function determines the node name. The F\$EXTRACT lexical function extracts the first three characters of the name. The SET PROMPT command changes the prompt from a dollar sign to the first three characters of the node name followed by the right angle bracket character (>) and a space.
- ③ This command displays the system notices that your system manager keeps in the file SYS\$SYSTEM:NOTICE.TXT.
- ④ This command runs a user-written program that displays your daily appointments. If you have written programs that you always run after you log in, you may prefer to execute them directly from your LOGIN.COM file.
- ⑤ This command requests the command interpreter to continue executing the procedure if any warning or error status value is returned when the command procedure is executed.

The system manager assigns the file specification for your login command procedure. In most installations, the login command procedure is called LOGIN.COM.

13.4 Using Loops

A loop is a group of commands that executes repeatedly until a condition is met. The following arrangement is recommended for statements that form a loop:

1. Begin the loop.
2. Change the termination variable.
3. Test the termination variable. If the condition is met, go to the end of the loop.
4. Perform the commands in the body of the loop.

13-8 Command Procedures: Programming with DCL

5. Return to the beginning of the loop.
6. End the loop.

You can also write loops that test the termination variable at the end of the loop rather than at the beginning as follows:

1. Begin the loop.
2. Perform the commands in the body of the loop.
3. Change the termination variable.
4. Test the termination variable. If the condition is not met, go to the beginning of the loop.
5. End the loop.

Note that when you test the termination variable at the end of the loop, the commands in the body of the loop execute at least once, regardless of the value in the termination variable.

Both of the following examples execute a loop that terminates when `COMMAND` equals "EX" (EXIT). (`F$EXTRACT` truncates `COMMAND` to its first two characters.) In the first example, `COMMAND`, the termination variable, is tested at the beginning of the loop; in the second, it is tested at the end of the loop.

```
$ ! EXAMPLE 1
$ !
$GET_COMMAND:
$ INQUIRE COMMAND-
  "Command (EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY) "
$ COMMAND = F$EXTRACT(0,2,COMMAND)
$ IF COMMAND .EQS. "EX" THEN GOTO END_LOOP
.
.
$ GOTO GET_COMMAND
$END_LOOP:

$ ! EXAMPLE 2
$ !
$GET_COMMAND:
$ INQUIRE COMMAND-
  "Command (EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY) "
$ COMMAND = F$EXTRACT(0,2,COMMAND)
.
.
$ IF COMMAND .NES. "EX" THEN GOTO GET_COMMAND
$ ! End of loop
```


To perform a loop a specific number of times, use a counter as the termination variable. In the following example, 10 file names are input by the user and placed into the local symbols FIL1, FIL2, . . . , FIL10:

```
$ NUM = 1                                ! Set counter
$LOOP:                                   ! Begin loop
$ INQUIRE FIL'NUM' "File"                ! Get file name
$ NUM = NUM + 1                          ! Update counter
$ IF NUM .LT. 11 THEN GOTO LOOP          ! Test for termination
$END_LOOP:                               ! End loop
.
.
.
```

To perform a loop for a known sequence of values, use F\$ELEMENT. In the following example, the files CHAP1, CHAP2, CHAP3, CHAPA, CHAPB, and CHAPC are processed in order:

```
$ FILE_LIST = "1,2,3,A,B,C"
$ INDEX = 0
$PROCESS:
$ NUM = F$ELEMENT(INDEX,"",FILE_LIST)
$ IF NUM .EQS. "" THEN GOTO END_LOOP
$ FILE = "CHAP'"NUM'"
$ ! process file named by FILE
.
.
.
$ INDEX = INDEX + 1
$ GOTO PROCESS
$END_LOOP:
$ EXIT
```

13.5 Passing Data

Command procedures frequently require data provided by a user. To specify the same data each time the command procedure is executed, place the data on data lines following the command that requires the data. (A data line is a line that does not begin with a dollar sign.) To include a data line that begins with a dollar sign, use the DCL commands DECK and EOD, which are described in the Reference Section.) The following command procedure executes the image CENSUS.EXE, which reads the data 1990, 1991, and 1992 each time the procedure is executed:

```
$ ! CENSUS.COM
$ !
$ RUN CENSUS
1990
1991
1992
$ EXIT
```

13-10 Command Procedures: Programming with DCL

The text on a data line is passed directly to the image; DCL does not process data lines. Therefore, DCL does not translate symbols or evaluate arithmetic expressions on data lines before passing the symbols or arithmetic expressions to the image. Logical names are not translated by DCL; therefore, a logical name included on a data line is not translated before it is passed to an image.

To specify different data each time a command procedure executes, use one of the following mechanisms, which are described in the following sections:

- Pass the data as one or more parameter values.
- Use the INQUIRE or READ command within the command procedure to prompt for data.
- Specify a device or file from which to read the data by redefining the logical name SYS\$INPUT.

13.5.1 Using Parameters to Pass Data

When you invoke a command procedure, you can pass it up to eight parameters. Place the parameters after the file specification of the command procedure. Separate the parameters with one or more spaces or tabs. For example, the following command invokes SUM.COM and passes eight parameters to the procedure:

```
§ @SUM 34 52 664 89 2 7 87 3
```

DCL places parameters passed to a command procedure in the local symbols P1 through P8; P1 is assigned the first parameter value; P2 the second; P3 the third; and so on. If you pass more than eight values, you receive the following error message and the command procedure does not execute:

```
%DCL-W-DEFOVF, too many command procedure parameters - limit to eight
```

If you pass fewer than eight values, the extra symbols are assigned **null values**. A null value is a string with no characters and is represented by double quotation marks ("").

To pass parameters to a command procedure executed in batch mode, use the /PARAMETERS qualifier of the SUBMIT command. If you pass more than one parameter, place the parameters in parentheses and separate them with commas. If you execute more than one command procedure using a single SUBMIT command, the specified parameters are used for each command procedure in the batch job. The following command passes three parameters to the command procedures ASK.COM and GO.COM, which are executed as batch jobs:

```
§ SUBMIT/PARAMETERS=(TODAY,TOMORROW,YESTERDAY) ASK.COM, GO.COM
```

Specify a parameter value as one of the following:

- **Integer**—When you specify an integer, it is converted to a string as follows:

```
$ @ADDER 24 25
```

In this example, P1 is the string value 24; P2 is the string value 25. (You can, however, use the symbols P1 and P2 in both integer and character string expressions; DCL performs the necessary conversions automatically.)

- **String**—Specify character strings as follows:

```
$ @DATA Paul Cramer
```

In this example, the strings Paul and Cramer are converted to uppercase letters; P1 is PAUL, and P2 is CRAMER.

To preserve spaces, tabs, or lowercase characters, place quotation marks before and after the string as follows:

```
$ @DATA "Paul Cramer"
```

In this example, P1 is Paul Cramer, and P2 is null.

- **Symbol**—To pass the value of a symbol, place an apostrophe character before and after the symbol, as shown in the following example. When passing a symbol, DCL removes quotation marks that enclose a string. (To preserve spaces, tabs, and lowercase characters in a symbol value, surround the symbol with quotation marks.)

```
$ NAME = "Paul Cramer"
$ @DATA 'NAME'
```

In this example, P1 is Paul, and P2 is Cramer.

To include a quotation mark as part of a string, enter three quotation marks as follows:

```
$ NEW_NAME = "" "Paul Cramer" ""
$ @DATA 'NEW_NAME'
```

In this example, P1 is "Paul Cramer", and P2 is null.

- **Null**—To pass a null parameter, use a set of quotation marks as a placeholder in the command string. In the following example, the first parameter passed to DATA.COM is a null parameter:

```
$ @DATA "" "Paul Cramer"
```

In this example, P1 is null, and P2 is Paul Cramer.

13-12 Command Procedures: Programming with DCL

For example, when DATA.COM is invoked with the following command, P1 through P8 are defined in DATA.COM as follows:

```
P1 = Paul Cramer
P2 = 24
P3 = (555) 111-1111
P4-P8 = null
```

```
$ @DATA "Paul Cramer" 24 "(555) 111-1111"
```

You can pass up to eight parameters to a nested command procedure. The local symbols P1 through P8 in the nested procedure are not related to the local symbols P1 through P8 in the invoking procedure. In the following example, DATA.COM invokes the nested command procedure NAME.COM:

```
$ ! DATA.COM
$ @NAME 'P1' Joe Cooper
```

Because P1 in DATA.COM is the string Paul Cramer, which contains no quotation marks, it is passed to NAME.COM as two parameters. In NAME.COM, P1 through P8 are defined as follows:

```
P1 = PAUL
P2 = CRAMER
P3 = JOE
P4 = COOPER
P5-P8 = null
```

Because DCL removes quotation marks when passing a symbol, you must enclose the value in three sets of quotation marks to preserve spaces, tabs, and lowercase characters in the symbol value. For example, in the following command procedure, the literal value in P1 is enclosed in three sets of quotation marks and passed to NAME.COM. If P1 originally contains the value "Paul Cramer", the value "Paul Cramer" is passed to NAME.COM.

```
$ ! DATA.COM
$ QUOTE = ""
$ P1 = QUOTE + P1 + QUOTE
$ @NAME 'P1' "Joe Cooper"
```

In this example, P1 is Paul Cramer and P2 is Joe Cooper in the command procedure NAME.COM.

An alternative is to enclose the text in quotation marks and, where a symbol appears, precede the symbol with two apostrophes and follow it with one apostrophe as follows:

```
$ ! DATA.COM
$ @NAME "'P1'"
```

Passing Data and Parameters to a Batch Job

To specify parameters for a job submitted in batch mode, use the /PARAMETERS qualifier of the SUBMIT command.

TIP: You can also pass data to a batch job by including the data in a command procedure or by defining SYS\$INPUT to be a file. The specified parameters are used for each command procedure in the batch job.

For example, the following SUBMIT command passes two parameters to the command procedures LIBRARY.COM and SORT.COM:

```
$ SUBMIT-
_ $ /PARAMETERS=(DISK:[ACCOUNT.BILLS]DATA.DAT,DISK:[ACCOUNT]NAME.DAT) -
_ $ LIBRARY.COM, SORT.COM
```

Your batch job executes as if you had logged in and executed each of the submitted command procedures. For example, the previous SUBMIT command executes a batch job that logs in under your account, executes your login command procedure, and then executes the following commands:

```
$ @LIBRARY DISK:[ACCOUNT.BILLS]DATA.DAT DISK:[ACCOUNT]NAME.DAT
$ @SORT DISK:[ACCOUNT.BILLS]DATA.DAT DISK:[ACCOUNT]NAME.DAT
```

13.5.2 Using the INQUIRE Command

You can use the INQUIRE command to obtain data for command procedures that you execute interactively. The INQUIRE command prompts for a value, reads the value from the terminal, and assigns it to a symbol. The response to the prompt is interpreted as a character string. By default, the response is converted to uppercase, multiple blanks and tabs are replaced by a single space, and leading and trailing spaces are removed. To preserve lowercase characters, multiple spaces, and tabs, enclose your response in quotation marks. For example, the following command procedure writes the prompt *Filename:* and puts your response into the local symbol FILE:

```
$ INQUIRE FILE "Filename"
```

To suppress the colon and space automatically added to the end of the prompt, use the /NOPUNCTUATION qualifier. To make the symbol global instead of local, use the /GLOBAL qualifier. For example, the following command procedure writes the prompt *Do you want to use default values?* and puts the response into the global symbol DEFAULT:

```
$ INQUIRE/NOPUNCTUATION/GLOBAL DEFAULT-
"Do you want to use default values?"
```

When a command procedure is submitted as a batch job, the value for a symbol specified in an INQUIRE command is read from the data line following the INQUIRE command. If you do not include a data line, the symbol is assigned a null value.

13.5.3 Using the READ Command

You can use the READ command to obtain data for command procedures that you execute interactively. The READ command prompts for a value, reads the value from the source specified by the first parameter, and assigns it to the symbol named as the second parameter. If you do not specify a prompt, the READ command outputs *Data:* as the default prompt. To specify a different prompt, use the /PROMPT qualifier. All characters typed on the terminal in response to the prompt are taken as an exact character string value (case, spaces, and tabs are preserved). The following command writes the prompt *Filename:*, reads the response from the source specified by the logical name SYS\$COMMAND (by default, the terminal), and assigns the response to the symbol FILE:

```
$ READ/PROMPT="Filename: " SYS$COMMAND FILE
```

13.5.4 Obtaining Data from SYS\$INPUT

Commands, utilities, and other system images usually get their input from the source specified by the logical name SYS\$INPUT. SYS\$INPUT is a process-permanent logical name that the system defines automatically. You can specify SYS\$INPUT as any one of the following:

- **Data line**—In a command procedure, the default value of SYS\$INPUT is the data lines of the procedure. For example, in the following command procedure, the image CENSUS.EXE uses the default value of SYS\$INPUT to take input (1990, 1991, and 1992) from the data lines:

```
$ ! CENSUS.COM
$ !
$ ! Execute CENSUS
$ RUN CENSUS
1990
1991
1992
$
```

- **Terminal**—A command procedure can get input from a terminal by defining SYS\$INPUT as the terminal. This allows you to perform interactive tasks from a command procedure. For example, the following command procedure defines SYS\$INPUT as SYS\$COMMAND, which is, by default, the terminal. The command procedure then invokes the EDT editor, beginning an interactive editing session.

```
$ ! EDIT.COM
$ !
$ ! Edit the file STATS.DAT
$ WRITE SYS$OUTPUT "Edit STATS.DAT:"
$ DEFINE/USER_MODE SYS$INPUT SYS$COMMAND:
$ EDIT STATS.DAT
```

(The /USER_MODE qualifier redefines SYS\$INPUT for the next image; you should use this qualifier whenever you redefine a process-permanent logical name.)

- **File**—A command procedure can get input from a file by defining `SYSS$INPUT` as a file. For example, the following command procedure defines `SYSS$INPUT` as the file `YEARS.DAT`, then invokes the program `CENSUS`. `CENSUS` reads its input from `SYSS$INPUT`, which points to the file `YEARS.DAT`.

```

$ ! CENSUS.COM
$ !
$ ! Execute CENSUS
$ DEFINE/USER_MODE SYSS$INPUT YEARS.DAT
$ RUN CENSUS

```

13.6 Returning Data

To return a value from a command procedure (either to a calling procedure or to DCL command level), you must assign the value to a global symbol. The global symbol can be read at any command level. Use comments to explain the use of any global symbols.

To create a global symbol, specify the value to be passed on the right side of a global assignment statement. For example, in the following command procedure, `DATA.COM` invokes the command procedure `NAME.COM`, passing `NAME.COM` a full name. `NAME.COM` places the last name in the global symbol `LAST_NAME`. When `NAME.COM` completes, DCL continues executing `DATA.COM`, which reads the last name by specifying the global symbol `LAST_NAME`. (The command procedure `NAME.COM` would be in a separate file; it is indented here for clarity.)

```

$ @DATA "Paul Cramer"

$ ! DATA.COM
$ !
$ ! P1 is a full name.
$ ! NAME.COM returns the last name in the
$ ! global symbol LAST_NAME.
$ !
$ @NAME 'P1'
    $ ! NAME.COM
    $ ! P1 is a first name
    $ ! P2 is a last name
    $ ! return P2 in the global symbol LAST_NAME
    $ LAST_NAME == P2
    $ EXIT
$ ! write LAST_NAME to the terminal
$ WRITE SYSS$OUTPUT "LAST_NAME = 'LAST_NAME'"

LAST_NAME = CRAMER

```

13.7 Displaying Data

Commands, utilities, and other system images normally write their output to the source specified by the logical name SYS\$OUTPUT. By default, SYS\$OUTPUT is equated to the terminal. However, you can redirect the output of a command procedure to a file by using the /OUTPUT qualifier. In the following example, output from the command procedure SETD.COM is written to the file RESULTS.TXT instead of to the terminal:

```
§ @SETD/OUTPUT=RESULTS.TXT
```

DCL commands that accept the /OUTPUT qualifier include ACCOUNTING, CALL, DIRECTORY, HELP, LIBRARY, RUN (process), SPAWN, and TYPE.

NOTE: When using (@) to execute a command procedure, the /OUTPUT qualifier must immediately follow the file name of the procedure.

13.7.1 Displaying Character Strings and Symbols

To display character strings and symbols on the terminal, use the WRITE command as follows:

- **Character string**—Enclose the text to be displayed in quotation marks. For example, the following command displays the text *Two files are written*.

```
§ WRITE SYS$OUTPUT "Two files are written."
```

- **Symbol value**—The WRITE command automatically substitutes symbols and lexical functions. For example, the following command displays the text *STAT1.DAT*, which is the value of the symbol FILE:

```
§ FILE = "STAT1.DAT"
§ WRITE SYS$OUTPUT FILE
```

- **Combination of character strings and symbol values**—Enclose the text to be displayed in quotation marks. Preface a symbol with two apostrophes, and follow it with one apostrophe. For example, the following lines display the text *STAT1.DAT and STAT2.DAT are written*. STAT1.DAT is the translation of the symbol AFILE; STAT2.DAT is the translation of the symbol BFILE.

```
§ AFILE = "STAT1.DAT"
§ BFILE = "STAT2.DAT"
§ WRITE SYS$OUTPUT "'AFILE' and 'BFILE' are written."
```

You can also use commas and quotation marks to display a combination of character strings and symbol values. For example, the following lines display the same text as the previous example:

```
§ AFILE = "STAT1.DAT"
§ BFILE = "STAT2.DAT"
§ WRITE SYS$OUTPUT AFILE, " and " ,BFILE, " were written."
```


13.7.2 Displaying Text

To display text that is more than one line long, use the `TYPE` command. `TYPE` writes data to `SYS$OUTPUT` (the terminal, by default). Using `SYS$INPUT` as the parameter causes `TYPE` to read the data from the command procedure. For example, when the following command procedure is executed, the text on the data lines is displayed on the terminal:

```
$ ! CLEAN.COM
$ !
$ TYPE SYS$INPUT
```

This command procedure executes a command that cleans up a directory.

Please enter one of the following commands after the prompt:

```
EXIT, DIRECTORY, TYPE, PURGE, DELETE, COPY
```

```
$ INQUIRE COMMAND "Command"
.
.
.
```

13.7.3 Displaying Files

To display the contents of a file, use the `TYPE` command. For example, the following command displays the file `STAT1.DAT` on the terminal:

```
$ TYPE DUA0:[HORACE]STAT1.DAT
```

13.8 Reading and Writing Files (File I/O)

To move data to and from files, use the `OPEN`, `CLOSE`, `READ`, and `WRITE` commands. The logical name you specify in the `OPEN` command is used to refer to the file in the `WRITE`, `READ`, and `CLOSE` commands.

13.8.1 Specifying Files in Batch Job Command Procedures

A batch job command procedure executes as if you had logged in and executed the command procedure interactively. Even if invoked from a subdirectory, the command procedure will begin executing within your top-level directory. Because your top-level directory may not be the default directory needed to access files required in a command procedure, command procedures that will be executed in batch mode should use one of the following mechanisms to ensure that the correct files are accessed:

- Use complete file specifications—When specifying a file in a command procedure or passing a file to a command procedure, include the device and directory names as part of the file specification, as shown in the previous example.

- Use the SET DEFAULT command—Before accessing a file in a command procedure, use the SET DEFAULT command to specify the proper device and directory.

13.8.2 Writing to a File

To write data to a file, use the following procedure:

1. Open the file—The OPEN command assigns to the logical name specified in the first parameter the file name specified in the second parameter.

Use the /APPEND qualifier with the OPEN command to write data to the end of an existing file. If you use the /APPEND qualifier to open a nonexistent file, an error occurs and no file is opened.

Use the /WRITE qualifier with the OPEN command to create a new file and to open this file for write access. If you use the /WRITE qualifier to open an existing file, a new version of that file is created.
2. Begin the write loop with a label—File I/O is always done in a loop unless you are writing or reading a single record.
3. Read the data to be written—Use the INQUIRE command or the READ command to read data into a symbol.
4. Test the data—Check the symbol containing the data. If the symbol is null (you pressed RETURN and entered no data on the line), you have reached the end of the data to be written to the file and should go to the end of the loop. Otherwise, continue.
5. Write the data to the file—Use the WRITE command to write the value of the symbol (one record) to the file.
6. Return to the beginning of the loop—You remain in the loop until there is no more data to be written to the file.
7. End the loop and close the file—The CLOSE command disassociates the file name from the logical name and closes the file. (Files opened by the OPEN command remain open until you log out unless you explicitly close them.)

For example, the following command procedure writes data to the new file STAT.DAT. If a file of that name exists, a new version is created.

```

$ ! Write a file
$ ON ERROR THEN EXIT                ! EXIT if the command procedure
$ !                                  ! cannot open the file
$ OPEN/WRITE IN_FILE STAT.DAT      ! Open the file
$ ON CONTROL_Y THEN GOTO END_WRITE  ! Close the file if you abort
$ !                                  ! execution with a CTRL/Y
$ ON ERROR THEN GOTO END_WRITE      ! Close the file if an error
$ !                                  ! occurs
$WRITE:                              ! Begin loop
$ INQUIRE STUFF "Input data"        ! Get input
$ IF STUFF .EQS. "" THEN GOTO END_WRITE ! Test for end of file
$ WRITE IN_FILE STUFF                ! Write to the file
$ GOTO WRITE                          ! Goto beginning
$END_WRITE:                          ! End loop
$ !
$ CLOSE IN_FILE                      ! Close the file

```

NOTE: The logical name in the OPEN command must be unique. If the OPEN command does not work and your commands seem correct, change the logical name in the OPEN command. Use the SHOW LOGICAL command to display logical name definitions.

If you want to create a file with a unique name, use the F\$SEARCH lexical function to see whether the name is already in the directory. (See the lexical function descriptions in the Reference Section for more information about F\$SEARCH.) The following command procedure prompts the user for a file name, then uses the F\$SEARCH lexical function to search the default directory for the name. If a file with that name already exists, control is passed to ERROR_1, the procedure prints the message *The file already exists*, and control returns to the label GET_NAME. You are again prompted for a file name.

```

$ ! FILES.COM
$ !
$GET_NAME:
$ INQUIRE FILE "File"              ! Get a file name
$ IF F$SEARCH (FILE) .NES. ""       ! Make sure file is name is unique
$ THEN
$   WRITE SYS$OUTPUT "The file already exists"
$   GOTO GET_NAME
$ ELSE
$   OPEN/WRITE IN_FILE 'FILE'      ! Open file with write access
$ ENDIF
.
.
.
$ EXIT

```

13.8.3 Reading from a File

To read data from a file, use the following procedure:

1. Open the file—The OPEN/READ command opens the file for read access and associates the file name with a logical name.
2. Begin the read loop—File I/O is always done in a loop unless you are reading or writing a single record.
3. Read the data from the file—Use the READ command with the /END_OF_FILE qualifier to read the next record in the file to a symbol. The /END_OF_FILE qualifier causes DCL to pass control to the label specified by the /END_OF_FILE qualifier when you reach the end of the file. Generally, you specify the label that marks the end of the read loop.
4. Process the data—When you read a file sequentially, process the current record before reading the next one.
5. Return to the beginning of the loop—You remain in the loop until you reach the end of the file.
6. End the loop and close the file—The CLOSE command disassociates the file name from the logical name and closes the file.

For example, the following command procedure reads and processes each record in the file STAT.DAT:

```

$ OPEN/READ OUT_F STAT.DAT           !Open the file
$ !
$READ_DATA:                          !Begin the loop
$ READ/END_OF_FILE=END_READ OUT_F STUFF !Read a record; test for
$                                     ! end of file
$                                     ! Process the data
$
.
.
.
$ GOTO READ_DATA                     !Go to the beginning
$                                     ! of the loop
$END_READ:                            !End of loop
$ !
$ CLOSE OUT_F                        !Close the file

```

13.8.4 Modifying a File

You can modify a file in the following ways:

- Rewrite records—This method allows you to make minor changes to a small number of records in a file. You cannot change the size of a record or the number of records in the file.
- Rewrite the file—This method allows you to change, delete, and insert records. You create a new file using the old file as the main source of input.
- Append records to a file—This method allows you to add new records to the end of the file.

Making Minor Modifications

To make minor changes to the records in a file, use the following procedure:

1. Open the file for both read and write access.
2. Use the READ command to read through the file until you reach the record that you want to modify.
3. Create a symbol containing the modified record. The modified record must be exactly the same size as the original record. If the text of the modified record is shorter, use spaces to make the record the same size. If the text of the modified record is longer, you cannot use this method to modify the file.
4. Use the WRITE/UPDATE command to write the modified record back to the file.
5. Repeat steps 2 through 4 until you have changed all records you intend to change.
6. Close the file.

Since this method does not allow you to modify the size of the record, use it only if you have formatted the records in a file (for example, in a data file).

The following command procedure reads each record in a data file. The record is displayed on the terminal, and you are asked whether the record needs to be modified. If you choose to modify the record, a new record is read from the terminal, and its length is compared to the length of the original record. If the original record is longer, extra spaces make the new record the same size. If the original record is shorter, an error message is displayed, and you are again prompted for a new record. If you choose not to modify the record, the next record is read from the file.

13-22 Command Procedures: Programming with DCL

```
$ ! MODIFY.COM
$ !
$ SPACES = "          "          ! Initialize string of spaces
$                               ! to make new record same size
$ OPEN/READ/WRITE FILE STATS.DAT ! Open the file
$ !
$BEGIN_LOOP:                    ! Begin the loop
$ !
$ READ/END_OF_FILE=END_LOOP FILE RECORD ! Read and display a record
$PROMPT:
$ WRITE SYS$OUTPUT RECORD
$ ! Does the user want to change the record?
$ INQUIRE/NOPUNCTUATION YN "Do you want to change this info? [N] "
$ !
$ IF YN .EQS. "Y"
$ THEN
$     ! Get the new record
$     INQUIRE NEW_RECORD "New Record"
$     OLD_LEN = F$LENGTH (RECORD)
$     ! Compare the old and new records
$     IF OLD_LEN .GE. F$LENGTH (NEW_RECORD)
$     THEN
$         IF OLD_LEN .NE. F$LENGTH (NEW_RECORD)
$         THEN
$             ! New record shorter than old record
$             PAD = F$EXTRACT (0,OLD_LEN-F$LENGTH(NEW_RECORD), SPACES)
$             NEW_RECORD = NEW_RECORD + PAD
$         ENDIF
$         ! Write the new record
$         WRITE/UPDATE FILE NEW_RECORD
$     ELSE
$         ! New record longer than old record
$         WRITE SYS$OUTPUT "ERROR -- New record is too long"
$         GOTO PROMPT
$     ENDIF
$ ENDIF
$ ! Get the next record
$ GOTO BEGIN_LOOP
$ !
$END_LOOP:
$ CLOSE FILE
$ EXIT
```

Making Major Modifications

To make extensive changes to a file, open that file for read access and open a new file for write access. Since the `/WRITE` qualifier opens a new file for write access, the new file can have the same name as the original file. The new file has a version number one higher than the version number of the old file.

NOTE: To ensure that the correct file is opened for reading, you must open the existing file for read access before you open the new version for write access.

To make major modifications to a file, use the following procedure:

1. Open the file for read access. This is the file you are modifying.

2. Open a new file for write access.
3. Use the READ command to read each record from the file you are modifying.

As you read each record from the original file, decide how the record is to be treated. In the following examples, the symbol RECORD contains the record read from the original file:

- No change—Write the same symbol to the new file.

```
$ ! No change
$ WRITE NEW_FILE RECORD
```

- Change—Use the INQUIRE command to read a different record into the symbol, then write the modified symbol to the new file.

```
$ ! Change
$ INQUIRE NEW_RECORD "New record"
$ WRITE NEW_FILE NEW_RECORD
```

- Delete—Do not write the symbol to the new file.
- Insert—Use a loop to read records into the symbol and to write the symbol to the new file, as shown in the following example:

```
$ ! Insertion
$LOOP:
$ !Get new records to insert
$ INQUIRE NEW_RECORD "New record"
$ IF RECORD .EQS. "" THEN GOTO END_LOOP
$ WRITE NEW_FILE NEW_RECORD
$ GOTO LOOP
$END_LOOP:
```

4. Continue reading and processing records until you have finished.
5. Use the CLOSE command to close both the input file and the output file.

Appending Records to a File

The OPEN/APPEND command allows you to append records to the end of an existing file. Use the following steps to append records to a file:

1. Use the OPEN command with the /APPEND qualifier to position the record pointer at the end of the file. The /APPEND qualifier does not create a new version of the file.
2. Use the WRITE command to write new data records. Continue adding records until you are finished.
3. Use the CLOSE command to close the file.

13.8.5 Handling Input/Output (I/O) Errors

Use the /ERROR qualifier with the OPEN, READ, or WRITE command to suppress error messages and to pass control to a specified label if an error occurs during an input or output operation. This qualifier overrides all other error-control mechanisms (except the /END_OF_FILE qualifier on the READ command). For example, in the following command procedure, if an error occurs during execution of the OPEN command, the message *Error opening STAT.DAT* is printed and the procedure exits:

```
$ OPEN/READ/ERROR=READ_ERR OUT_F STAT.DAT
.
.
.
$ EXIT
$READ_ERR:
$ WRITE SYS$OUTPUT "Error opening STAT.DAT"
$ EXIT
```

13.9 Restarting Batch Jobs

Chapter 10 describes how to reexecute your batch job if the system crashes before the job is finished. By default, a batch job is reexecuted beginning with the first line. However, you can use the following symbols in your command procedure to specify a different restarting point:

- **\$RESTART**—A global symbol whose value is true if the batch job has been started at least once before this execution. Do not specify a value for \$RESTART; the system will assign the appropriate value.
- **BATCH\$RESTART**—A global symbol whose value you specify using the SET RESTART_VALUE command.

The following procedure describes how to use these symbols in a command procedure:

1. Begin each possible starting point of the procedure with a label.
2. As the first step in each section, equate the value of BATCH\$RESTART to the label using the SET RESTART_VALUE command.
3. At the beginning of the procedure, test \$RESTART. If \$RESTART is true, issue a GOTO statement using BATCH\$RESTART as the transfer label.

The following command procedure extracts a number of modules from a library, concatenates those modules, and then sorts the resulting file. If aborted, the command procedure reexecutes from the beginning of the file, the statement labeled CONCATENATE_LIBRARIES or the statement labeled SORT_FILE, depending on the value of BATCH\$RESTART. (If you were extracting a number of separate modules, you could make each extraction a separate section.)


```

$ ! SORT_MODULES.COM
$ !
$ ! set default to the directory containing
$ ! the library whose modules are to be sorted
$ SET DEFAULT WORKDISK:[ACCOUNTS.DATA83]
$
$ ! check for restarting
$ IF $RESTART THEN GOTO "BATCH$RESTART"
$
$ EXTRACT_LIBRARIES:
$ SET RESTART_VALUE=EXTRACT_LIBRARIES
.
.
.
$ CONCATENATE_LIBRARIES:
$ SET RESTART_VALUE=CONCATENATE_LIBRARIES
.
.
.
$ SORT_FILE:
$ SET RESTART_VALUE=SORT_FILE
.
.
.
$ EXIT

```

13.10 Cleanup Operations

In general, execution of a command procedure should not change the user's process state. Therefore, a command procedure should include a set of commands that returns the process to its original state. Common cleanup operations include the following (see the lexical function descriptions in the Reference Section for lexical function specifications):

- **Closing files**—If you have opened any files, make sure that they are closed before the command procedure exits. You can use the lexical function `F$GETJPI` to examine the remaining open file quota (`FILCNT`) for the process. If `FILCNT` is the same at the beginning and end of the command procedure, you know that no files have been left open. For example, the following lines display a warning message if a file is left open:

```

$ FIL_COUNT = F$GETJPI("", "FILCNT")
.
.
.
$ IF FIL_COUNT .NE. F$GETJPI("", "FILCNT") THEN-
WRITE SYS$OUTPUT "WARNING -- file left open"

```
- **Deleting temporary or extraneous files**—If you have created temporary files, delete them. In general, if you have updated any files, you should purge them to delete the previous copies. Before you delete files you have not created, make sure you want to delete them. For example, if you have updated a file that contains crucial data, you might want to make the purging operation optional.

13-26 Command Procedures: Programming with DCL

- **Resetting default device and directory**—If you change the default device and/or directory, reset the original defaults before the command procedure exits.

To save the name of the original default directory, use the **DEFAULT** keyword of the **F\$ENVIRONMENT** lexical function. At the end of the command procedure, include a **SET DEFAULT** command that restores the saved device and directory.

The following example saves and restores device and directory defaults:

```
$ SAV_DEFAULT = F$ENVIRONMENT("DEFAULT")
.
.
.
$ SET DEFAULT 'SAV_DEFAULT'
```

The following table lists other commonly changed process characteristics as well as the lexical functions and commands used to save and restore the original settings:

Characteristic	To Save ...	To Restore ...
DCL prompt	F\$ENVIRONMENT	SET PROMPT
Default protection	F\$ENVIRONMENT	SET PROTECTION/DEFAULT
Privileges	F\$SETPRV	F\$SETPRV or SET PROCESS/PRIVILEGES
Control characters	F\$ENVIRONMENT	SET CONTROL
Verification	F\$VERIFY	F\$VERIFY
Message format	F\$ENVIRONMENT	SET MESSAGE
Key state	F\$ENVIRONMENT	SET KEY

To ensure that cleanup operations are performed even if the command procedure is aborted, begin each command level in the command procedure with the following statement:

```
$ ON CONTROL_Y THEN GOTO CLEAN_UP
```

In each command level of the command procedure, place cleanup operations after the **CLEAN_UP** label.

Reference Section



DCL Commands

This section describes each DCL command and lexical function. The commands are listed in alphabetical order, with the command name appearing at the top of every page. The lexical functions are grouped alphabetically under "Lexical Functions" (after the JOB command description); the name of the lexical function appears at the top of each page.

= (Assignment Statement)

Defines a symbolic name for a character string or integer value.

format

symbol-name *=[=]* *expression*

parameters

symbol-name

Specifies a 1 to 255 character alphanumeric string name for the symbol. The name can contain any alphanumeric characters from the DEC Multinational Character Set, the underscore (`_`), and the dollar sign (`$`). However, the name must begin *only* with an alphabetic character, an underscore, or a dollar sign. Using one equal sign (=) places the symbol name in the local symbol table for the current command level. Using two equal signs (==) places the symbol name in the global symbol table.

expression

Names the value on the right-hand side of an assignment statement. Can consist of a character string, an integer, a symbol name, a lexical function, or a combination of these entities. The components of the expression are evaluated, and the result is assigned to the symbol. All literal character strings must be enclosed in quotation marks. If the expression contains a symbol, the expression is evaluated using the symbol's value.

example

```
$ LIST == "DIRECTORY"
```

The assignment statement in this example assigns the user-defined synonym LIST as a global symbol definition for the DCL command DIRECTORY.

DCL-2 DCL Commands
:= (String Assignment)

:= (String Assignment)

Defines a symbolic name for a character string value.

format

symbol-name :=[=] string

parameters

symbol-name

Specifies a 1 to 255-character string name for the symbol. The name can contain any alphanumeric characters from the DEC Multinational Character Set, the underscore, and the dollar sign. However, the name must begin *only* with an alphabetic character, an underscore (_), or a dollar sign (\$). Using one equal sign (:=) places the symbol name in the local symbol table for the current command level. Using two equal signs (:=) places the symbol name in the global symbol table.

string

Names the character string value to be equated to the symbol. The string can contain any alphanumeric or special characters. String values are automatically converted to uppercase. Also, any leading and trailing spaces and tabs are removed, and multiple spaces and tabs between characters are compressed to a single space. To prohibit uppercase conversion and retain required space and tab characters in a string, place quotation marks around the string.

example

```
$ TIME := SHOW TIME
$ TIME
19-APR-1990 11:55:44
```

In this example, the symbol TIME is equated to the command string SHOW TIME. Because the symbol name appears as the first word in a command string, the command interpreter automatically substitutes it with its string value and executes the command SHOW TIME.

@ (Execute Procedure)

Executes a command procedure or requests the command interpreter to read subsequent command input from a specific file or device.

format

@ *file-spec* [*p1* [*p2* [... *p8*]]]

parameters

file-spec

Specifies either the input device or file for the preceding command, or the command procedure to be executed. The default file type is COM. Wildcard characters are not allowed in the file specification.

p1 [p2 [... p8]]

Specifies from one to eight optional parameters to pass to the command procedure. The symbols (P1, P2, . . . P8) are assigned character string values in the order of entry. The symbols are local to the specified command procedure. Separate each parameter with one or more blanks. Use two consecutive quotation marks ("") to specify a null parameter.

qualifier

/OUTPUT=file-spec

The name of the file to which the command procedure output is written. By default, the output is written to the current SYS\$OUTPUT device. The default output file type is LIS. Wildcard characters are not allowed in the output file specification. System responses and error messages are written to SYS\$COMMAND as well as to the specified file. The /OUTPUT qualifier must immediately follow the file specification of the command procedure; otherwise, the qualifier is interpreted as a parameter to pass to the command procedure.

example

```
$ CREATE DOFOR.COM
$ ON WARNING THEN EXIT
$ IF P1.EQS."" THEN INQUIRE P1 FILE
$ FORTRAN/LIST 'P1'
$ LINK 'P1'
$ RUN 'P1'
$ PRINT 'P1'
CTRLZ
$ @DOFOR AVERAGE
```

This example shows a command procedure, named DOFOR.COM, that executes the FORTRAN, LINK, and RUN commands to compile, link, and execute a program. The ON command requests that the procedure not continue if any of the commands result in warnings or errors.

When you execute DOFOR.COM, you can pass the file specification of the FORTRAN program as the parameter P1. If you do not specify a value for P1 when you execute the procedure, the INQUIRE command issues a prompting message to the terminal and equates what you enter with the symbol P1. In this example, the file name AVERAGE is assigned to P1. The file type is not included because the commands FORTRAN, LINK, RUN, and PRINT provide default file types.

ACCOUNTING

Invokes the Accounting Utility to collect, record, and report accounting data. For more information about the Accounting Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ALLOCATE

Provides your process with exclusive access to a device until you deallocate the device or terminate your process. Optionally associates a logical name with the device.

format

ALLOCATE *device-name[:][,...]* [*logical-name[:]*]

parameters

device-name[:][,...]

Specifies the name of a physical device or a logical name that translates to the name of a physical device. The device name can be generic: if no controller or unit number is specified, any device that satisfies the specified part of the name is allocated. If more than one device is specified, the first available device is allocated.

logical-name

Specifies a character string of 1 through 255 characters. Enclose the string in quotation marks (") if it contains blanks. Trailing colons are not used. The name becomes a process logical name with the device name as the equivalence name. The logical name remains defined until it is explicitly deleted or your process terminates.

qualifiers

/GENERIC

/NOGENERIC (default)

Indicates that the first parameter is a device *type* rather than a device *name*. Example device types are RX50, RD52, TK50, RC25, RCF25, RL02. The first free, nonallocated device of the specified name and type is allocated.

/LOG (default)

/NOLOG

Displays a message indicating the name of the device allocated. If the operation specifies a logical name that is currently assigned to another device, displays the superseded value.

example

```
§ ALLOCATE /GENERIC RX50 ACCOUNTS
```

The **ALLOCATE** command in this example allocates the first free floppy disk drive and makes its name equivalent to the process logical name **ACCOUNTS**.

ANALYZE/AUDIT

Invokes the Audit Analysis Utility (**ANALYZE/AUDIT**) to selectively extract and display information from security audit log files or security archive files. For more information about the Audit Analysis Utility, see the Audit Analysis Utility in the VMS base documentation set.

format

ANALYZE/AUDIT *file-spec*

ANALYZE/CRASH_DUMP

Invokes the System Dump Analyzer Utility (**SDA**) for analysis of a system dump file. The **/CRASH_DUMP** qualifier is required.

format

ANALYZE/CRASH_DUMP *file-spec*

ANALYZE/DISK_STRUCTURE

Invokes the Analyze/Disk_Structure Utility to do the following:

- Check the readability and validity of Files-11 Structure Level 1 and Files-11 Structure Level 2 disk volumes
- Report errors and inconsistencies

The **/DISK_STRUCTURE** qualifier is required. For more information about the Analyze/Disk_Structure Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ANALYZE/ERROR_LOG

Invokes the Errorlog Report Formatter (ERF) to report selectively the contents of an error log file. The /ERROR_LOG qualifier is required. For more information about the Error Log Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ANALYZE/IMAGE

Analyzes the contents of an executable image file or a shareable image file and checks for obvious errors in the image file. See the description of the linker for general information about image files.

format

ANALYZE/IMAGE *file-spec* [...]

parameter

file-spec[,...]

Specifies the name of one or more image files that you want analyzed. You must specify at least one file name. If you specify more than one file, separate the file specifications with either commas or plus signs. The default file type is EXE.

Wildcard characters are allowed in the file specification.

qualifiers

/FIXUP_SECTION

Positional Qualifier. If you specify /FIXUP_SECTION after the ANALYZE/IMAGE command, the fixup section of each image file in the parameter list is analyzed. If you specify /FIXUP_SECTION after a file specification, only the information in the fixup section of that image file is analyzed.

/GST

Positional Qualifier. This qualifier is valid only for shareable images. If you specify /GST after the ANALYZE/IMAGE command, the global symbol table records of each image file in the parameter list are analyzed. If you specify /GST after a file specification, only the global symbol table records of that file are analyzed.

/HEADER

Positional Qualifier. Specifies that the analysis should include all header items and image section descriptions.

/INTERACTIVE

/NOINTERACTIVE (*default*)

Specifies whether or not the analysis is interactive.

/OUTPUT=*file-spec*

Identifies the output file for storing the results of the image analysis. No wildcard characters are allowed in the file specification.

/PATCH_TEXT

Positional Qualifier. If you specify **/PATCH_TEXT** after the **ANALYZE/IMAGE** command, the patch text records of each image file in the parameter list are analyzed. If you specify **/PATCH_TEXT** after a file specification, only the patch text records of that file are analyzed.

example

```
$ ANALYZE/IMAGE/OUTPUT=LIALPHEX/FIXUP_SECTION/PATCH_TEXT LINEDT, ALPHA
```

The **ANALYZE/IMAGE** command in this example produces a description and an error analysis of the fixup sections and patch text records of **LINEDT.EXE** and **ALPHA.EXE** in file **LIALPHEX.ANL**. Output is sent to the file **LIALPHEX.ANL**.

ANALYZE/MEDIA

Invokes the Bad Block Locator Utility (BAD), which analyzes block-addressable devices and records the location of blocks that cannot reliably store data. For more information about the Bad Block Locator Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

ANALYZE/OBJECT

Analyzes the contents of an object file and checks for any obvious errors.

format

```
ANALYZE/OBJECT file-spec[,...]
```

parameter

file-spec[,...]

Specifies the object files or object module libraries you want analyzed (default file type is OBJ). Use commas or plus signs to separate file specifications. Wildcard characters are allowed.

qualifiers

/DBG

Positional qualifier. If you want the analysis to include debugger information for all files in the parameter list, insert the */DBG* qualifier immediately following the */OBJECT* qualifier. If you want the analysis to include debugger information selectively, insert the */DBG* qualifier immediately following the selected file specification(s).

/EOM

Positional qualifier. Specifies that the analysis should be limited to MHD records, EOM records, and records explicitly specified by the command. If you want this to apply to all files in the parameter list, insert the */EOM* qualifier immediately following the */OBJECT* qualifier. To make this applicable selectively, insert the */EOM* qualifier immediately following the selected file specification(s).

/GSD

Positional qualifier. If you want the analysis to include global symbol directory records for each file in the parameter list, specify */GSD* immediately following the */OBJECT* qualifier. If you want the analysis to include global symbol directory records selectively, insert the */GSD* qualifier immediately following the selected file specification(s).

/INCLUDE[=(module[,...])]

When the specified file is an object module library, use this qualifier to list selected object modules within the library for analysis. If you omit the list or specify an asterisk, all modules are analyzed.

/INTERACTIVE

/NOINTERACTIVE (default)

Controls whether the analysis occurs interactively.

/LNK

Positional qualifier. If you want the analysis to include link option specification records for each file in the parameter list, specify */LNK* immediately following the */OBJECT* qualifier. If you want the analysis to include link option specification records selectively, insert the */LNK* qualifier immediately following the selected file specification(s).

/MHD

Positional qualifier. Specifies that the analysis should be limited to MHD records, EOM records, and records explicitly specified by the command. If you want this to apply to all files in the parameter list, insert the */MHD* qualifier immediately following the */OBJECT* qualifier. To make this applicable selectively, insert the */MHD* qualifier immediately following the selected file specification(s).

/OUTPUT[=file-spec]

Directs the output of the object analysis (default is SYS\$OUTPUT). No wildcard characters are allowed in the file specification.

/TBT

Positional qualifier. If you want the analysis to include traceback records for each file in the parameter list, specify */TBT* immediately following the */OBJECT* qualifier. If you want the analysis to include traceback records selectively, insert the */TBT* qualifier immediately following the selected file specification(s).

/TIR

Positional qualifier. If you want the analysis to include text information and relocation records for each file in the parameter list, specify */TIR* immediately following the */OBJECT* qualifier. If you want the analysis to include text information and relocation records selectively, insert the */TIR* qualifier immediately following the selected file specification(s).

example

```
$ ANALYZE/OBJECT/OUTPUT=LIOBJ/DBG LINEDT
```

In this example, the *ANALYZE/OBJECT* command analyzes only the debugger information records of the file *LINEDT.OBJ*. Output is to the file *LIOBJ.ANL*.

ANALYZE/PROCESS_DUMP

Invokes the VMS Debugger for analysis of a process dump file that was created when an image failed during execution (use the */DUMP* qualifier with the *RUN* or *SET PROCESS* commands to generate a dump file).

Requires read (R) access to the dump file.

format

```
ANALYZE/PROCESS_DUMP dump-file
```

DCL-10 DCL Commands
ANALYZE/PROCESS_DUMP

parameter

dump-file

Specifies the dump file to be analyzed with the debugger.

qualifiers

/FULL

Displays all known information about the failing process.

/IMAGE=image-name

/NOIMAGE

Specifies the image whose symbols are to be used in analyzing the dump. If you use the */NOIMAGE* qualifier, no symbols are taken from any image. By default, symbols are taken from the image with the same name as the image that was running at the time of the dump.

/INTERACTIVE

/NOINTERACTIVE (default)

Causes the display of information to pause when your terminal screen is filled. Press RETURN to display additional information. By default, the display is continuous.

/MISCELLANEOUS

Displays all the miscellaneous information in the dump.

/OUTPUT=file-spec

Writes the information to the specified file. By default, the information is written to the current SYS\$OUTPUT device.

/RELOCATION

Displays the addresses to which data structures saved in the dump are mapped in P0 space.

example

```
$ ANALYZE/PROCESS/FULL ZIPLIST

R0 = 00018292 R1 = 8013DE20 R2 = 7FFE6A40 R3 = 7FFE6A98
R4 = 8013DE20 R5 = 00000000 R6 = 7FFE7B9A R7 = 0000F000
R8 = 00000000 R9 = 00000000 R10 = 00000000 R11 = 00000000
SP = 7FFAEF44 AP = 7FFAEF48 FP = 7FFAEF84
FREE_P0_VA 00001600 FREE_P1_VA 7FFAC600
Active ASTs 00 Enabled ASTs 0F
Current Privileges FFFFFFF80 1010C100
Event Flags 00000000 E0000000
Buffered I/O count/limit 6/6
Direct I/O count/limit 6/6
File count/limit 27/30
Process count/limit 0/0
Timer queue count/limit 10/10
AST count/limit 6/6
Enqueue count/limit 30/30
Buffered I/O total 7 Direct I/O total 18
Link Date 27-DEC-1988 15:02:00.48 Patch Date 17-NOV-1988 00:01:53.71
ECO Level 0030008C 00540040 00000000 34303230
Kernel stack 00000000 pages at 00000000 moved to 00000000
Exec stack 00000000 pages at 00000000 moved to 00000000
Vector page 00000001 page at 7FFEFEE0 moved to 00001600
PIO (RMS) area 00000005 pages at 7FFE1200 moved to 00001800
Image activator context 00000001 page at 7FFE3400 moved to 00002200
User writeable context 0000000A pages at 7FFE1C00 moved to 00002400
Creating a subprocess
VAX DEBUG Version X5.0-2
DBG>
```

This example shows the output of the ANALYZE/PROCESS command when used with the /FULL qualifier. The file specified, ZIPLIST, contains the dump of a process that encountered a fatal error. The DBG> prompt indicates that the debugger is ready to accept commands.

ANALYZE/RMS_FILE

Invokes the Analyze/RMS_File Utility (ANALYZE/RMS_FILE) to inspect and analyze the internal structure of a VMS RMS file. The /RMS_FILE qualifier is required.

format

ANALYZE/RMS_FILE *file-spec[,...]*

ANALYZE/SYSTEM

Invokes the System Dump Analyzer (SDA) for analysis of the running system. The /SYSTEM qualifier is required.

format

ANALYZE/SYSTEM

APPEND

Adds the contents of one or more specified input files to the end of the specified output file.

format

APPEND *input-file-spec[,...]* *output-file-spec*

parameters

input-file-spec[,...]

Specifies the names of one or more input files to be appended. Multiple input files are appended to the output file in the order specified. If you specify more than one input file, separate multiple file specifications with either commas or plus signs. You can use wildcard characters in the input file specifications.

output-file-spec

Specifies the name of the file to which the input files will be appended. You must specify at least one field in the output file specification. If you do not specify a device or directory, the APPEND command uses the current default device and directory. Other unspecified fields default to the corresponding fields of the first input file specification.

qualifiers

/ALLOCATION=number-of-blocks

Output-file-spec qualifier. Forces the initial allocation of the output file to the specified number of 512-byte blocks. If you do not specify the /ALLOCATION qualifier, the initial allocation of the output file is determined by the size of the input file. Relevant only with the /NEW_VERSION qualifier.

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM***/NOCONFIRM (default)***

Controls whether a request is issued before each APPEND operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	<input type="checkbox"/> RET	

You can use any combination of upper- and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CONTIGUOUS***/NOCONTIGUOUS***

Output-file-spec qualifier. Specifies that the output file must occupy physically contiguous disk blocks. By default, the APPEND command creates an output file in the same format as the corresponding input file and does not report an error if not enough space exists for a contiguous allocation. Relevant only with the /NEW_VERSION qualifier.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifiers. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the append operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */EXPIRED* selects files according to their expiration dates. (The expiration date is set with the *SET FILE/EXPIRATION_DATE* command.) The */EXPIRED* qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

/EXTENSION=number-of-blocks

Output-file-spec qualifier. Specifies the number of blocks to be added to the output file each time the file is extended. When you specify */EXTENSION*, the */NEW_VERSION* qualifier is assumed and need not be typed on the command line. Relevant only with the */NEW_VERSION* qualifier.

/LOG

/NOLOG (default)

Controls whether the APPEND command displays the file specifications of each file appended. If */LOG* is specified, displays the file specifications of the input and output files as well as the number of blocks or records appended after each append operation.

/MODIFIED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/NEW_VERSION

/NONNEW_VERSION (default)

Output-file-spec qualifier. Controls whether the APPEND command creates a new output file if the specified output file does not exist. If the specified output file does not already exist, use the */NEW_VERSION* qualifier to create a new output file. If the output file does exist, the */NEW_VERSION* qualifier is ignored and the input file is appended to the output file.

APPEND

/PROTECTION=(code)

Output-file-spec qualifier. Specifies protection for the output file. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and access as R (read), W (write), E (execute), or D (delete). The default protection, including any protection attributes not specified, is that of the existing output file. If no output file exists, the current default protection applies. Relevant only with the */NEW_VERSION* qualifier.

/READ_CHECK***/NOREAD_CHECK (default)***

Input-file-spec qualifier. Reads each record in the input files twice to verify that it has been read correctly.

/SINCE[=time]

Selects for the append operation only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with */BEFORE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

/WRITE_CHECK***/NOWRITE_CHECK (default)***

Output-file-spec qualifier. Reads each record in the output file after the record is written to verify that it was appended successfully and that the output file can subsequently be read without error.

example

```
$ APPEND/NEW_VERSION/LOG *.TXT MEM.SUM
%APPEND-I-CREATED, USE$:[MAL]MEM.SUM;1 created
%APPEND-S-COPIED, USE$:[MAL]A.TXT;2 copied to USE$:[MAL]MEM.SUM;1 (1 block)
%APPEND-S-APPENDED, USE$:[MAL]B.TXT;3 appended to USE$:[MAL]MEM.SUM;1 (3 records)
%APPEND-S-APPENDED, USE$:[MAL]G.TXT;7 appended to USE$:[MAL]MEM.SUM;1 (51 records)
```

The APPEND command appends all files with file types of TXT to a file named MEM.SUM. The */LOG* qualifier requests a display of the specifications of each input file appended. If the file MEM.SUM does not exist, the APPEND command creates it, as the output shows. The number of blocks or records shown in the output refers to the source file and not to the target file total.

ASSIGN

Creates a logical name and assigns an equivalence string, or a list of strings, to the specified logical name. If you specify an existing logical name, the new equivalence name replaces the existing equivalence name.

format

ASSIGN *equivalence-name[,...]* *logical-name[:]*

parameters

equivalence-name[,...]

Specifies a character string of 1 to 255 characters. Defines the equivalence name, usually a file specification, device name, or other logical name, to be associated with the logical name in the specified logical name table. If the string contains other than uppercase alphanumeric, dollar sign, or underscore characters, enclose it in quotation marks ("). Use two consecutive quotation marks ("") to denote an actual quotation mark. Specifying more than one equivalence name for a logical name creates a search list.

logical-name

Specifies the logical name string, which is a character string containing up to 255 characters. You choose a logical name to represent the equivalence name in the specified logical name table. If the string contains other than uppercase alphanumeric, dollar sign, or underscore characters, enclose it in quotation marks ("). Use two consecutive quotation marks ("") to denote an actual quotation mark. If you terminate the logical-name parameter with a colon, the system removes the colon before placing the name in a logical name table. (This differs from the DEFINE command, which saves the colon.) If the logical name is to be entered into the process directory (LNM\$PROCESS_DIRECTORY) or system directory (LNM\$SYSTEM_DIRECTORY) logical name tables, then the name may only have from 1 to 31 alphanumeric characters (including the dollar sign and underscore). By default, the logical name is placed in the process logical name table.

qualifiers

/EXECUTIVE_MODE

Requires SYSNAM privilege. Specifies the mode of the logical name. If you specify executive mode, but do not have SYSNAM privilege, the qualifier is ignored and a supervisor mode logical name is created. The mode of the logical name must be the same as or external to (less privileged than) the mode of the table in which you are placing the name.

/GROUP

Requires SYSPRV or GRPNAM privilege. Places the logical name in the group logical name table. Other users who have the same group number in their user identification codes (UICs) can access the logical name. The /GROUP qualifier is synonymous with /TABLE=LNМ\$GROUP.

/JOB

Places the logical name in the jobwide logical name table. All processes within the same job tree as the process creating the logical name can access the logical name. The /JOB qualifier is synonymous with /TABLE=LNМ\$JOB.

/LOG (default)

/NOLOG

Displays a message when a new logical name supersedes an existing name.

/NAME_ATTRIBUTES[=(keyword[,...])]

Specifies the attributes for a logical name. By default, no attributes are set. You can specify the following keywords for attributes:

- | | |
|----------|---|
| CONFINE | Does not copy the logical name into a spawned subprocess; relevant only for logical names in a private table. |
| NO_ALIAS | Prohibits creation of logical names with the same name in an outer (less privileged) access mode within the specified table. If another logical name with the same name and an outer access mode already exists in this table, the name is deleted. |

/PROCESS (default)

Places the logical name in the process logical name table. The /PROCESS qualifier is synonymous with /TABLE=LNМ\$PROCESS.

/SUPERVISOR_MODE (default)

Creates a supervisor mode logical name in the specified table.

/SYSTEM

Requires SYSNAM or SYSPRV privilege. Places the logical name in the system logical name table. All system users can access the logical name. The /SYSTEM qualifier is synonymous with /TABLE=LNМ\$SYSTEM.

/TABLE=name

Requires WRITE (W) access to the table if the table is shareable. Specifies the logical name table in which the logical name is to be entered. You can use the /TABLE qualifier to specify a user-defined logical name table (created with the CREATE/NAME_TABLE command); to specify the process, job, group, or system logical name tables; or to specify the process or system logical name directory tables. If you specify the table name using a logical name that has more than one translation, the logical

DCL-18 DCL Commands

ASSIGN

name is placed in the first table found. If you do not explicitly specify the /TABLE qualifier, the default is /TABLE=LNM\$PROCESS (or /PROCESS).

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

Equivalence-name qualifier. Specifies attributes of the equivalence-name parameter. Possible keywords are as follows:

- CONCEALED** Indicates that the equivalence string is the name of a concealed device.
- TERMINAL** Indicates that the equivalence string should not be translated iteratively; logical name translation should terminate with the current equivalence string.

/USER_MODE

Creates a user mode logical name in the specified table.

If you specify a user mode logical name in the process logical name table, that logical name is used for the execution of a single image only; user mode entries are deleted from the logical name table when any image executing in the process exits; that is, after any DCL command that executes an image or user program completes execution.

example

```
$ ASSIGN XXX1:[CHARLES] CHARLIE
$ PRINT CHARLIE:TEST.DAT
Job 274 entered on queue SYS$PRINT
```

The ASSIGN command in this example associates the logical name CHARLIE with the directory name [CHARLES] on the disk XXX1. 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 XXX1:[CHARLES]TEST.DAT to the system printer.

ASSIGN/MERGE

Removes all jobs from one queue and merges them into another existing queue. Does not affect jobs that are executing.

Requires OPER privilege or EXECUTE access to both queues.

format

ASSIGN/MERGE *target-queue[:]* *source-queue[:]*

parameters

target-queue[:]

Specifies the name of the queue into which the jobs are being merged.

source-queue[:]

Specifies the name of the queue from which the jobs are being removed.

example

```
$ STOP/QUEUE/NEXT LPB0  
$ STOP/QUEUE/REQUEUE=LPA0 LPB0  
$ ASSIGN/MERGE LPA0 LPB0
```

In this example, the STOP/QUEUE/NEXT command prevents another job from executing on queue LPB0. The STOP/QUEUE/REQUEUE command requeues the current job running on LPB0 to the target queue LPA0. The ASSIGN/MERGE command removes the remaining jobs from the LPB0 printer queue and places them in the LPA0 printer queue.

ASSIGN/QUEUE

Assigns, or redirects, a logical queue to a single execution queue. ASSIGN/QUEUE can be used only with printer or terminal queues.

Requires OPER privilege or EXECUTE access to both queues.

format

ASSIGN/QUEUE *queue-name[:]* *logical-queue-name[:]*

parameters

queue-name[:]

Name of the execution queue. The queue cannot be a logical queue, a generic queue, or a batch queue.

logical-queue-name[:]

Name of the logical queue.

example

```
$ INITIALIZE/QUEUE/DEFAULT=FLAG=ONE/START LPA0  
$ INITIALIZE/QUEUE TEST_QUEUE  
$ ASSIGN/QUEUE LPA0 TEST_QUEUE  
$ START/QUEUE TEST_QUEUE
```

This example first initializes and starts the printer queue LPA0. The LPA0 queue is set to have a flag page precede each job. The second INITIALIZE/QUEUE command creates the logical queue TEST_QUEUE. The ASSIGN/QUEUE command assigns the logical queue TEST_QUEUE to the printer queue LPA0. The START/QUEUE command starts the logical queue.

ATTACH

Transfers control from your current process (which then hibernates) to the specified process.

The ATTACH and SPAWN commands cannot be used if your terminal has an associated mailbox.

format

ATTACH [*process-name*]

parameter

process-name

Specifies the name of a parent process or spawned subprocess to which control passes. The process must already exist, be part of your current job, and share the same input stream as your current process. However, the process cannot be your current process or a subprocess created with the /NOWAIT qualifier. The process-name parameter is incompatible with the /IDENTIFICATION qualifier.

qualifier

/IDENTIFICATION=pid

Specifies the process identification (PID) of the process to which terminal control will be transferred. Leading zeros can be omitted. The /IDENTIFICATION qualifier is incompatible with the process-name parameter.

example

```
$ ATTACH JONES_2
```

Transfers the terminal's control to the subprocess JONES_2.

BACKUP

Invokes the Backup Utility (BACKUP) to perform one of the following BACKUP operations:

- Make copies of disk files.
- Save disk files as data in a file created by BACKUP on disk or magnetic tape. (Files created by BACKUP are called save sets.)
- Restore disk files from a BACKUP save set.

- Compare disk files or files in a BACKUP save set with other disk files.
- List information about files in a BACKUP save set to an output device or file.

Note that standalone BACKUP cannot be invoked this way, but must be bootstrapped in order to run. For more information about the Backup Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

CALL

Transfers control to a labeled subroutine within a command procedure. The CALL command creates a new procedure level as does the @ (execute procedure) command.

format

CALL *label* [*p1*[*p2*[... *p8*]]]

parameters

label

Specifies a 1- to 255-alphanumeric character label appearing as the first item on a command line. A label may not contain embedded blanks. When the CALL command is executed, control passes to the command following the specified label.

The label can precede or follow the CALL statement in the current command procedure. A label in a command procedure must be terminated with a colon. Labels for subroutines must be unique.

p1 [*p2* [... *p8*]]

Specifies from one to eight optional parameters to pass to the command procedure. Use two consecutive quotation marks ("") to specify a null parameter. The parameters assign character string values to the symbols named P1, P2, and so on in the order of entry, to a maximum of eight. The symbols are local to the specified command procedure. Separate each parameter with one or more blanks.

qualifier

/OUTPUT=*file-spec*

Writes all output to the file or device specified. By default, the output is written to the current SYS\$OUTPUT device and the output file type is LIS. System responses and error messages are written to SYS\$COMMAND as well as to the specified file. If you specify /OUTPUT, the qualifier must immediately follow the CALL command. No wildcard characters are allowed in the output file specification.

DCL-22 DCL Commands

CALL

example

```
$
$! CALL.COM
$
$! Define subroutine SUB1
$!
$ SUB1: SUBROUTINE
    .
    .
    .

$ CALL SUB2 !Invoke SUB2 from within SUB1
    .
    .
    .

$ @FILE !Invoke another procedure command file
    .
    .
    .

$ EXIT
$ ENDSUBROUTINE !End of SUB1 definition
$!
$! Define subroutine SUB2
$!
$ SUB2: SUBROUTINE
    .
    .
    .
```

```
$ EXIT
$ ENDSUBROUTINE !End of SUB2 definition
$!
$! Start of main routine. At this point, both SUB1 and SUB2
$! have been defined but none of the previous commands have
$! been executed.
$!
$ START:
$ CALL/OUTPUT= NAMES.LOG SUB1 "THIS IS P1"
```

```
.
.
.
```

```
$ CALL SUB2 "THIS IS P1" "THIS IS P2"
```

```
.
.
.
```

```
$ EXIT !Exit this command procedure file
```

The command procedure in this example shows how to use CALL to transfer control to labeled subroutines. The example also shows that you can call a subroutine or another command file from within a subroutine. The CALL command invokes the subroutine SUB1, directing output to the file NAMES.LOG and allowing other users write access to the file. The subroutine SUB2 is called from within SUB1. The procedure executes SUB2 and then uses the @ (Execute Procedure) command to invoke the command procedure FILE.COM. When all the commands in SUB1 have executed, the CALL command in the main procedure calls SUB2 a second time. The procedure continues until SUB2 has executed.

CANCEL

Cancels wakeup requests for a specified process, including wakeups scheduled with either the RUN command or the \$SCHDWK system service.

Requires one of the following:

- **Ownership of the process.**
- **GROUP privilege to cancel scheduled wakeups for processes in the same group but not owned by you.**
- **WORLD privilege to cancel scheduled wakeups for any process in the system.**

DCL-24 DCL Commands

CANCEL

format

CANCEL [*process-name*]

parameter

process-name

Specifies the name of the process for which wakeup requests are to be canceled. Process names can be up to 23 alphanumeric characters in the following format:

[*node-name::*]process-name

- The node name can have as many as 6 alphanumeric characters.
- The colons count for 2 characters.
- The process name can have as many as 15 characters.

A local process name can look like a remote process name. Therefore, if you specify ATHENS::SMITH, the system checks for a process named ATHENS::SMITH on the local node before checking node ATHENS for a process named SMITH.

The specified process must have the same group number in its user identification code (UIC) as the current process. If both the /IDENTIFICATION qualifier and the process name are specified, the process name is ignored. If neither the process-name parameter nor the /IDENTIFICATION qualifier are specified, the CANCEL command cancels scheduled wakeup requests for the current (that is, the issuing) process.

qualifier

/IDENTIFICATION=pid

Identifies the process by its process identification (PID). You can omit leading zeros when you specify the PID.

example

```
$ RUN/SCHEDULE=14:00 STATUS
%RUN-S-PROC_ID, identification of created process is 0013012A
```

```
.
.
.
```

```
$ CANCEL/IDENTIFICATION=13012A
```

The RUN command in this example creates a process to execute the image STATUS. The process hibernates and is scheduled to be awakened at 14:00. Before the process is awakened, the CANCEL command cancels the wake-up request.

CLOSE

Closes a file opened with the OPEN command and deassigns the associated logical name.

format

CLOSE *logical-name[:]*

parameter

logical-name[:]

Specifies the logical name assigned to the file when it was opened with the OPEN command.

qualifiers

/ERROR=label

Specifies a label in the command procedure to receive control if the CLOSE operation results in an error. Overrides any ON condition action specified. If an error occurs and the target label is successfully given control, the global symbol \$STATUS retains the code for the error that caused the error path to be taken.

/LOG (default)

/NOLOG

Generates a warning message when you attempt to close a file that was not opened by DCL. If you specify the /ERROR qualifier, the /LOG qualifier has no effect. If the file has not been opened by DCL, the error branch is taken and no message is displayed.

example

```
$ OPEN/READ INPUT_FILE TEST.DAT
$ READ_LOOP:
$ READ/END_OF_FILE=NO_MORE INPUT_FILE DATA_LINE
```

.
.
.

DCL-26 DCL Commands

CLOSE

```
$ GOTO READ_LOOP  
$ NO_MORE:  
$ CLOSE INPUT_FILE
```

The OPEN command in this example opens the file TEST.DAT and assigns it the logical name of INPUT_FILE. The /END_OF_FILE qualifier on the READ command requests that, when the end-of-file is reached, the command interpreter should transfer control to the line at the label NO_MORE. The CLOSE command closes the input file.

CONNECT

Connects your physical terminal to a virtual terminal that is connected to another process.

You must connect to a virtual terminal that is connected to a process with your user identification code (UIC). No other physical terminals may be connected to the virtual terminal.

format

```
CONNECT virtual-terminal-name
```

parameter

virtual-terminal-name

Specifies the name of the virtual terminal to which you are connecting. A virtual terminal name always begins with VTA. To determine the name of the virtual terminal that is connected to a process, enter the SHOW USERS command.

qualifiers

/CONTINUE

/NOCONTINUE (default)

Controls whether the CONTINUE command is executed in the current process just before connecting to another process. This allows an interrupted image to continue processing after you connect to another process. The /CONTINUE qualifier is incompatible with the /LOGOUT qualifier.

/LOGOUT (default)

/NOLOGOUT

Logs out your current process when you connect to another process using a virtual terminal. The /LOGOUT qualifier is incompatible with the /CONTINUE qualifier.

example

```
$ RUN AVERAGE  
[CTRL/Y]  
$ CONNECT/CONTINUE VTA72
```

In this example, you use the RUN command to execute the image AVERAGE.EXE. You enter this command from a terminal that is connected to a virtual terminal. Next, you enter CTRL/Y to interrupt the image. After you interrupt the image, enter the CONNECT command with the /CONTINUE qualifier. This issues the CONTINUE command, so the image continues to run and connects you to another virtual terminal. You can reconnect to the process later.

CONTINUE

Resumes execution of a DCL command, a program, or a command procedure that was interrupted by CTRL/Y or CTRL/C. You cannot resume execution of the image if you have entered a command that executes another image or if you have invoked a command procedure. You can abbreviate the CONTINUE command to a single letter, C.

format

CONTINUE

parameters

None.

example

```
$ RUN MYPROGRAM_A  
[CTRL/Y]  
$ SHOW TIME  
19-APR-1990 13:40:12  
$ CONTINUE
```

In this example, the RUN command executes the program MYPROGRAM_A. While the program is running, pressing CTRL/Y interrupts the image. The SHOW TIME command requests a display of the current date and time. The CONTINUE command resumes the image.

CONVERT

Invokes the Convert Utility (CONVERT) to copy records from one file to another, changing the organization and format of the input file to those of the output file.

format

CONVERT *input-file-spec[,...] output-file-spec*

CONVERT/DOCUMENT

Invokes the CDA Converter to translate a revisable format file to another revisable or final form file from the DCL command line. Please note that you can use this command only if you have DECwindows installed on your system. For a complete description of the conversion process including more information about the CONVERT/DOCUMENT command and its qualifiers, see the *VMS Compound Document Architecture Manual*.

format

CONVERT/DOCUMENT *input-file output-file*

CONVERT/RECLAIM

Invokes the Convert/Reclaim Utility (CONVERT/RECLAIM) to make empty buckets in Prolog 3 indexed files available so that new records can be written in them. If all the records in a bucket have been deleted, that bucket is locked until CONVERT/RECLAIM makes it available. Unlike CONVERT, CONVERT/RECLAIM maintains record file addresses (RFAs). The /RECLAIM qualifier is required.

format

CONVERT/RECLAIM *file-spec*

COPY

Creates a new file from one or more existing files. If you do not specify the device or directory, the COPY command uses your current default device and directory.

format

COPY *input-file-spec[,...]* *output-file-spec*

parameters

input-file-spec[,...]

Specifies the name of an existing file to be copied. Wildcard characters are allowed. Use a plus sign (+) or a comma (,) to indicate multiple file specifications.

output-file-spec

Specifies the name of the output file into which the input is copied. You must specify at least one field in the output file specification. If you do not specify the device or directory, the COPY command uses your current default device and directory. You can use the asterisk wildcard character in place of any two of the following: the file name, file type, or version number.

description

When you specify multiple input and output files you can use the /LOG qualifier to verify that the files were copied as you intended.

Note that there are special considerations for using the COPY command with DECwindows compound documents. For more information, see the *Guide to VMS File Applications*.

Version Numbers

If you do not specify version numbers for input and output files, the COPY command (by default) assigns a version number to the output files that is either of the following:

- The version number of the input file
- A version number one greater than the highest version number of an existing file with the same file name and file type

When you specify the output file version number by an asterisk wildcard character, the COPY command uses the version numbers of the associated input files as the version numbers of the output files.

If you specify the output file version number by an explicit version number, the COPY command uses that number for the output file specification. If a higher version of the output file exists, the COPY command issues a warning message and copies the file. If an equal version of the output file exists, the COPY command issues a message and does *not* copy the input file.

File Protection and Creation/Revision Dates

The COPY command considers an output file to be new when you specify any portion of the output file name explicitly. The COPY command sets the creation date for a new file to the current time and date.

If you specify the output file by one or more wildcard characters, the COPY command uses the creation date of the input file.

The COPY command always sets the revision date of the output file to the current time and date; it sets the backup date to zero. The file system assigns the output file a new expiration date. (The file system sets expiration dates if retention is enabled; otherwise it sets expiration dates to zero.)

The protection and access control list (ACL) of the output file is determined by the following parameters, in the following order:

- Protection of previously existing versions of the output file
- Default Protection and ACL of the output directory
- Process default file protection

(Note that the BACKUP command takes the creation and revision dates as well as the file protection from the input file.)

Use the /PROTECTION qualifier to change the output file protection.

Normally, the owner of the output file will be the same as the creator of the output file. However, if a user with extended privileges creates the output file, the owner will be the owner of the parent directory or of a previous version of the output file if one exists.

Extended privileges include any of the following:

- SYSPRV or BYPASS
- System UIC
- GRPPRV if the owner of the parent directory (or previous version of the output file) is in the same group as the creator of the new output file
- An identifier (with the resource attribute) representing the owner of the parent directory (or the previous version of the output file)

Copying Directory Files

If you copy a file that is a directory, the COPY command creates a new *empty* subdirectory of the named directory. The COPY command does *not* copy any files from the named directory to the new subdirectory. For example:

```
$ COPY [SMITH]CATS.DIR [JONES]
```

This COPY command creates the new empty subdirectory [JONES]CATS.DIR. Once the COPY command creates the new subdirectory [JONES]CATS.DIR, you can copy the files in the directory [SMITH]CATS.DIR.

qualifiers

/ALLOCATION=*n*

Output-file-spec qualifier. Forces the initial allocation of the output file to the number of 512-byte blocks specified by *n*. If you do not specify the number of 512-byte blocks, the size of the input file being copied determines the initial allocation of the output file.

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=*time*]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. To indicate the time attribute to be used as the basis for selection, specify one of the following qualifiers with /BEFORE: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=*uic*]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONCATENATE (default)

/NOCONCATENATE

Creates one output file from multiple input files when you do not use wildcard characters in the output file specification. The /NOCONCATENATE qualifier generates multiple output files. Files from Files-11 Structure Level 2 disks are concatenated in alphanumeric order; if you specify a wildcard in the file version field, files are copied in descending order by version number. Files from Files-11 Structure Level 1 disks are concatenated in random order.

DCL-32 DCL Commands

COPY

/CONFIRM

/NOCONFIRM (default)

Controls whether a request is issued before each COPY operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL

RET

You can use any combination of uppercase and lowercase letters for word responses. You can abbreviate word responses to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CONTIGUOUS

/NOCONTIGUOUS

Output-file-spec qualifier. Specifies that the output file must occupy contiguous physical disk blocks. By default, the COPY command creates an output file in the same format as the corresponding input file. Also, by default, if not enough space exists for a contiguous allocation, the COPY command does not report an error.

The /CONTIGUOUS qualifier has no effect when you copy files to or from tapes because the size of the file on tape cannot be determined until after it is copied to the disk. If you copy a file from a tape and want the file to be contiguous, use the COPY command twice: once to copy the file from the tape, and a second time to create a contiguous file.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. The /CREATED qualifier selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the COPY operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifiers. */EXPIRED* selects files according to their expiration dates. (You set the expiration date with the *SET FILE/EXPIRATION_DATE* command.) The */EXPIRED* qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

/EXTENSION=n

Output-file-spec qualifier. Specifies the number of blocks to be added to the output file each time the file is extended.

/LOG

/NOLOG (default)

Controls whether the *COPY* command displays the file specifications of each file copied.

When you use the */LOG* qualifier, the *COPY* command displays the following for each copy operation: (1) the file specifications of the input and output files, (2) the number of blocks or the number of records copied (depending on whether the file is copied on a block-by-block or record-by-record basis), and (3) the total number of new files created.

/MODIFIED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. The */MODIFIED* qualifier selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/OVERLAY

/NOOVERLAY (default)

Output-file-spec qualifier. Requests that data in the input file be copied into the existing specified file, overlaying the existing data, rather than allocating new space for the file. The physical location of the file on disk does not change.

/PROTECTION=(code)

Output-file-spec qualifier. Specifies protection for the output file. Specify ownership as *SYSTEM*, *OWNER*, *GROUP*, or *WORLD* and access as *R* (read), *W* (write), *E* (execute), or *D* (delete). The protection of the existing output file is the default. If no output file exists, the current default protection applies.

/READ_CHECK

/NOREAD_CHECK (default)

Input-file-spec qualifier. Reads each record in the input files twice to verify that it has been read correctly.

DCL-34 DCL Commands

COPY

/REPLACE

/NOREPLACE (default)

Output-file-spec qualifier. Requests that, if a file exists with the same file specification as that entered for the output file, the existing file is to be deleted. The COPY command allocates new space for the output file. In general, when you use the /REPLACE qualifier, include version numbers with the file specifications. By default, the COPY command creates a new version of a file if a file with that specification exists, incrementing the version number. The /NOREPLACE qualifier signals an error when a conflict in version numbers occurs.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/TRUNCATE

/NOTRUNCATE (default)

Output-file-spec qualifier. Controls whether or not the COPY command truncates an output file at the end-of-file when copying it. By default, the allocation of the input file determines the size of the output file.

/VOLUME=n

Output-file-spec qualifier. Places the output file on the specified relative volume number of a multivolume set. By default, the COPY command places the output file arbitrarily in a multivolume set.

/WRITE_CHECK

/NOWRITE_CHECK (default)

Output-file-spec qualifier. Reads each record in the output file after it was written to verify that the record was copied successfully and that the file can be read subsequently without error.

example

```
$ COPY/LOG A.DAT,B.MEM C.*
%COPY-S-COPIED, DBA0:[MAL]A.DAT;5 copied to DBA0:[MAL]C.DAT;11 (1 block)
%COPY-S-COPIED, DBA0:[MAL]B.MEM;2 copied to DBA0:[MAL]C.MEM;24 (58 records)
%COPY-S-NEWFILES, 2 files created
```

In this example, the two input file specifications are separated with a comma. The asterisk wildcard character in the output file specification indicates that two output files are to be created. For each copy operation, the COPY command uses the file type of the input file to name the output file.

CREATE

Creates a sequential text file (or files). Specify the content of the file on the lines following the command, one record per line. In interactive mode, terminate the file input with CTRL/Z. In a command procedure, terminate the file input with a line beginning with a dollar sign in column 1 (or with the end of the command procedure).

format

CREATE *file-spec[,...]*

parameter

file-spec[,...]

Specifies the name of one or more input files to be created. Wildcard characters are not allowed. If you omit either the file name or the file type, the CREATE command does not supply any defaults. The file name or file type is null. If the specified file already exists, a new version is created.

qualifiers

/LOG

/NOLOG (default)

Displays the file specification of each new file created as the command executes.

/OWNER_UIC=*uic*

Requires SYSPRV privilege to specify a UIC other than your own. Specifies the user identification code (UIC) to be associated with the file being created.

/PROTECTION=(*code*)

Specifies protection for the file. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and protection as R (read), W (write), E (execute), or D (delete). If you do not specify a value for each access category, the command applies the current default protection for each unspecified category.

/VOLUME=*n*

Places the file on the specified relative volume of a multivolume set. By default, the file is placed arbitrarily in a multivolume set.

DCL-36 DCL Commands

CREATE

example

```
$ CREATE MEET.TXT
```

```
John, Residents in the apartment complex will hold their annual meeting  
this evening. We hope to see you there, Regards, Elwood
```

```
CTRL/Z
```

The CREATE command in this example creates a text file named MEET.TXT in your default directory. The text file MEET.TXT contains the lines that follow until the CTRL/Z.

CREATE/DIRECTORY

Creates one or more new directories or subdirectories. The /DIRECTORY qualifier is required.

Requires WRITE (W) access to the master file directory (MFD) to create a first-level directory. Requires WRITE access to the lowest level directory that currently exists to create a subdirectory.

format

```
CREATE/DIRECTORY directory-spec[,...]
```

parameter

directory-spec[,...]

Specifies the name of one or more directories or subdirectories to be created. The directory specification optionally can be preceded by a device name (and colon). The default is the current default directory. Wildcard characters are not allowed. When creating a subdirectory, separate the names of the directory levels with periods.

qualifiers

/LOG

/NOLOG (default)

Controls whether the CREATE/DIRECTORY command displays the directory specification of each directory after creating it.

/OWNER_UIC[=*option*]

Requires SYSPRV privilege for a UIC (user identification code) other than your own.

Specifies an owner UIC for the directory. The default is your UIC. You can specify the keyword PARENT in place of a UIC to mean the UIC of the parent (next-higher-level) directory. If a user with privileges creates a subdirectory, by default, the owner of the subdirectory will be the owner of the parent directory (or the owner of the Master File Directory, if creating

a main level directory). If you do not specify the /OWNER_UIC qualifier when creating a directory, the command assigns ownership as follows: (1) if you specify the directory name in either alphanumeric or subdirectory format, the default is your UIC (unless you are privileged in which case the UIC defaults to the parent directory); (2) if you specify the directory in UIC format, the default is the specified UIC.

/PROTECTION=(code)

Specifies protection for the directory. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and protection as R (read), W (write), E (execute), or D (delete). The default protection is the protection of the parent directory (the next-higher level directory, or the master directory for top-level directories) minus any delete access.

/VERSION_LIMIT=n

Specifies the number of versions of any one file that can exist in the directory. If you exceed the limit, the system deletes the lowest numbered version. A specification of 0 means no limit. The maximum number of versions allowed is 32,767. The default is the limit for the parent (next-higher-level) directory.

/VOLUME=n

Requests that the directory file be placed on the specified relative volume of a multivolume set. By default, the file is placed arbitrarily within the multivolume set.

example

```
$ CREATE/DIRECTORY/VERSION_LIMIT=2 $DISK1:[ACCOUNTS.MEMOS]
```

In this example, the CREATE/DIRECTORY command creates a subdirectory named MEMOS in the ACCOUNTS directory on \$DISK1. No more than two versions of each file can exist in the directory.

CREATE/FDL

Invokes the Create/FDL Utility (CREATE/FDL) to use the specifications in an FDL file to create a new, empty data file. Use this utility to create a data file from a particular FDL specification. The /FDL qualifier is required.

format

```
CREATE/FDL =fdl-file-spec [file-spec]
```

CREATE/NAME_TABLE

Creates a new logical name table.

format

CREATE/NAME_TABLE *table-name*

parameter

table-name

Specifies a string of 1 to 31 characters that identifies the logical name table you are creating. The string can include alphanumeric characters, the dollar sign, and the underscore.

qualifiers

/ATTRIBUTES[(keyword[,...])]

Specifies attributes for the logical name table. If you specify only one keyword, you can omit the parentheses. If you do not specify the **/ATTRIBUTES** qualifier, no attributes are set.

You can specify the following keywords for attributes:

- | | |
|------------------|--|
| CONFINE | Does not copy the table name or the logical names contained in the table into a spawned subprocess; used only when creating a private logical name table. |
| NO_ALIAS | No identical names (either logical names or names of logical name tables) may be created in an outer (less privileged) mode in the current directory. Deletes any previously created identical table names in an outer access mode in the same logical name table directory. |
| SUPERSEDE | Creates a new table that supersedes any previous (existing) table that contains the name, access mode, and directory table that you specify. |

/EXECUTIVE_MODE

Requires SYSNAM privilege. Creates an executive mode logical name table.

/LOG (default)

/NOLOG

Controls whether or not an informational message is generated when the **SUPERSEDE** attribute is specified, or when the table already exists but the **SUPERSEDE** attribute is not specified. The default is **/LOG**; that is, the informational message is displayed.

/PARENT_TABLE=table

Requires EXECUTE (E) access to the parent table and SYSPRV privilege to create a shareable logical name table. Specifies the name of the parent table. If you do not specify a parent table, the default table is **LNMS\$PROCESS_DIRECTORY**. A shareable table has **LNMS\$SYSTEM_DIRECTORY** as its parent table. The parent table must

have the same access mode or a higher-level access mode than the one you are creating.

/PROTECTION

Applies the specified protection to shareable name tables. The ownership categories are SYSTEM, OWNER, GROUP, WORLD; the access categories are R (READ), W (WRITE), E (EXECUTE) and D (DELETE). The default protection is (SYSTEM:RWED,OWNER:RWED,GROUP:,WORLD:)

/QUOTA=number-of-bytes

Specifies the size limit of the logical name table. If you do not specify the /QUOTA qualifier, or if you specify /QUOTA=0, the table has unlimited quota.

/SUPERVISOR_MODE (default)

Creates a supervisor mode logical name table. If you do not specify a mode, a supervisor mode logical name table is created.

/USER_MODE

Creates a user mode logical name table. If you do not explicitly specify a mode, a supervisor mode logical name table is created.

example

```
$ CREATE/NAME_TABLE TEST_TAB
$ SHOW LOGICAL TEST_TAB
%SHOW-S-NOTRAN, no translation for logical name TEST_TAB
$ SHOW LOGICAL/TABLE=LNMS$PROCESS_DIRECTORY TEST_TAB
```

In this example, the CREATE/NAME_TABLE command creates a new table called TEST_TAB. By default, the name of the table is entered in the process directory. The first SHOW LOGICAL command does not find the name TEST_TAB because it does not, by default, search the process directory table. You must use the /TABLE qualifier to request that the process directory be searched.

DEALLOCATE

Makes an allocated device available to other processes (but does not deassign any logical name associated with the device).

format

DEALLOCATE *device-name[:]*

parameter

device-name[:]

Name of the device to be deallocated. The device name can be a physical device name or a logical name. On a physical device name, the controller defaults to A and the unit to 0. Incompatible with the /ALL qualifier.

DCL-40 DCL Commands

DEALLOCATE

qualifier

/ALL

Deallocates all devices currently allocated by your process. Incompatible with the device-name parameter.

example

```
$ ALLOCATE MT: TAPE
%DCL-I-ALLOC, _MTB1: allocated
```

.
. .
. . .

```
$ DEALLOCATE TAPE:
```

In this example, the ALLOCATE command requests that any magnetic tape drive be allocated and assigns the logical name TAPE to the device. The response to the ALLOCATE command indicates the successful allocation of the device MTB1. The DEALLOCATE command specifies the logical name TAPE to release the tape drive.

DEASSIGN

Cancels logical name assignments made with the ALLOCATE, ASSIGN, DEFINE, or MOUNT command. The DEASSIGN command also deletes logical name tables created with the CREATE/NAME_TABLE command. Logical names in private tables are deleted automatically when your process terminates. All logical names in the job table and the job table itself are deleted when your process terminates. User mode logical names in the process table are deleted automatically when the next image exits. All other logical names in shareable tables remain unless explicitly deassigned. All names in descendant tables are deleted when the parent table logical name is deassigned.

format

```
DEASSIGN [logical-name[:]]
```

parameter

logical-name[:]

Specifies the logical name to be deassigned. Logical names can have from 1 to 255 characters. If the logical name contains any characters other than alphanumeric, dollar signs, or underscores, enclose it in quotation marks. The logical-name parameter is required unless you use the /ALL qualifier. If a colon is present in the logical name, you must type two

colons in the logical-name parameter of the DEASSIGN command (for example, DEASSIGN FILE:).

qualifiers

/ALL

Deletes all logical names in the same or an outer (less privileged) access mode. If no logical name table is specified, the default is the process table, LNM\$PROCESS. If you specify /ALL, you cannot enter a logical-name parameter.

/EXECUTIVE_MODE

Requires SYSNAM privilege to deassign executive mode logical names. Deletes only entries that were created in the specified mode or an outer (less privileged) mode. If you do not have SYSPRV privilege for executive mode, a supervisor mode operation is assumed.

/GROUP

Requires GRPNAM or SYSPRV privilege to delete entries from the group logical name table. Indicates that the specified logical name is in the group logical name table. The /GROUP qualifier is synonymous with /TABLE=LNM\$GROUP.

/JOB

Indicates that the specified logical name is in the jobwide logical name table. The /JOB qualifier is synonymous with /TABLE=LNM\$JOB. If you do not explicitly specify a logical name table, the default is /PROCESS.

/PROCESS (default)

Indicates that the specified logical name is in the process logical name table. The /PROCESS qualifier is synonymous with /TABLE=LNM\$PROCESS.

/SUPERVISOR_MODE (default)

Deletes entries in the specified logical name table that were created in supervisor mode. If you specify the /SUPERVISOR_MODE qualifier, the DEASSIGN command also deassigns user mode entries with the same name.

/SYSTEM

Requires SYSNAM or SYSPRV privilege to delete entries from the system logical name table. Indicates that the specified logical name is in the system logical name table. The /SYSTEM qualifier is synonymous with /TABLE=LNM\$SYSTEM.

/TABLE=name

Requires WRITE (W) access to the table to delete a shareable logical name. Requires SYSPRV or DELETE (D) access to delete a shareable logical name table. Specifies the table from which the logical name is to be deleted. Defaults to LNM\$PROCESS. The table can

DCL-42 DCL Commands

DEASSIGN

be the process, group, job, or system table, one of the directory tables, or the name of a user-created table.

/USER_MODE

Deletes entries in the process logical name table that were created in user mode. If you specify the **/USER_MODE** qualifier, the **DEASSIGN** command can deassign only user mode entries.

example

```
$ DEASSIGN/TABLE=LNMS$PROCESS_DIRECTORY TAX
```

The **DEASSIGN** command in this example deletes the logical name table **TAX**, and any descendant tables. When you delete a logical name table, you must specify either **/TABLE=LNMS\$PROCESS_DIRECTORY** or **/TABLE=LNMS\$SYSTEM_DIRECTORY**, because the names of all tables are contained in these directories.

DEASSIGN/QUEUE

Deassigns a logical queue from a printer or terminal queue and stops the logical queue. The **DEASSIGN/QUEUE** command is the complement of the **ASSIGN/QUEUE** command.

Requires OPER privilege or EXECUTE access to the queue.
Cannot be used with batch queues.

format

```
DEASSIGN/QUEUE logical-queue-name[:]
```

parameter

logical-queue-name[:]

Specifies the name of the logical queue that you want to deassign from a specific printer or terminal queue.

example

```
$ ASSIGN/QUEUE LPA0 ASTER
```

```
.  
. .  
. . .
```

```
$ DEASSIGN/QUEUE ASTER  
$ ASSIGN/MERGE LPB0 ASTER
```

The **ASSIGN/QUEUE** command in this example associates the logical queue **ASTER** with the print queue **LPA0**. Later, you deassign the logical

queue with the DEASSIGN/QUEUE command. The ASSIGN/MERGE command reassigns the jobs from ASTER to the print queue LPB0.

DEBUG

Invokes the VMS Debugger after program execution is interrupted by CTRL/Y, but only if the /NOTRACEBACK qualifier was not specified with the LINK command when the program was linked.

format

DEBUG

DECK

Marks the beginning of an input stream for a command or program. The DECK command is required in command procedures when the first nonblank character in any data record in the stream is a dollar sign.

Can be used only after a request to execute a command or program that requires input data.

format

DECK

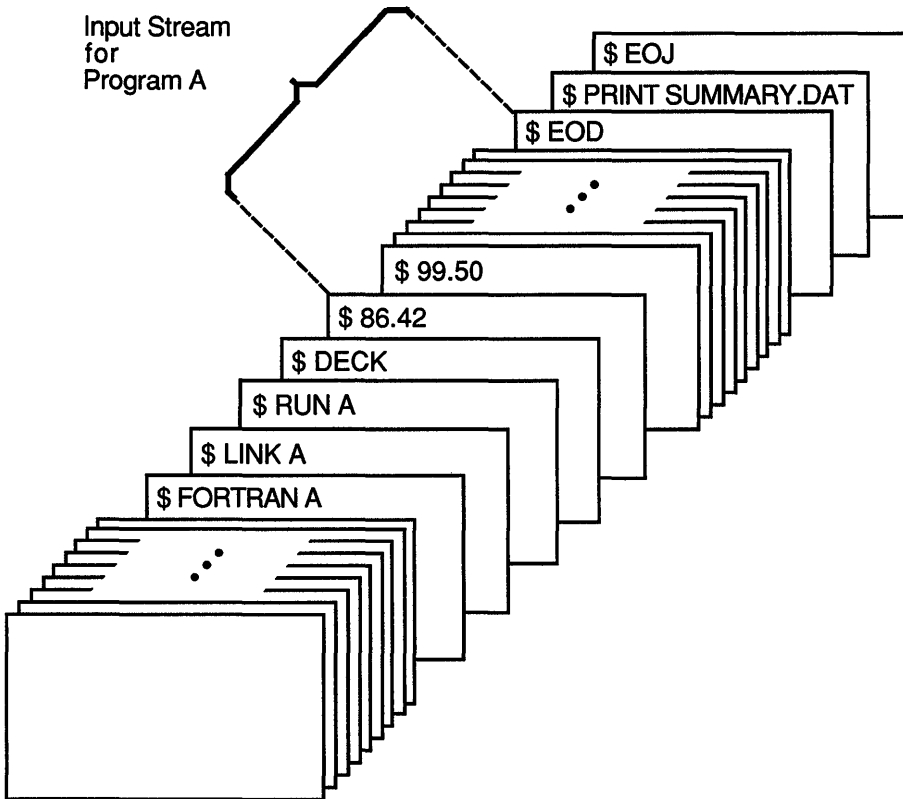
qualifier

/DOLLARS[=string]

Sets the end-of-file indicator to the specified string of 1 through 15 characters. Enclose the string in quotation marks if it contains literal lowercase letters, multiple blanks, or tabs. If you do not specify /DOLLARS, or if you specify /DOLLARS without specifying a string, you must use the EOD command to signal the end-of-file.

**DCL-44 DCL Commands
DECK**

example



ZK-0783-GE

In this example, the FORTRAN and LINK commands compile and link program A. When the program is run, any data the program reads from the logical device SYS\$INPUT is read from the command stream. The DECK command indicates that the input stream can contain dollar signs in column 1 of the record. The EOD command signals end-of-file for the data.

DEFINE

Associates equivalence names with a logical name. If you specify an existing logical name, the new equivalence names replace the existing equivalence name.

format

DEFINE *logical-name equivalence-name[,...]*

parameters

logical-name

Specifies the logical name string, which is a character string containing from 1 to 255 characters. If the logical name is to be entered into the process or system directory logical name tables (LNM\$PROCESS_DIRECTORY, LNM\$SYSTEM_DIRECTORY), then the name may only have from 1 to 31 alphanumeric characters (including the dollar sign and underscore). If the string contains any characters other than uppercase alphanumerics, the dollar sign, or the underscore character, enclose the string in quotation marks ("). Use two consecutive quotation marks ("") to denote an actual quotation mark.

equivalence-name[,...]

Specifies a character string containing from 1 to 255 characters. If the string contains any characters other than uppercase alphanumerics, the dollar sign, or the underscore character, enclose the string in quotation marks. Use two consecutive quotation marks ("") to denote an actual quotation mark. Specifying more than one equivalence name for a logical name creates a search list.

qualifiers

/EXECUTIVE_MODE

Requires SYSNAM privilege to create an executive mode logical name. Creates an executive mode logical name in the specified table.

If you specify the */EXECUTIVE_MODE* qualifier and you do not have SYSNAM, the DEFINE command ignores the qualifier and creates a supervisor mode logical name. The mode of the logical name must be the same or less privileged than the mode of the table in which you are placing the name.

/GROUP

Requires GRPNAM or SYSPRV privilege to place a name in the group logical name table. Places the logical name in the group logical name table. The */GROUP* qualifier is synonymous with */TABLE=LNM\$GROUP*.

/JOB

Places the logical name in the jobwide logical name table. The ***/JOB*** qualifier is synonymous with ***/TABLE=LNМ\$JOB***.

/LOG (default)

/NOLOG

Displays a message when a new logical name supersedes an existing name.

/NAME_ATTRIBUTES[(keyword[,...])]

Specifies attributes for a logical name. By default, no attributes are set. Possible keywords are as follows:

- | | |
|-----------------|---|
| CONFINE | The logical name is not copied into a spawned subprocess. This qualifier is relevant only for logical names in a private table. |
| NO_ALIAS | A logical name cannot be duplicated in the specified table in a less privileged access mode; any previously created identical names in an outer (less privileged) access mode within the specified table are deleted. |

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

/PROCESS (default)

Places the logical name in the process logical name table. The ***/PROCESS*** qualifier is synonymous with ***/TABLE=LNМ\$PROCESS***.

/SUPERVISOR_MODE (default)

Creates a supervisor mode logical name in the specified table. The mode of the logical name must be the same as or less privileged than the mode of the table in which you are placing the name.

/SYSTEM

Requires SYSNAM or SYSPRV privilege to place a name in the system logical name table. Places the logical name in the system logical name table. The ***/SYSTEM*** qualifier is synonymous with ***/TABLE=LNМ\$SYSTEM***.

/TABLE=name

Requires WRITE (W) access to the table to specify the name of a shareable logical name table. Specifies the name of the logical name table in which the logical name is to be entered. You can use the ***/TABLE*** qualifier to specify a user-defined logical name table (created with the ***CREATE/NAME_TABLE*** command); to specify the process, job, group, or system logical name tables; or to specify the process or system logical name directory tables.

If you specify the table name using a logical name that has more than one translation, the logical name is placed in the first table found. The default is ***/TABLE=LNМ\$PROCESS*** (or ***/PROCESS***).

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

Equivalence-name qualifier. Specifies one or more attributes that modify an equivalence string of the logical name. Possible keywords are as follows:

- | | |
|------------------|--|
| CONCEALED | Indicates that the equivalence string is the name of a concealed device. |
| TERMINAL | Logical name translation should terminate with the current equivalence string; indicates that the equivalence string should not be translated iteratively. |

/USER_MODE

Creates a user mode logical name in the specified table. User mode logical names created within the process logical name tables are used for the execution of a single image.

example

```
$ DEFINE/USER_MODE TM1 $DISK1:[ACCOUNTS.MEMOS]WATER.TXT
```

In this example, the **DEFINE** command defines **TM1** as equivalent to a file specification. After the next image runs, the logical name **TM1** is automatically deassigned.

DEFINE/CHARACTERISTIC

Assigns a numeric value to a queue characteristic. If a value has been assigned to the characteristic, the **DEFINE/CHARACTERISTIC** command alters the assignment of the existing characteristic. The **/CHARACTERISTIC** qualifier is required.

Requires OPER privilege.

format

DEFINE/CHARACTERISTIC *characteristic-name characteristic-number*

parameters

characteristic-name

Assigns a name to the characteristic being defined. The characteristic name can be the name of an existing characteristic or a string of 1 to 31 characters that defines a new characteristic. The character string can include any uppercase and lowercase letters, digits, the dollar sign (\$), and the underscore (_), and must include at least one alphabetic character.

characteristic-number

Assigns a number in the range 0 through 127 to the characteristic being defined.

DCL-48 DCL Commands

DEFINE/CHARACTERISTIC

example

```
$ DEFINE/CHARACTERISTIC REDINK 3
```

The DEFINE/CHARACTERISTIC command in this example defines the characteristic REDINK with the number 3. When a user enters the command PRINT/CHARACTERISTICS=REDINK (or PRINT/CHARACTERISTICS=3), the job is printed only if the printer queue has been established with the REDINK or 3 characteristic.

DEFINE/FORM

Assigns a numeric value and attributes to a print form name. If a value has been assigned already to the form name, the DEFINE/FORM command alters the definition of the existing form. The /FORM qualifier is required.

Requires OPER privilege.

format

```
DEFINE/FORM form-name form-number
```

parameters

form-name

Assigns a name to the form being defined. The form name can be the name of an existing form type or a string of 1 to 31 characters that defines a new form type. The character string can include any uppercase and lowercase letters, digits, the dollar sign (\$), and the underscore (_), and must include at least one alphabetic character.

form-number

Assigns a number in the range 0 through 999,999,999 to the form being defined. The DEFAULT form, which is defined automatically when the system is bootstrapped, is assigned number 0.

qualifiers

/DESCRIPTION=string

A string of up to 255 characters used to provide operator-information about the form. The default string is the specified form name. Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (").

/LENGTH=n

Specifies the physical length of a form page in lines. The default page length is 66 lines. The n parameter must be a positive integer greater than 0 and not more than 255.

/MARGIN=(option[,...])

Specifies one or more of the four margin options: BOTTOM, LEFT, RIGHT, and TOP.

- BOTTOM=n** Specifies the number of blank lines between the end of the print image area and the end of the physical page; the value of n must be between 0 and the value of the /LENGTH parameter. The default value is 6, which generally means a one-inch bottom margin.
- LEFT=n** Specifies the number of blank columns between the leftmost printing position and the print image area; the value of n must be between 0 and the value of the /WIDTH parameter. The default is 0.
- RIGHT=n** Specifies the number of blank columns between the /WIDTH parameter and the image area; the value of n must be between 0 and the value of the /WIDTH parameter. The default value is 0.
- TOP=n** Specifies the number of blank lines between the top of the physical page and the top of the print image; the value of n must be between 0 and the value of the /LENGTH parameter. The default value is 0.

/PAGE_SETUP=(module[,...])

/NOPAGE_SETUP (default)

Specifies one or more modules that set up the device at the start of each page. The modules are located in the device control library. While the form is mounted, the system extracts the specified module and copies it to the printer before each page is printed.

/SETUP=(module[,...])

Specifies one or more modules that set up the device at the start of each file. The modules are located in the device control library. While the form is mounted, the system extracts the specified module and copies it to the printer before each file is printed.

/SHEET_FEED

/NOSHEET_FEED (default)

Specifies that print jobs pause at the end of every physical page so that a new sheet of paper can be inserted.

/STOCK=string

Specifies the type of paper stock to be associated with the form. The string parameter can be a string of 1 to 31 characters, including the dollar sign, underscore, and all alphanumeric characters. If you specify the /STOCK qualifier you must specify the name of the stock to be associated with the form. If you do not specify the /STOCK qualifier, the name of the stock will be the same as the name of the form.

/TRUNCATE (default)

/NOTRUNCATE

Discards any characters that exceed the current line length (specified by /WIDTH and /MARGIN=RIGHT). /TRUNCATE is incompatible with the

DCL-50 DCL Commands

DEFINE/FORM

/WRAP qualifier. If you specify both **/NOTRUNCATE** and **/NOWRAP**, the printer prints as many characters on a line as possible.

/WIDTH=n

Specifies the physical width of the paper in terms of columns or character positions. The *n* parameter must be an integer from 0 through 65,535; the default value is 132. The **/MARGIN=RIGHT** qualifier overrides the **/WIDTH** qualifier when determining when to wrap lines of text.

/WRAP

/NOWRAP (default)

Causes lines that exceed the current line length (specified by **/WIDTH** and **/MARGIN=RIGHT**) to wrap onto the next line. **/WRAP** is incompatible with the **/TRUNCATE** qualifier. If you specify both **/NOWRAP** and **/NOTRUNCATE**, the printer prints as many characters on a line as possible.

example

```
$ DEFINE/FORM /MARGIN=(TOP=6,LEFT=10) CENTER 3
```

The **DEFINE/FORM** command in this example defines the form **CENTER** to have a top margin of 6 and a left margin of 10. The form is assigned the number 3.

DEFINE/KEY

Associates an equivalence string and a set of attributes with a key on the terminal keyboard. The **/KEY** qualifier is required.

format

DEFINE/KEY *key-name equivalence-string*

parameters

key-name

Specifies the name of the key that you are defining. The following table lists the key names in column one. The remaining three columns indicate the key designations on the keyboards of the three different types of terminals that allow key definitions.

Key-Name	LK201	VT100-Series	VT52
PF1	PF1	PF1	[blue]
PF2	PF2	PF2	[red]
PF3	PF3	PF3	[gray]

Key-Name	LK201	VT100-Series	VT52
PF4	PF4	PF4	--
KP0, KP1, ..., KP9	0, 1, ..., 9	0, 1, ..., 9	0, 1, ..., 9
PERIOD	.	.	.
COMMA	,	,	n/a
MINUS	-	-	n/a
ENTER	Enter	ENTER	ENTER
LEFT	←	←	←
RIGHT	→	→	→
Find (E1)	Find	--	--
Insert Here (E2)	Insert Here	--	--
Remove (E3)	Remove	--	--
Select (E4)	Select	--	--
Prev Screen (E5)	Prev Screen	--	--
Next Screen (E6)	Next Screen	--	--
HELP	Help	--	--
DO	Do	--	--
F6, F7, ..., F20	F6, F7, ..., F20	--	--

On LK201 keyboards, you cannot define the UP and DOWN arrow keys or function keys F1 through F5. The LEFT and RIGHT arrow keys and the F6 through F14 keys are reserved for command line editing. You must enter the SET TERMINAL/NOLINE_EDITING command before defining these keys. You can also press CTRL/V to enable keys F7 through F14. Note that CTRL/V will not enable the F6 key.

equivalence-string

Specifies the character string to be processed when you press the key. Enclose the string in quotation marks to preserve spaces and lowercase characters.

qualifiers

/ECHO (default)

/NOECHO

Displays the equivalence string on your screen after the key has been pressed. You cannot use /NOECHO with the /NOTERMINATE qualifier.

/ERASE

/NOERASE (default)

Determines whether the current line is erased before the key translation is inserted.

DCL-52 DCL Commands
DEFINE/KEY

/IF_STATE=(state-name,...)
/NOIF_STATE

Specifies a list of one or more states, one of which must be in effect for the key definition to work. The */NOIF_STATE* has the same meaning as */IF_STATE=current_state*.

/LOCK_STATE
/NOLOCK_STATE (default)

Specifies that the state set by the */SET_STATE* qualifier remain in effect until explicitly changed. (By default, the */SET_STATE* qualifier is in effect only for the next definable key you press or the next read-terminating character that you type.) Can only be specified with the */SET_STATE* qualifier.

/LOG (default)
/NOLOG

Displays a message indicating that the key definition has been successfully created.

/SET_STATE=state-name
/NOSET_STATE (default)

Causes the specified state-name to be set when the key is pressed. (By default, the current locked state is reset when the key is pressed.) The state name can be any alphanumeric string; specify the state as a character string enclosed in quotation marks (").

/TERMINATE
/NOTERMINATE (default)

Specifies whether the current equivalence string is to be processed immediately when the key is pressed (equivalent to entering the string and pressing RETURN). By default, you can press other keys before the definition is processed.

example

```
$ DEFINE/KEY PF1 "SHOW " /SET_STATE=GOLD/NOTERMINATE/ECHO
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
$ DEFINE/KEY PF1 " DEFAULT" /TERMINATE/IF_STATE=GOLD/ECHO
%DCL-I-DEFKEY, GOLD key PF1 has been defined
$ SHOW DEFAULT
DISK1: [JOHN.TEST]
```

In this example, the first *DEFINE/KEY* command defines the PF1 key to be the string *SHOW*. The state is set to *GOLD* for the subsequent key. The */NOTERMINATE* qualifier instructs the system not to process the string when the key is pressed. The second *DEFINE/KEY* command defines the use of the PF1 key when the keypad is in the *GOLD* state. When the keypad is in the *GOLD* state, pressing PF1 causes the current read to be terminated.

If you press the PF1 key twice, the system displays and processes the *SHOW DEFAULT* command.

The word *DEFAULT* in the second line of the example indicates that the PF1 key has been defined in the default state. Note the space before the word *DEFAULT* in the second DEFINE/KEY command. If the space is omitted, the system fails to recognize DEFAULT as the keyword for the SHOW command.

DELETE

Deletes one or more files from a mass storage disk volume.

format

DELETE *file spec[,...]*

parameter

file-spec[,...]

Specifies the names of one or more files to be deleted from a mass storage disk volume. The first file specification must contain an explicit or default directory specification plus an explicit file name, file type, and version number. Subsequent file specifications need contain only a version number; the defaults will come from the preceding specification. Wildcard characters can be used in any of the file specification fields. If you omit the directory specification or device name, the current default device and directory are assumed. If the file specification contains a null version number (a semicolon followed by no file version number), a version number of 0, or one or more spaces in the version number, the latest version of the file is deleted. To delete more than one file, separate the file specifications with commas or plus signs.

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

DCL-54 DCL Commands
DELETE

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

/NOCONFIRM (default)

Controls whether a request is issued before each DELETE operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL

RET

You can use any combination of upper- and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/ERASE

/NOERASE (default)

When you delete a file, the area in which the file was stored is returned to the system for future use. The data that was stored in that location still exists in the system until new data is written over it. When you specify the /ERASE qualifier, the storage location is overwritten with a system specified pattern so that the data no longer exists.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the DELETE operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */EXPIRED* selects files according to their expiration dates. (The expiration date is set with the *SET FILE/EXPIRATION_DATE* command.) The */EXPIRED* qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

/LOG

/NOLOG (default)

Controls whether the *DELETE* command displays the file specification of each file after its deletion.

/MODIFIED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: *TODAY* (default), *TOMORROW*, or *YESTERDAY*. Specify one of the following qualifiers with */BEFORE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

example

```
$ DIRECTORY [.SUBTEST]
%DIRECT-W-NOFILES, no files found
$ SET PROTECTION SUBTEST.DIR/PROTECTION=OWNER:D
$ DELETE SUBTEST.DIR;1
```

Before the directory file *SUBTEST.DIR* is deleted, the *DIRECTORY* command is used to verify that there are no files cataloged in the directory. The *SET PROTECTION* command redefines the protection for the directory file so that it can be deleted; then the *DELETE* command deletes it.

DELETE/CHARACTERISTIC

Deletes the definition of a queue characteristic.

Requires OPER privilege.

format

DELETE/CHARACTERISTIC *characteristic-name*

parameter

characteristic-name

Specifies the name of the characteristic to be deleted.

qualifier

/LOG

/NOLOG (default)

Controls whether the DELETE/CHARACTERISTIC command displays the name of each characteristic after its deletion.

example

```
$ DEFINE/CHARACTERISTIC BLUE 7
```

```
.  
. .  
. . .
```

```
$ DELETE/CHARACTERISTIC BLUE
```

```
$ DEFINE/CHARACTERISTIC BLUE_INK 7
```

The DEFINE/CHARACTERISTIC command in this example establishes the characteristic BLUE, with number 7, to mean blue ink ribbons for printers. To change the name of the characteristic, enter the DELETE/CHARACTERISTIC command. Then enter another DEFINE/CHARACTERISTIC command to rename the characteristic to BLUE_INK, using the characteristic number 7.

DELETE/ENTRY

Deletes one or more print or batch jobs. The jobs can be in progress or waiting in the queue.

Requires OPER privilege, EXECUTE access to the queue, or DELETE access to the specified jobs.

format

DELETE/ENTRY=(*entry-number*[,...]) [*queue-name*[:]]

parameters

***entry-number*[,...]**

Specifies the entry number (or a list of entry numbers) of jobs to be deleted.

[queue-name[:]]

Specifies the name of the queue where the jobs are located.

qualifier

/LOG

/NOLOG (default)

Controls whether the DELETE/ENTRY command displays the entry-number of each batch or print job that it deletes.

example

```
$ PRINT/HOLD ALPHA.TXT
Job ALPHA (queue SYS$PRINT, entry 110) holding
```

.
.
.

```
$ DELETE/ENTRY=110 SYS$PRINT
```

The PRINT command in this example queues a copy of the file ALPHA.TXT in a HOLD status, to defer its printing until a SET ENTRY/RELEASE command is entered. The system displays the job name, the entry-number, the name of the queue in which the job was entered, and the status. Later, the DELETE/ENTRY command requests that the entry be deleted from the queue SYS\$PRINT.

DELETE/FORM

Deletes a form (for printer or terminal queues) previously established with the DEFINE/FORM command.

Requires OPER privilege.

format

DELETE/FORM *form-name*

parameter

form-name

Specifies the name of the form to be deleted.

qualifier

/LOG

/NOLOG (default)

Controls whether the DELETE/FORM command displays the name of each form after its deletion.

example

```
$ DELETE/FORM CENTER
```

The DELETE/FORM command in this example deletes the form named CENTER.

DELETE/INTRUSION_RECORD

Removes an entry from the break-in database.

Requires CMKRNL and SECURITY privileges.

format

DELETE/INTRUSION_RECORD *source*

parameter

source

Source field of the entry to be removed from the break-in database.

example

```
$ DELETE/INTRUSION_RECORD TTC2:
```

In this example, the DELETE/INTRUSION_RECORD command removes all intrusion records generated by break-in attempts on TTC2. No username is specified because none of the login failures occurred for valid users.

DELETE/KEY

Deletes key definitions that have been established by the DEFINE/KEY command.

format

DELETE/KEY [*key-name*]

parameter

key-name

Specifies the name of the key to be deleted. Incompatible with the /ALL qualifier.

qualifiers

/ALL

Deletes all key definitions in the specified state; the default is the current state. If you use the /ALL qualifier, do not specify a key name.

/LOG (default)

/NOLOG

Controls whether messages are displayed indicating that the specified key definitions have been deleted.

/STATE=(state-name[,...])

/NOSTATE (default)

Specifies the name of the state for which the specified key definition is to be deleted. The default state is the current state.

example

```
$ DEFINE/KEY PF3 "SHOW TIME" /TERMINATE
%DCL-I-DEFKEY, DEFAULT key PF3 has been defined
$ PF3
$ SHOW TIME
  19-APR-1990 14:43:59
.
.
.

$ DELETE/KEY PF3
%/DCL-I-DELKEY, DEFAULT key PF3 has been deleted
$ PF3
$
```

DCL-60 DCL Commands

DELETE/KEY

In this example, the **DEFINE/KEY** command defines the PF3 key on the keypad as **SHOW TIME**. To undefine the PF3 key, use the **DELETE/KEY** command. When the user presses PF3, only the system prompt is displayed.

DELETE/QUEUE

Deletes a print or batch queue created by the **INITIALIZE/QUEUE** command. Also deletes all the jobs in the queue. The specified queue must be stopped first. The **/QUEUE** qualifier is required.

Requires OPER privilege.

format

```
DELETE/QUEUE queue-name[:]
```

parameter

queue-name[:]

Specifies the name of the queue to be deleted.

qualifier

/LOG

/NOLOG (default)

Controls whether the **DELETE/QUEUE** command displays the name of each queue after it is deleted.

example

```
$ INITIALIZE/QUEUE/DEFAULT=FLAG/START/ON=LPAO LPAO_QUEUE
```

```
.  
. .  
. . .
```

```
$ STOP/QUEUE/NEXT LPAO_QUEUE
```

```
$ DELETE/QUEUE LPAO_QUEUE
```

In this example, the first command initializes and starts the printer queue **LPAO_QUEUE**. The **STOP/QUEUE/NEXT** command stops the queue. The **DELETE/QUEUE** command deletes the queue.

DELETE/SYMBOL

Deletes one or all symbol definitions from a local or global symbol table. The /SYMBOL qualifier is required.

format

DELETE/SYMBOL [*symbol-name*]

parameter

symbol-name

Specifies the name of the symbol to be deleted. A name is required unless the /ALL qualifier is specified. The symbol-name parameter is incompatible with the /ALL qualifier.

qualifiers

/ALL

Deletes all symbols from the specified table. The /ALL qualifier is incompatible with the symbol-name parameter.

/GLOBAL

Deletes the symbol from the global symbol table of the current process.

/LOCAL (default)

Deletes the symbol from the local symbol table of the current process.

/LOG

/NOLOG (default)

Controls whether an informational message listing each symbol being deleted is displayed.

example

```
$ DELETE/SYMBOL/LOG FOO
%DCL-I-DELSYM, LOCAL symbol FOO has been deleted
```

In this example, the DELETE/SYMBOL command deletes the symbol FOO from the local symbol table for the current process. In addition, the /LOG qualifier causes an informational message, listing the symbol being deleted, to be displayed.

DEPOSIT

Replaces the contents of the specified locations in virtual memory and displays the new contents. If the specified address can be read but not written by the current access mode, the original contents are displayed; if the specified address can be neither read nor written, asterisks are displayed in the data field. The DEPOSIT command maintains a pointer at that location (at the byte following the last byte modified).

Requires user mode read (R) and write (W) access to the virtual memory location whose contents you wish to change.

format

DEPOSIT *location=*data[...]

parameters

location

Specifies the starting virtual address or range of virtual addresses (where the second address is larger than the first) whose contents are to be changed. A location can be any valid integer expression containing an integer value, a symbol name, a lexical function, or a combination of these entities. Radix qualifiers determine the radix in which the address is interpreted; hexadecimal is the initial default radix. Symbol names are always interpreted in the radix in which they were defined. The radix operators %X, %D, or %O can precede the location. A hexadecimal value must begin with a number (or be preceded by %X). The specified location must be within the virtual address space of the image currently running in the process.

data[...]

Specifies the data to be deposited into the specified locations. By default, the data is assumed to be in hexadecimal format; it is then converted to binary format and is written into the specified location.

qualifiers

/ASCII

Indicates that the specified data is ASCII. Only one data item is allowed; all characters to the right of the equal sign are considered to be part of a single string. Unless they are enclosed within quotation marks, characters are converted to uppercase and multiple spaces are compressed to a single space before the data is written in memory. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory. When you specify /ASCII, or when ASCII mode is the default, the location you specify is assumed to be hexadecimal.

/BYTE

Requests that data be deposited one byte at a time.

/DECIMAL

Indicates that the data is decimal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/HEXADECIMAL

Indicates that the data is hexadecimal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/LONGWORD

Requests that data be deposited a longword at a time.

/OCTAL

Indicates that the data is octal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/WORD

Requests that the data be deposited one word at a time.

example

```
$ DEPOSIT/ASCII 2C00=FILE: NAME: TYPE:  
00002C00: FILE: NAME: TYPE:...
```

In this example, the DEPOSIT command deposits character data at hexadecimal location 2C00 and displays the contents of the location after modifying it. Because the current default length is a longword, the response from the DEPOSIT command displays full longwords. Trailing dots (ellipses) indicate that the remainder of the last longword of data contains information that was not modified by the DEPOSIT command.

DIFFERENCES

Compares the contents of two disk files and displays a listing of the records that do not match.

format

DIFFERENCES *input1-file-spec* [*input2-file-spec*]

parameters

input1-file-spec

Specifies the first file to be compared. The file specification must include a file name and a file type. Wildcard characters are not allowed.

input2-file-spec

Specifies the second file to be compared. Unspecified fields default to the corresponding fields in *input1-file-spec*. Wildcard characters are not allowed. If you do not specify a secondary input file, the DIFFERENCES command uses the next lower version of the primary input file.

qualifiers

/CHANGE_BAR=[(change-char],[NO]NUMBER)]

Marks with the specified character in the left margin each line in the input1 file that differs from the corresponding line in the input2 file. If you do not specify a change bar character, the default is an exclamation point (!) for ASCII output. If you specify hexadecimal or octal output (see /MODE qualifier), the change bar character is ignored and differences are marked by a "****CHANGE****" string in the record header. The keyword NONUMBER suppresses line numbers in the listing. If neither the NUMBER nor NONUMBER keyword is specified, the default is controlled by the /[NO]NUMBER command qualifier. If only one option is specified, the parentheses can be omitted.

/COMMENT_DELIMITER=(character[,...])

Ignores lines starting with a specified comment character. If the comment character is an exclamation point or semicolon, it can appear anywhere in the line and characters to the right of the character are ignored. If you specify just one character, you can omit the parentheses. Lowercase characters are automatically converted to uppercase unless they are enclosed in quotation marks. Non-alphanumeric characters (such as ! and ,) must be enclosed in quotation marks. You can specify up to 32 comment characters by typing the character itself or one of the following keywords. (Keywords can be abbreviated provided that the resultant keyword is not ambiguous and has at least two characters; single letters are treated as delimiters.)

Keyword	Character
COLON	Colon (:)
COMMA	Comma (,)
EXCLAMATION	Exclamation point (!)
FORM_FEED	Form feed
LEFT	Left bracket ([)
RIGHT	Right bracket (])
SEMI_COLON	Semicolon (;)
SLASH	Slash (/)
SPACE	Space
TAB	Tab

The following characters are the default comment delimiters for files with the specified file types:

File Type	Default Comment Character
B2S, B32, BAS, BLI	!
CBL, CMD	! and ;
COB	* or / in the first column
COM, COR	!
FOR	! anywhere and C, D, c, d in the first column
HLP	!
MAC, MAR	;
R32, REQ	!

/IGNORE=(keyword[,...])

Inhibits the comparison of the specified characters, strings, or records; also controls whether the comparison records are output to the listing file as edited records or exactly as they appeared in the input file. If you specify only one keyword, you can omit the parentheses. The keyword parameter refers either to a character or a keyword. The first set of keywords determines what, if anything, is ignored during file comparison; the second set of keywords determines whether or not ignored characters are included in the output. The following keywords are valid options for the **/IGNORE** qualifier:

BLANK_LINES	Blank lines between data lines.
COMMENTS	Data following a comment character.
FORM_FEEDS	Form feed character.
HEADER[=n]	First <i>n</i> records of the file, beginning with a record whose first character is a form feed. The first record is not ignored if the only character it contains is a form feed. (<i>N</i> indicates the number of records and defaults to 2. A record with a single form feed is not counted.)
TRAILING_SPACES	Space and tab characters at the end of a data line.
SPACING	Extra blank spaces or tabs within data lines.
EDITED	Omits ignored characters from the output records.
EXACT	Includes ignored characters in the output records.
PRETTY	Formats output records.

DCL-66 DCL Commands

DIFFERENCES

If you specify `/PARALLEL`, output records are always formatted. To format output records, specify the following characters:

Character	Formatted Output
Tab (CTRL/I)	1-8 spaces
RETURN (CTRL/M)	<CR>
Line feed (CTRL/J)	<LF>
Vertical tab (CTRL/K)	<VT>
Form feed (CTRL/L)	<FF>
Other nonprinting characters	. (period)

/MATCH=size

Specifies the number of records that should indicate matching data after a difference is found. By default, after `DIFFERENCES` finds unmatched records, it assumes that the files once again match after it finds three sequential records that match. Use the `/MATCH` qualifier to override the default match size of 3.

/MAXIMUM_DIFFERENCES=n

Terminates `DIFFERENCES` after a specified number of unmatched records (specified with the `n` parameter) is found.

/MERGED[=n]

Specifies that the output file contain a merged list of differences with the specified number of matched records listed after each group of unmatched records. The specified number (the value `n`) must be less than or equal to the number specified in the `/MATCH` qualifier. By default, `DIFFERENCES` produces a merged listing with one matched record listed after each set of unmatched records (that is, `/MERGED=1`). If neither `/MERGED` nor `/SEPARATED` nor `/PARALLEL` is specified, the resulting output is merged, with one matched record following each unmatched record.

/MODE=(radix[,...])

Specifies the format of the output. You can request that the output be formatted in one or more radix modes by specifying the following keywords, which may be abbreviated: ASCII (default), HEXADECIMAL, or OCTAL. If you specify only one radix, you can omit the parentheses. If you specify `/PARALLEL` or `/SLP`, `/MODE` is ignored for that listing form.

/NUMBER (default)

/NONUMBER

Includes line numbers in the listing of differences.

/OUTPUT[=file-spec]

Specifies an output file to receive the list of differences. By default, the output is written to the current SYS\$OUTPUT device. If the file-spec parameter is not specified, the output is directed to the first input file with a file type of DIF. No wildcard characters are allowed.

/PARALLEL[=n]

Lists the records with differences side by side. The value n specifies the number of matched records to merge after each unmatched record; the value n must be a non-negative decimal number less than or equal to the number specified in /MATCH.

/SEPARATED[=(input1-file-spec,input2-file-spec)]

Lists sequentially only the records from the specified file that contain differences. If no files are specified, a separate listing is generated for each file. If only one file is specified, you can omit the parentheses. To specify the input1-file-spec parameter, use either the first input file specified as the DIFFERENCES parameter or the keyword MASTER. To specify the input2-file-spec parameter, use either the second input file specified as the DIFFERENCES parameter or the keyword REVISION. By default, DIFFERENCES creates only a merged list of differences.

/SLP

Requests that DIFFERENCES produce an output file suitable for input to the SLP editor. If you use the /SLP qualifier, you cannot specify any of the following output file qualifiers: /MERGED, /PARALLEL, /SEPARATED, or /CHANGE_BAR.

Use the output file produced by the SLP qualifier as input to SLP to update the master input file, that is, to make the master input file match the revision input file.

When you specify /SLP and you do not specify /OUTPUT, DIFFERENCES writes the output file to a file with the same file name as the master input file with the file type DIF.

/WIDTH=n

Specifies the width of the lines in the output file. The default is 132 characters. If output is written to the terminal, /WIDTH is ignored and the terminal line width is used.

/WINDOW=size

Searches the number of records specified (the value n) before a record is declared as unmatched. By default, DIFFERENCES searches to the ends of both input files before listing a record as unmatched.

DCL-68 DCL Commands

DIFFERENCES

example

```
$ DIFFERENCES EXAMPLE.TXT
*****
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
  1 DEMONSTRATION
  2 OF V3.0 DIFFERENCES
  3 UTILITY
*****
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
  1 DEMONSTRATION
  2 OF VMS DIFFERENCES
  3 UTILITY
*****
Number of difference sections found: 1
Number of difference records found: 2
DIFFERENCES/MERGED=1-
DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
```

In this example, the DIFFERENCES command compares the contents of the two most recent versions of the file EXAMPLE.TXT in the current default directory. DIFFERENCES compares every character in every record and displays the results at the terminal.

DIRECTORY

Provides a list of files or information about a file or group of files.

Requires READ (R) access to the directories or sufficient privilege to override the protection to obtain information. Requires READ access to the files or sufficient privilege to override the protection to obtain information other than the file name.

format

DIRECTORY [*file-spec*[,...]]

parameter

***file-spec*[,...]**

Specifies one or more files to be listed. The syntax of a file specification determines which files will be listed, as follows:

- If you do not enter a file specification, the DIRECTORY command lists all versions of the files in the current default directory.
- If you specify only a device name, the DIRECTORY command uses your default directory specification.
- Whenever the file specification does not include a file name, file type and a version number, all versions of all files in the specified directory are listed.

- If a file specification contains a file name or a file type, or both, and no version number, the DIRECTORY command lists all versions.
- If a file specification contains only a file name, the DIRECTORY command lists all files in the current default directory with that file type, regardless of file type and version number.
- If a file specification contains only a file type, the DIRECTORY command lists all files in the current default directory with that file type, regardless of file name and version number.

Wildcard characters can be used. Separate multiple file specifications with either commas or plus signs.

qualifiers

/ACL

Controls whether the access control list (ACL) is displayed for each file. The /ACL qualifier overrides the /COLUMNS qualifier.

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BRIEF (default)

Displays only a file's name, type, and version number. You can use the /ACL, /DATE, /FILE_ID, /NOHEADING, /OWNER, /PROTECTION, /SECURITY, and /SIZE qualifiers to expand a brief display.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/COLUMNS=n

Specifies the number of columns in a brief display. The default is four. However, you can request as many columns as you like, restricted by the value of the /WIDTH qualifier. The /COLUMNS qualifier is incompatible with /ACL, /FULL, and /SECURITY.

DCL-70 DCL Commands

DIRECTORY

/CREATED (default)

Modifies the time value specified with the **/BEFORE** or **/SINCE** qualifier. **/CREATED** selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: **/BACKUP**, **/EXPIRED**, and **/MODIFIED**. If you specify none of these four time qualifiers, the default is **/CREATED**.

/DATE[=option]

/NODATE (default)

Includes the backup, creation, expiration, or modification date for each specified file; the default is **/NODATE**. If you use the **/DATE** qualifier without an option, the creation date is provided. Possible options are as follows:

ALL	Creation, expiration, backup, and last modification dates
BACKUP	Last backup date
CREATED	Creation date
EXPIRED	Expiration date
MODIFIED	Last modification date

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the **DIRECTORY** operation. When using **/EXCLUDE** in a **DIRECTORY** operation of a different device, use only the file name in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the **/BEFORE** or **/SINCE** qualifier. **/EXPIRED** selects files according to their expiration dates. (The expiration date is set with the **SET FILE/EXPIRATION_DATE** command.) The **/EXPIRED** qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: **/BACKUP**, **/CREATED**, and **/MODIFIED**. If you specify none of these four time qualifiers, the default is **/CREATED**.

/FILE_ID

Controls whether the file's identification number (FID) is displayed. By default, a file's identification is not displayed unless the **/FULL** qualifier is specified.

/FULL

Displays the following information for each file:

- File name
- File type
- Version number
- Number of blocks used

Number of blocks allocated
Date of creation
Date last modified and revision number
Date of expiration
Date of last backup
File owner's UIC
File protection
File identification number (FID)
File organization
Journaling information
Other file attributes
Record attributes
Record format
Access control list (ACL)
Value of the stored semantics tag (where applicable)

/GRAND_TOTAL

Displays only the totals for all files and directories that have been specified.

/HEADING

/NOHEADING

Controls whether heading lines consisting of a device description and directory specification are printed. The default output format provides this heading. When */NOHEADING* is specified, the display is in single-column format and the device and directory information appears with each file name. The */NOHEADING* qualifier overrides */COLUMNS*.

/MODIFIED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/OUTPUT[=file-spec]

/NOOUTPUT

Controls where the output of the command is sent. By default, the display is written to the current *SYS\$OUTPUT* device. No wildcard characters are allowed.

/OWNER

/NOOWNER (default)

Controls whether the file owner's UIC is listed.

DCL-72 DCL Commands

DIRECTORY

/PRINTER

Puts the display in a file and queues the file to SYS\$PRINT for printing under the name given by the /OUTPUT qualifier. If you do not specify the /OUTPUT qualifier, output is directed to a temporary file named DIRECTORY.LIS, which is queued for printing and then deleted.

/PROTECTION

/NOPROTECTION (default)

Controls whether the file protection for each file is listed.

/SECURITY

Controls whether information about file security is displayed; using /SECURITY is equivalent to using the /ACL, /OWNER, and /PROTECTION qualifiers together.

/SELECT=(keyword[,...])

Allows you to select files for display according to size. Choose one of the following keywords:

SIZE=MAXIMUM=n

Displays files that have fewer blocks than the value of n, which defaults to 1,073,741,823. Use with MINIMUM=n to specify a size range for files to be displayed.

SIZE=MINIMUM=n

Displays files that have blocks equal to or greater than the value of n, which defaults to 0. Use with MAXIMUM=n to specify a size range for files to be displayed.

SIZE=(MAXIMUM=n,MINIMUM=m)

Displays files whose blocksize falls within the specified MAXIMUM and MINIMUM range.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/SIZE[=option]

/NOSIZE (default)

Displays the size in blocks of each file. If you omit the option parameter, the default lists the file size in blocks used (USED). Specify one of the following options:

ALL Lists the file size both in blocks allocated and blocks used

ALLOCATION Lists the file size in blocks allocated

USED Lists the file size in blocks used

/TOTAL

Displays only the directory name and total number of files.

/TRAILING
/NOTRAILING

Controls whether trailing lines that provide the following summary information are displayed:

- Number of files listed
- Total number of blocks used per directory
- Total number of blocks allocated
- Total number of directories and total blocks used or allocated in all directories (only if more than one directory is listed)

By default, the output format includes most of this summary information. The */SIZE* and */FULL* qualifiers determine more precisely what summary information is included. Used by itself, */TRAILING* lists the number of files in the directory. Used with */SIZE*, */TRAILING* lists the number of files and the number of blocks (displayed according to the option of the */SIZE* qualifier, *FULL* or *ALLOCATION*). Used with */FULL*, */TRAILING* lists the number of files as well as the number of blocks used and allocated. If more than one directory is listed, the summary includes the total number of directories, the total number of blocks used, and the total number of blocks allocated.

/VERSIONS=n

Specifies the number of versions of a file to be listed. The default is all versions of each file. A value less than 1 is not allowed.

/WIDTH=(keyword[,...])

Formats the width of the display. Possible keywords are as follows:

DISPLAY=n	Specifies the total width of the display as an integer in the range 1 through 256 and defaults to 0 (setting the display width to the terminal width).
FILENAME=n	Specifies the width of the file name field; defaults to 19 characters.
OWNER=n	Specifies the width of the owner field; defaults to 20.
SIZE=n	Specifies the width of the size field; defaults to 6.

DCL-74 DCL Commands

DIRECTORY

example

```
$ DIRECTORY/FULL [JONES.ITALIA]PROJECTIONS.LIS
```

```
Directory WORK:[JONES.ITALIA]
```

```
PROJECTIONS.LIS;1          File ID: (7449,36222,2)
Size:          21/21        Owner:      [DOC,JONES]
Created:       5-MAY-1988 15:49:03.11
Revised:       5-MAY-1988 15:49:49.39 (2)
Expires:       <None specified>
Backup:        <No backup recorded>
File organization: Sequential
File attributes: Allocation: 21, Extend: 0, Global buffer count: 0,
                  No version limit
Record format:  Variable length, maximum 80 bytes
Record attributes: Carriage return carriage control
RMS attributes: None
Journaling enabled: None
File protection: System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List: None
```

```
Total of 1 file, 21/21 blocks.
```

The **DIRECTORY** command in this example shows the date/time format using the default language, English, and the default VMS format. You can also select other languages and formats that have been defined on your systems with international date/time formatting routines available in the run-time library.

DISCONNECT

Breaks the connection between a physical terminal and a virtual terminal. After the physical terminal is disconnected, both the virtual terminal and the process using it remain on the system.

Requires that your physical terminal is connected to a virtual terminal.

format

```
DISCONNECT
```

parameters

None.

qualifier

/CONTINUE

/NOCONTINUE (default)

Controls whether the CONTINUE command is executed in the current process just before connecting to another process. This permits an interrupted image to continue processing after the disconnection until the process needs terminal input or attempts to write to the terminal. At that point, the process waits until the physical terminal is reconnected to the virtual terminal.

example

```
$ RUN PAYROLL
```

```
CTRL/Y
```

```
$ DISCONNECT/CONTINUE
```

In this example, the RUN command is entered from a physical terminal that is connected to a virtual terminal. After the image PAYROLL.EXE is interrupted, the DISCONNECT command disconnects the physical and the virtual terminals without logging out the process. The /CONTINUE qualifier allows the image PAYROLL.EXE to continue to execute until the process needs terminal input or attempts to write to the terminal. At that point, the process waits until the physical terminal is reconnected to the virtual terminal. However, you can use the physical terminal to log in again and perform other work.

DISMOUNT

Closes a mounted disk or magnetic tape volume for further processing and cancels the logical name associated with the device. If the volume is mounted with the /SHARE qualifier, its logical name is canceled but the volume remains mounted until all processes using it dismount it or terminate.

DISMOUNT checks for open files and other conditions that prevent a Files-11 volume from dismounting. If such a condition exists, DISMOUNT displays messages indicating that the volume cannot be dismounted and lists the conditions that prevent the dismount. By default, if such a condition exists, DISMOUNT does *not* mark the volume for dismount.

Requires the GRPNAM and SYSNAM user privileges to dismount group and system volumes.

format

DISMOUNT *device-name[:]*

parameter

device-name[:]

Name of the device containing the volume—either a logical name or a physical name. If a physical name is specified, the controller defaults to A and the unit defaults to 0.

qualifiers

/ABORT

Requires volume ownership or the user privilege VOLPRO to use this qualifier with a volume that is mounted neither group nor system. Specifies that the volume is to be dismounted, regardless of who mounted it. The primary purpose of the /ABORT qualifier is to terminate mount verification. DISMOUNT/ABORT also cancels any outstanding I/O requests. If the volume was mounted with the /SHARE qualifier, the /ABORT qualifier causes the volume to be dismounted for all of the users who mounted it.

/CLUSTER

Dismounts a volume clusterwide. If you specify DISMOUNT/CLUSTER, DISMOUNT checks for open files or other conditions that will prevent a Files-11 volume on the local node from dismounting. If DISMOUNT does not find any open files or other conditions, it checks for conditions on all other nodes in the cluster. If DISMOUNT finds one of the conditions on any node, it displays an error message identifying the device and the nodes on which the error occurred, followed by an error message indicating open files or other conditions on the volume.

After DISMOUNT successfully dismounts the volume on the local node, it dismounts the volume on every other node in the existing VAXcluster environment. If the system is not a member of a cluster, the /CLUSTER qualifier has no effect.

/OVERRIDE=CHECKS

Marks a Files-11 volume for dismounting even if files are open on the volume. If you specify DISMOUNT/OVERRIDE=CHECKS, DISMOUNT displays messages indicating any open files or other conditions that prevent dismounting, immediately followed by a message indicating that the volume has been marked for dismounting.

A substantial amount of time can pass between the time you enter DISMOUNT/OVERRIDE=CHECKS and the completion of the dismount. Always wait for the dismount to complete before you remove the volume. (To verify that the dismount has completed, enter the SHOW DEVICES command.) Note that the final phase of volume dismounting occurs in

the file system, and all open files on the volume must be closed before the actual dismount can be done. Note, also, that the file system cannot dismount a volume while any known file lists associated with it contain entries.

/UNIT

Dismounts only one volume of a volume set on the specified device. By default, all volumes in a set are dismounted.

NOTE: Avoid dismounting the root volume of a volume set, because it contains the master file directory. It may be impossible to access files on a volume set if the MFD is not accessible.

/UNLOAD
/NOUNLOAD

Unloads the device on which the volume is mounted. If you specify DISMOUNT/UNLOAD, the volume is unloaded physically. If you specify DISMOUNT/NOUNLOAD, the volume is not unloaded physically. If you specify DISMOUNT without the /UNLOAD or the /NOUNLOAD qualifier, the qualifier that you specified with the MOUNT command (either /UNLOAD or /NOUNLOAD) determines whether or not the volume is unloaded physically.

example

```
$ MOUNT MT: PAYVOL TAPE
```

```
.  
. .  
. . .
```

```
$ DISMOUNT TAPE
```

The MOUNT command in this example mounts the tape whose volume identification is PAYVOL on the device MTA0: and assigns the logical name TAPE to the device. By default, the volume is not shareable. The DISMOUNT command releases access to the volume, deallocates the device, and deletes the logical name TAPE.

```
$ DISMOUNT $10$DJA100  
%DISM-W-CANNOTDMT, $10$DJA100: cannot be dismounted  
%DISM-W-INSWPGFIL, 4 swap or page files installed on volume  
%DISM-W-SPOOLEDEV, 3 devices spooled to volume  
%DISM-W-INSTIMAGE, 7 images installed on volume  
%DISM-W-USERFILES, 6 user files open on volume
```

The DISMOUNT command in this example displays the open files and other conditions that prevent device \$10\$DJA100 from dismounting.

DCL-78 DCL Commands

DISMOUNT

```
$ DISMOUNT/CLUSTER $10$DJA100
%DISM-W-RMTDMTFAIL, $10$DJA100: failed to dismount on node SALT
%DISM-W-FILESOPEN, volume has files open on remote node
%DISM-W-RMTDMTFAIL, $10$DJA100: failed to dismount on node PEPPER
%DISM-W-FILESOPEN, volume has files open on remote node
%DISM-W-CANNOTDMT, $10$DJA100: cannot be dismounted
```

The DISMOUNT command in this example displays messages identifying device \$10\$DJA100 and nodes SALT and PEPPER on which errors occurred followed by messages indicating open files on the volume.

DUMP

Displays the contents of a file, disk volume, or magnetic tape volume in decimal, hexadecimal, or octal format, as well as the ASCII conversion.

format

DUMP *file-spec* [...]

parameter

file-spec

Specifies the file or name of the device being dumped.

qualifiers

/ALLOCATED

Includes in the dump all blocks allocated to the file. (By default, the dump does not include blocks following the end-of-file.) /ALLOCATED and /RECORDS are mutually exclusive.

/BLOCKS[=(option[,...])]

Dumps the specified blocks one block at a time, which is the default method for all devices except network devices. Block numbers are specified as integers relative to the beginning of the file. Typically, blocks are numbered beginning with 1. If a disk device is mounted /FOREIGN, blocks are numbered beginning with 0. Select a range of blocks to be dumped by specifying one of the following options:

START:n	Specifies the number of the first block to be dumped; the default is the first block.
END:n	Specifies the number of the last block to be dumped; the default is the last block or the end-of-file block, depending on the /ALLOCATED qualifier.
COUNT:n	Specifies the number of files to be dumped. COUNT provides an alternative to END; you may not specify both.

If you specify only one option, you can omit the parentheses. /BLOCKS and /RECORDS are mutually exclusive.

DUMP***/BYTE***

Formats the dump in bytes. */BYTE*, */LONGWORD*, and */WORD* are mutually exclusive. The default format is composed of longwords.

/DECIMAL

Dumps the file in decimal radix. */DECIMAL*, */HEXADECIMAL* (default), and */OCTAL* are mutually exclusive.

/FILE_HEADER

Dumps each data block that is a valid Files-11 header in Files-11 header format rather than the selected radix and length.

/FORMATTED (default)***/NOFORMATTED***

Dumps the file header in Files-11 format; */NOFORMATTED* dumps the file header in octal format. This qualifier is useful only when */HEADER* is specified.

/HEADER

Dumps the file header and access control list. To dump only the file header, and not the file contents, also specify */BLOCK=(COUNT:0)*. */HEADER* is invalid for devices mounted */FOREIGN*.

/HEXADECIMAL (default)

Dumps the file in hexadecimal radix. */DECIMAL*, */HEXADECIMAL* (default), and */OCTAL* are mutually exclusive.

/LONGWORD (default)

Formats the dump in longwords. */BYTE*, */LONGWORD*, and */WORD* are mutually exclusive.

/NUMBER[=n]

Specifies how byte offsets are assigned to the lines of output. If you specify */NUMBER*, the byte offsets increase continuously through the dump, beginning with *n*; if you omit */NUMBER*, the first byte offset is 0. By default, the byte offset is reset to 0 at the beginning of each block or record.

/OCTAL

Dumps the file in octal radix. */DECIMAL*, */HEXADECIMAL* (default), and */OCTAL* are mutually exclusive.

/OUTPUT[=file-spec]

Specifies the output file for the dump. If you do not specify a file specification, the default is the file name of the file being dumped and the file type DMP. If */OUTPUT* is not specified, the dump goes to *SY\$OUTPUT*. No wildcard characters are allowed. */OUTPUT* and */PRINTER* are mutually exclusive.

DCL-80 DCL Commands
DUMP

/PRINTER

Queues the dump to SYS\$PRINT in a file named with the file name of the file being dumped and the file type DMP. If /PRINTER is not specified, the dump goes to SYS\$OUTPUT. No wildcard characters are allowed. /OUTPUT and /PRINTER are mutually exclusive.

/RECORDS[=(option[,...])]

Dumps the file a record at a time rather than a block at a time. (By default, input is dumped one block at a time for all devices except network devices.) Blocks are numbered beginning with 1.

Select a range of blocks to be dumped by specifying one of the following options:

- START:n Specifies the number of the first record to be dumped; the default is the first record.
- END:n Specifies the number of the last record to be dumped; the default is the last record of the file.
- COUNT:n Specifies the number of records to be dumped. COUNT provides an alternative to END; you may not specify both.

If you specify only one option, you can omit the parentheses. If you specify /RECORDS, you cannot specify /ALLOCATED or /BLOCKS.

/WORD

Formats the dump in words. /BYTE, /LONGWORD, and /WORD are mutually exclusive.

example

```
$ DUMP TEST.DAT
Dump of file DISK0:[NORMAN]TEST.DAT;1 on 19-APR-1990 15:43:26.08
File ID (3134,818,2) End of file block 1 / Allocated 3
Virtual block number 1 (00000001), 512 (0200) bytes
 706D6173 20612073 69207369 68540033 3.This is a samp 000000
 73752065 62206F74 20656C69 6620656C le file to be us 000010
 61786520 504D5544 2061206E 69206465 ed in a DUMP exa 000020
 00000000 00000000 0000002E 656C706D mple..... 000030
 00000000 00000000 00000000 00000000 ..... 000040
 00000000 00000000 00000000 00000000 ..... 000050
 00000000 00000000 00000000 00000000 ..... 000060
.
.
00000000 00000000 00000000 00000000 ..... 0001E0
00000000 00000000 00000000 00000000 ..... 0001F0
```

The DUMP command displays the contents of TEST.DAT both in hexadecimal longword format and in ASCII beginning with the first block in the file.

EDIT/ACL

Invokes the Access Control List (ACL) Editor to create or modify an access control list for a specified object. The /ACL qualifier is required.

format

EDIT/ACL *object-spec*

EDIT/EDT

Invokes the VAX EDT interactive text editor. The /EDT qualifier is not required, because EDT is the VMS default editor.

format

EDIT *file-spec*

parameter

file-spec

Specifies the file to be created or edited using the EDT editor. If the file does not exist, it is created by EDT. The EDT editor does not provide a default file type when creating files; if you do not include a file type, it is null. The file must be a disk file on a Files-11 formatted volume. No wildcard characters are allowed in the file specification.

qualifiers

/COMMAND[=*file-spec*]
/NOCOMMAND

Determines whether or not EDT uses a startup command file. The /COMMAND file qualifier should be followed by an equal sign and the specification of the command file. The default file type for command files is EDT. No wildcard characters are allowed in the file specification. If you do not include the /COMMAND=command file qualifier, EDT looks for the EDTSYS logical name assignment. If EDTSYS is not defined, EDT processes the systemwide startup command file SYS\$LIBRARY:EDTSYS.EDT. If this file does not exist, EDT looks for the EDTINI logical name assignment. If EDTINI is not defined, EDT looks for the file named EDTINI.EDT in your default directory. If none of these files exists, EDT begins your editing session in the default state. To prevent EDT from processing either the systemwide startup command file or the EDTINI.EDT file in your default directory, use the /NOCOMMAND qualifier as follows:

```
$ EDIT/NOCOMMAND MEMO.DAT
```

/CREATE (default)
/NOCREATE

Controls whether EDT creates a new file when the specified input file is not found.

/JOURNAL[=journal-file]
/NOJOURNAL

Determines whether EDT keeps a journal file during your editing session. A journal file contains a record of the keystrokes you enter during an editing session. The default file name for the journal file is the same as the input file name. The default file type is JOU. The */JOURNAL* qualifier enables you to use a different file specification for the journal file. If you are editing a file from another directory and want the journal file to be located in that directory, you must use the */JOURNAL* qualifier with a file specification that includes the directory name. Otherwise, EDT creates the journal file in the default directory. The directory that is to contain the journal file should not be write-protected. Once you have created a journal file, use EDT/RECOVER to execute the commands in the journal file. No wildcard characters are allowed in the file specification.

/OUTPUT=output-file
/NOOUTPUT

Determines whether EDT creates an output file at the end of your editing session. The default file specification for both the input file and the output file is the same. Use the */OUTPUT* qualifier to give the output file a different file specification from the input file. The */NOOUTPUT* qualifier suppresses the creation of an output file, but not the creation of a journal file. A system interruption does not prevent you from re-creating your editing session because a journal file is still being maintained. To save your editing session, even when you specify */NOOUTPUT*, use the line mode command WRITE to put the text in an external file before you end the session. No wildcard characters are allowed in the file specification.

/READ_ONLY
/NOREAD_ONLY (default)

Determines whether EDT keeps a journal file and creates an output file. With the default */NOREAD_ONLY*, EDT maintains the journal file and creates an output file when it processes the line mode command EXIT. Using the */READ_ONLY* qualifier has the same effect as specifying both the */NOJOURNAL* and */NOOUTPUT* qualifiers. Use */READ_ONLY* when you are searching a file and do not intend to make any changes to it. To modify the file, use the line mode command WRITE to save your changes. Remember, however, that you have no journal file.

/RECOVER
/NORECOVER (default)

Determines whether or not EDT reads a journal file at the start of the editing session. When you use the */RECOVER* qualifier, EDT reads the appropriate journal file and processes whatever commands it contains. If

the journal file type is not JOURNAL or the file name is not the same as the input file name, you must include both the /JOURNAL qualifier and the /RECOVER qualifier as follows:

```
$ EDIT/RECOVER/JOURNAL=SAVE.XXX MEMO.DAT
```

example

```
$ EDIT/OUTPUT=NEWFILE.TXT OLDFILE.TXT  
1        This is the first line of the file OLDFILE.TXT.  
*
```

This EDIT command invokes the EDT editor to edit the file OLDFILE.TXT. EDT looks for the EDTSYS logical name assignment. If EDTSYS is not defined, EDT processes the systemwide startup command file SYS\$LIBRARY:EDTSYS.EDT. If this file does not exist, EDT looks for the EDTINI logical name assignment. If EDTINI is not defined, EDT looks for the file named EDTINI.EDT in your default directory. If none of these files exists, EDT begins your editing session in the default state. When the session ends, the edited file has the name NEWFILE.TXT.

EDIT/FDL

Invokes the VMS FDL Editor (EDIT/FDL) to create and modify File Definition Language (FDL) files. The /FDL qualifier is required.

format

```
EDIT/FDL file-spec
```

EDIT/SUM

Invokes the SUMSLP batch-oriented editor, to update a single input file with multiple files of edit commands.

format

```
EDIT/SUM input-file
```

EDIT/TECO

Invokes the TECO interactive text editor. The /TECO qualifier is required.

format

EDIT/TECO [*file-spec*]

EDIT/TECO/EXECUTE=command-file [*argument*]

parameter

file-spec

Specifies the file to be created or edited using the TECO editor. If the file does not exist, it is created by TECO, unless you specify /NOCREATE. No wildcard characters are allowed in the file specification.

qualifiers

/COMMAND[=*file-spec*]

/NOCOMMAND

Controls whether a startup command file is used. The /COMMAND file qualifier may be followed by an equal sign and the specification of the command file. The default file type for command files is TEC. If you do not include the /COMMAND qualifier, or if you enter /COMMAND without specifying a command file, TECO looks for the TEC\$INIT logical name assignment. If TEC\$INIT is not defined, no startup commands are executed. No wildcards are allowed in the file specification.

/CREATE (default)

/NOCREATE

Creates a new file when the specified input file cannot be found. If /MEMORY is specified and no input file is specified, the file created is the one specified by the logical name TEC\$MEMORY. Normally, TECO creates a new file to match the input file specification if it cannot find the requested file name in the specified directory. When you use /NOCREATE in the TECO command line and type a specification for a file that does not exist, TECO displays an error message and returns you to the DCL command level. /EXECUTE is not compatible with /CREATE and /NOCREATE.

/EXECUTE=command-file [*argument*]

Invokes TECO and executes the TECO macro found in the command file. The argument, if specified, appears in the text buffer when macro execution starts. Blanks or special characters must be enclosed in quotes. /EXECUTE is incompatible with /CREATE and /MEMORY.

/MEMORY (default)

/NOMEMORY

Specifies that the last file you edited with TECO, identified by the logical name TEC\$MEMORY, will be the file edited if you omit the file specification to the EDIT/TECO command.

/OUTPUT=output-file

/NOOUTPUT (default)

Controls how the output file is named at the end of your editing session. By default, the output file has the same name as the input file but is given the next higher available version number. Use the /OUTPUT qualifier to give the output file a file specification different from the input file. No wildcard characters are allowed in the file specification.

/READ_ONLY

/NOREAD_ONLY (default)

Controls whether or not an output file is created. By default, an output file is created; /READ ONLY suppresses the creation of the output file.

example

```
$ EDIT/TECO/OUTPUT=NEWFILE.TXT OLDFILE.TXT
```

This EDIT command invokes the TECO editor to edit the file OLDFILE.TXT. TECO looks for the TEC\$INIT logical name assignment. If TEC\$INIT is not defined, TECO begins the editing session without using a command file. When the session ends, the edited file has the name NEWFILE.TXT.

EDIT/TPU

Invokes the VAX Text Processing Utility (VAXTPU), running the interactive text editor Extensible VAX Editor (EVE). VAXTPU is a language-like utility that is useful for creating applications that handle text. EVE is an editor that you can customize.

For information intended for general EVE users on using EDIT/TPU and its qualifiers, see Chapter 6.

format

EDIT/TPU *[file-spec]*

EOD

Signals the end of a data stream when a command or program is reading data from an input device other than an interactive terminal. Used to end a data line that begins with a dollar sign. Also, used to end an input file if more than one input file is contained in the command stream without intervening commands.

format

\$ EOD

parameters

None.

example

```
$ CREATE WEATHER.COM  
$ DECK  
$ FORTRAN WEATHER  
$ LINK WEATHER  
$ RUN WEATHER  
$ EOD  
$ @WEATHER
```

In this example, the command procedure creates a command procedure called WEATHER.COM. The lines delimited by the DECK and EOD commands are written to the file WEATHER.COM. Then the command procedure executes WEATHER.COM.

EOJ

Marks the end of a batch job submitted through a card reader. An EOJ card is not required; however, if present, the first nonblank character in the command line must be a dollar sign (\$). If issued in any other context, the EOJ command logs the process out. The EOJ command cannot be abbreviated.

The EOF card is equivalent to the EOJ card.

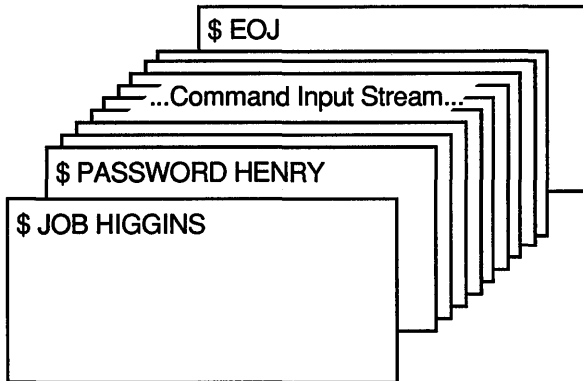
format

\$ EOJ

parameters

None.

example



ZK-0786-GE

The **JOB** and **PASSWORD** commands mark the beginning of a batch job submitted through the card reader; the **EOJ** command marks the end of the job.

ENDSUBROUTINE

Defines the end of a subroutine in a command procedure. For more information about the **ENDSUBROUTINE** command, refer to the description of the **CALL** command

format

ENDSUBROUTINE

EXAMINE

Displays the contents of virtual memory.

Requires user mode read (R) and write (W) access to the virtual memory location whose contents you want to examine.

format

EXAMINE *location[:location]*

parameter

location[:location]

Specifies a virtual address or a range of virtual addresses (where the second address is larger than the first) whose contents you want to examine. If you specify a range of addresses, separate the first and last with a colon. A location can be any valid arithmetic expression containing arithmetic or logical operators or previously assigned symbols. Radix qualifiers determine the radix in which the address is interpreted; hexadecimal is the initial default radix. Symbol names are always interpreted in the radix in which they were defined. The radix operators %X, %D, or %O can precede the location. A hexadecimal value must begin with a number (or be preceded by %X).

qualifiers

/ASCII

Requests that the data at the specified location be displayed in ASCII. Binary values that do not have ASCII equivalents are displayed as periods (.). When you specify /ASCII, or when ASCII mode is the default, hexadecimal is used as the default radix for numeric literals that are specified on the command line.

/BYTE

Requests that data at the specified location be displayed one byte at a time.

/DECIMAL

Requests that the contents of the specified location be displayed in decimal format.

/HEXADECIMAL

Requests that the contents of the specified location be displayed in hexadecimal format.

/LONGWORD

Requests that data at the specified location be displayed one longword at a time.

/OCTAL

Requests that the contents of the specified location be displayed in octal format.

/WORD

Requests that data at the specified location be displayed one word at a time.

example

```
$ RUN MYPROG
CTRL/Y
$ EXAMINE 2678
0002678: 1F4C5026
$ CONTINUE
```

In this example, the RUN command begins execution of the image MYPROG.EXE. While MYPROG is running, CTRL/Y interrupts its execution, and the EXAMINE command requests a display of the contents of virtual memory location 2678 (hexadecimal).

EXCHANGE

Invokes the Exchange Utility (EXCHANGE) to manipulate mass storage volumes that are written in formats other than those normally recognized by the VMS operating system.

EXCHANGE allows you to perform any of the following tasks:

- Create foreign volumes
- Transfer files to and from the volume
- List directories of the volume

For block-addressable devices, such as RT-11 disks, EXCHANGE performs additional operations such as renaming and deleting files. The Exchange Utility can also manipulate Files-11 files that are images of foreign volumes; these files are called **virtual devices**.

The `/[NO]MESSAGE` qualifier determines whether EXCHANGE displays information related to EXCHANGE INITIALIZE, MOUNT, and DISMOUNT subcommands. You can also use this qualifier with any of these three subcommands to reverse the default. Normally, EXCHANGE displays the information.

For more information about the Exchange Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

EXCHANGE/NETWORK

Enables the VMS operating system to transfer files to or from operating systems that do not support VMS file organizations. The transfer occurs over a DECnet network communications link that connects VMS and non-VMS operating system nodes.

Using DECnet services, the EXCHANGE/NETWORK command can:

- Transfer files between a VMS node and a non-VMS system node
- Transfer a group of input files to a group of output files
- Transfer files between two non-VMS nodes, provided those nodes share DECnet connections with the VMS node that issues the EXCHANGE/NETWORK command

format

EXCHANGE/NETWORK *input-file-spec[,...]* *output-file-spec*

parameters

input-file-spec[,...]

Specifies the name of an existing file to be transferred. Wildcard characters are allowed. Use a comma (,) to indicate multiple file specifications.

output-file-spec

Specifies the name of the output file into which the input is transferred. You must specify at least one field in the output file specification. If you omit the device or directory, your current default device and directory are used. You can use the asterisk wildcard character in place of the following: file name, file type, or version number.

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backup. This time qualifier is incompatible with the other time qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you do not specify any of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following time qualifiers with /BEFORE

to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

/NOCONFIRM (default)

Controls whether a request is issued before each file transfer operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	<input type="checkbox"/>	RET

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. The /CREATED qualifier selects files based on their date of creation. This time qualifier is incompatible with the other time qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you do not specify any of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the file transfer operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifiers. /EXPIRED selects files according to their expiration date. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) This time qualifier is incompatible with the other time qualifiers that also allow you

DCL-92 DCL Commands
EXCHANGE/NETWORK

to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you do not specify any of these four time qualifiers, the default is /CREATED.

/FDL=fdl-file-spec

Specifies that the output file characteristics are described in the File Definition Language (FDL) file. Use this qualifier when you require special output file characteristics. See the *VMS File Definition Language Facility Manual* for more information about FDL files.

Use of the /FDL qualifier implies that the transfer mode is block by block. However, the transfer mode you specify with the /TRANSFER_MODE qualifier prevails.

/LOG

/NOLOG (default)

Controls whether the EXCHANGE/NETWORK command displays the file specifications of each file copied.

When you use the /LOG qualifier, the EXCHANGE/NETWORK command displays the following for each copy operation: (1) the file specifications of the input and output files, and (2) the number of blocks or the number of records copied (depending on whether the file is copied on a block-by-block or record-by-record basis).

/MODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. The /MODIFIED qualifier selects files according to the date on which they were last modified. This time qualifier is incompatible with the other time qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you do not specify any of these four time qualifiers, the default is /CREATED.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following time qualifiers with /SINCE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/TRANSFER_MODE=option

Specifies the I/O method to be used in the transfer. This qualifier is useful for all file formats. You can specify any one of the following options:

Option	Function
AUTOMATIC	Allows EXCHANGE/NETWORK to determine the appropriate transfer mode.
BLOCK	Transfers block by block.
CONVERT[=option[,...]]	Reads records from the input file, packs them into blocks, and writes to the output file in block mode. The options determine what additional information is inserted during the transfer.
RECORD	Transfers record by record.

The AUTOMATIC transfer mode option allows EXCHANGE/NETWORK to determine the appropriate transfer mode. The default is the AUTOMATIC transfer mode.

If you explicitly select the BLOCK transfer mode option, EXCHANGE/NETWORK opens both the input and output files for block I/O. EXCHANGE/NETWORK then transfers the files block by block.

If you explicitly select the RECORD transfer mode option, EXCHANGE/NETWORK opens both the input and output files for record I/O. The target system must support record operations, and the input file must be record oriented.

If you select the CONVERT transfer mode option, EXCHANGE/NETWORK reads records in from the input file, packs them into blocks, and writes them to the output file in block mode. There are four options available with the CONVERT transfer mode to control the insertion of special characters in the records, as explained in the following paragraphs:

- CARRIAGE_CONTROL
- COUNTED
- FIXED_CONTROL
- RECORD_SEPARATOR=separator

If you specify CARRIAGE_CONTROL, any carriage control information in the input file is interpreted, expanded into actual characters, and included with each record.

If you specify COUNTED, the length of each record in bytes is included at the beginning of each record. The length includes all FIXED_CONTROL, CARRIAGE_CONTROL, and RECORD_SEPARATOR information in each record.

If you specify FIXED_CONTROL, all variable length with fixed control record (VFC) information is written to the output file as part of the data. This information follows the record length information if the COUNTED option was specified.

DCL-94 DCL Commands

EXCHANGE/NETWORK

If you specify `RECORD_SEPARATOR`, a 1- or 2-byte record separator is inserted between each record. Record separator characters are the last characters in the record. The three choices for separator characters are CR for carriage return only, LF for line feed only, or CRLF for carriage return and line feed.

example

```
$ EXCHANGE/NETWORK VMS_FILE.DAT FOO::FOREIGN_SYS.DAT
```

In this example, the `EXCHANGE/NETWORK` command transfers the file `VMS_FILE.DAT` located in the current default device and directory to the file `FOREIGN_SYS.DAT` on the non-VMS node `FOO`. Because the `/TRANSFER_MODE` qualifier was not explicitly specified, `EXCHANGE/NETWORK` automatically determines whether the transfer method should be block or record I/O.

EXIT

Terminates processing of a command procedure or subroutine and returns control to the next higher command level — either an invoking command procedure or interactive DCL. The `EXIT` command also terminates an image normally after a user enters `CTRL/Y` (executing another image has the same effect).

format

```
EXIT [status-code]
```

parameter

status-code

Defines a numeric value for the reserved global symbol `$STATUS`. You can specify the `status-code` as an integer or an expression equivalent to an integer value. The low-order three bits of the value determine the value of the global symbol `$SEVERITY`.

example

```
$ ON WARNING THEN EXIT
$ FORTRAN 'P1'
$ LINK 'P1'
$ RUN 'P1'
```

The `EXIT` command in this example is used as the target of an `ON` command; this statement ensures that the command procedure terminates whenever any warnings or errors are issued by any command in the procedure.

The procedure exits with the status value of the command or program that caused the termination.

GOSUB

Transfers control to a labeled subroutine in a command procedure without creating a new procedure level.

format

GOSUB *label*

parameter

label

Specifies a 1- through 255-alphanumeric character label appearing as the first item on a command line. A label may not contain embedded blanks. When the GOSUB command is executed, control passes to the command following the specified label. The label can precede or follow the GOSUB statement in the current command procedure. When you use a label in a command procedure, it must be terminated with a colon. If you use duplicate labels, control is always given to the label most recently read by DCL.

example

```
$!  
$! GOSUB.COM  
$!  
$ SHOW TIME  
$ GOSUB TEST1  
$ WRITE SYS$OUTPUT "success completion"  
$ EXIT  
$!  
$! TEST1 GOSUB definition  
$!  
$ TEST1:  
$   WRITE SYS$OUTPUT "This is GOSUB level 1."  
$   GOSUB TEST2  
$   RETURN %X1  
$!  
$! TEST2 GOSUB definition  
$!  
$ TEST2:  
$   WRITE SYS$OUTPUT "This is GOSUB level 2."  
$   GOSUB TEST3  
$   RETURN  
$!  
$! TEST3 GOSUB definition  
$!  
$ TEST3:  
$   WRITE SYS$OUTPUT "This is GOSUB level 3."  
$   RETURN
```

DCL-96 DCL Commands

GOSUB

This sample command procedure shows how to use the GOSUB command to transfer control to labeled subroutines. The GOSUB command transfers control to the subroutine labeled TEST1. The procedure executes the commands in subroutine TEST1, branching to the subroutine labeled TEST2. The procedure then executes the commands in subroutine TEST2, branching to the subroutine labeled TEST3. Each subroutine is terminated by the RETURN command. After TEST3 is executed, the RETURN command returns control back to the command line following each calling GOSUB statement. At this point, the procedure has been successfully executed.

GOTO

Transfers control to a labeled statement in a command procedure.

format

GOTO *label*

parameter

label

Specifies a 1- through 255-alphanumeric character label appearing as the first item on a command line. A label may not contain embedded blanks. When the GOTO command is executed, control passes to the command following the specified label. The label can precede or follow the GOTO statement in the current command procedure. When you use a label in a command procedure, it must be terminated with a colon. If you use duplicate labels, control is always given to the label most recently read by DCL.

example

```
$ IF P1 .EQS. "HELP" THEN GOTO TELL
$ IF P1 .EQS. "" THEN GOTO TELL
.
.
.
$ EXIT
$ TELL:
$ TYPE SYS$INPUT
To use this procedure, you must enter a value for P1.
.
.
.
$ EXIT
```

In this example, the IF command checks the first parameter passed to the command procedure; if this parameter is the string HELP or if the parameter is not specified, the GOTO command is executed and control is passed to the line labeled TELL. Otherwise, the procedure continues executing until the EXIT command is encountered. At the label TELL, a

TYPE command displays data in the input stream that documents how to use the procedure.

HELP

Displays information concerning use of the system, including formats and explanations of commands, parameters, and qualifiers.

format

HELP [*keyword ...*]

parameter

keyword ...

Specifies one or more keywords that refer to the topic or subtopic on which you want information from a HELP library. To use the HELP facility in its simplest form, enter the HELP command from your terminal. The HELP facility displays a list of topics at your terminal and the prompt Topic?. To see information on one of the topics, type the topic name after the prompt. The system displays information on that topic.

If the topic has subtopics, the HELP command lists the subtopics and displays the Subtopic? prompt. To get information on one of the subtopics, type the name after the prompt. To see information on another topic, press RETURN. You can now ask for information on another topic when HELP displays the Topic? prompt. Press RETURN to exit the HELP facility and return to the DCL command level.

example

```
$ HELP
HELP
.
. (HELP message text and list of topics)
.
Topic?
```

In this example, the HELP command is entered without any qualifiers or parameters. This produces a display of the HELP topics available from the root HELP library, SYS\$HELP:HELPLIB.HLB.

If you type one of the listed topics in response to the Topic? prompt, HELP displays information about that topic and a list of subtopics (if there are any). If one or more subtopics exist, HELP prompts you for a subtopic.

```
Topic? ASSIGN
ASSIGN
.
. (HELP message text and subtopics)
.
ASSIGN Subtopic?
```

DCL-98 DCL Commands

HELP

If you type a subtopic name, **HELP** displays information about that subtopic:

```
ASSIGN Subtopic? Name
ASSIGN
  Name
  .
  . (HELP message text and subtopics, if any)
  .
ASSIGN Subtopic?
```

If one or more sub-subtopics exist, **HELP** prompts for a sub-subtopic; otherwise, as in the previous example, the facility prompts you for another subtopic of the topic you are currently inspecting.

Typing a question mark redisplayes the **HELP** message and options at your current level. Pressing **RETURN** does either of the following: (1) move you back to the previous **HELP** level if you are in a subtopic level, or (2) terminate **HELP** if you are at the first level. Pressing **CTRL/Z** terminates **HELP** at any level.

IF

Tests the value of an expression and, depending on the syntax specified, executes

- One command following the **THEN** keyword if the expression is true
- Multiple commands following the **\$THEN** command if the expression is true
- One or more commands following the **\$ELSE** command if the expression is false

format

```
$ IF expression THEN [$] command
```

or

```
$ IF expression
$ THEN [command]
command
```

```
.
.
```

```
$ [ELSE] [command]
command
.
.
$ ENDIF
```

parameters

expression

Defines the test to be performed. The expression can consist of one or more numeric constants, string literals, symbolic names, or lexical functions separated by logical, arithmetic, or string operators. Expressions in IF commands are automatically evaluated during the execution of the command. Character strings beginning with alphabetic characters that are not enclosed in quotation marks are assumed to be symbol names or lexical functions. The Command Language Interpreter (CLI) replaces these strings with their current values. Symbol substitution in expressions in IF commands is not iterative; that is, each symbol is replaced only once. However, if you want iterative substitution, precede a symbol name with an apostrophe or ampersand.

command

The DCL command or commands to be executed, depending on the syntax specified, when the result of the expression is true or false.

example

```
$ IF P1 .EQS. "" THEN GOTO DEFAULT
$ IF (P1 .EQS. "A") .OR. (P1 .EQS. "B") THEN GOTO 'P1'
$ WRITE SYS$OUTPUT "Unrecognized parameter option 'P1' "
$ EXIT
$ A:      ! Process option a
.
.
.
$ EXIT
$ B:      ! Process option b
.
.
.
$ EXIT
$ DEFAULT: ! Default processing
.
.
.
$ EXIT
```

This example shows a command procedure that tests whether a parameter was passed. The GOTO command passes control to the label specified as the parameter.

DCL-100 DCL Commands

IF

If the procedure is executed with a parameter, the procedure uses that parameter to determine the label to branch to. For example:

```
@TESTCOM A
```

When the procedure executes, it determines that P1 is not null, and branches to the label A. Note that the EXIT command causes an exit from the procedure before the label B.

```
$ SET NOON
.
.
.
$ LINK CYGNUS,DRACO,SERVICE/LIBRARY
$ IF $STATUS
$ THEN
$ RUN CYGNUS
$ ELSE
$ WRITE SYS$OUTPUT "LINK FAILED"
$ ENDIF
$ EXIT
```

This command procedure uses the SET NOON command to disable error checking by the command procedure. After the LINK command, the IF command tests the value of the reserved global symbol \$STATUS. If the value of \$STATUS indicates that the LINK command succeeded, then the program CYGNUS is run. If the LINK command returns an error status value, the command procedure issues a message and exits.

INITIALIZE

Formats a disk or magnetic tape volume and writes a label on the volume. At the end of initialization, the disk is empty except for the system files containing the structure information. All former contents of the disk are lost.

Requires VOLPRO privilege for most INITIALIZE operations.

format

```
INITIALIZE device-name[:] volume-label
```

parameters

device-name[:]

Specifies the name of the device on which the volume to be initialized is physically mounted.

volume-label

Specifies the identification to be encoded on the volume. For a disk volume, you can specify a maximum of 12 alphanumeric characters;

for a magnetic tape volume, you can specify a maximum of 6 alphanumeric characters. Letters are automatically changed to uppercase. Nonalphanumeric characters are not allowed in the volume-label specification on disk.

qualifiers

/ACCESSED=number-of-directories

Requires OPER privilege. Affects Files-11 Structure Level 1 disks ONLY. Specifies, for disk volumes, the number of directories allowed in system space must be a value from 0 to 255. The default value is 3.

/BADBLOCKS=(area[,...])

Specifies, for disk volumes, faulty areas on the volume. The INITIALIZE command marks the areas as allocated so that no data is written in them.

Possible formats for area are as follows:

lbn[:count]	Logical block number of the first block and optionally a block count beginning with the first block, to be marked as allocated
sec.trk.cyl[:cnt]	Sector, track, and cylinder of the first block, and optionally a block count beginning with the first block, to be marked as allocated

/CLUSTER_SIZE=number-of-blocks

Defines, for disk volumes, the minimum allocation unit, in blocks. The maximum size you can specify for a volume is one-hundredth the size of the volume; the minimum size you can specify is calculated with the following formula:

$$\frac{\text{disk size}(\text{number of blocks})}{255 * 4096}$$

/DATA_CHECK[=(option[,...])]

Checks all read and write operations on the disk. By default, no data checks are made. Specify one or both options:

READ	Checks all read operations
WRITE	Checks all write operations; default if only /DATA_CHECK is specified

To override the checking you specify at initialization for disks, enter a MOUNT command to mount the volume.

/DENSITY=density-value

The /DENSITY qualifier is not applicable to the TK50 tape device. For floppy diskette volumes that are to be initialized on RX02 or RX33 diskette drives, specifies the density at which the floppy disk is to be formatted.

DCL-102 DCL Commands
INITIALIZE

RX02 dual-density diskette drives allow floppy diskettes to be initialized at single or double density. RX33 diskette drives allow floppy diskettes to be initialized at double density only. To specify single-density formatting of a floppy diskette, specify the density value SINGLE. To specify double-density formatting of a floppy diskette, specify the density value DOUBLE.

If you do not specify a density value for a floppy diskette being initialized on a drive, the system leaves the volume at the density to which the volume was last formatted.

For magnetic tape volumes, specifies the density in bytes per inch (bpi) at which the magnetic tape is to be written.

For magnetic tape volumes, the density value specified can be 800 bpi, 1600 bpi, or 6250 bpi, as long as the density is supported by the magnetic tape drive.

/DIRECTORIES=number-of-entries

Specifies, for disk volumes, the number of entries to preallocate for user directories. The number of entries must be an integer between 16 and 16000. The default value is 16.

/ERASE

/NOERASE (default)

Physically destroys deleted data (by writing over it).

/EXTENSION=number-of-blocks

Affects Files-11 Structure Level 1 disks ONLY. Specifies, for disk volumes, the number of blocks to use as a default extension size for all files on the volume. The extension default is used when a file increases to a size greater than its initial default allocation during an update. The value for the number-of-blocks parameter can range from 0 through 65,535. The default value is 5.

/FILE_PROTECTION=code

Affects Files-11 Structure Level 1 disks ONLY. Defines, for disk volumes, the default protection to be applied to all files on the volume.

/GROUP

Defines a group volume. The /GROUP qualifier applies protection of RWED to all ownership categories unless /GROUP is specified with /NOSHARE, in which case the volume protection is RWED for all but the world category. The owner UIC of the volume defaults to your group number and a member number of 0.

/HEADERS=number-of-headers

Specifies, for disk volumes, the number of file headers to be allocated for the index file. The minimum and default value is 16. The maximum is the value set with the /MAXIMUM_FILES qualifier.

This qualifier is useful when you want to create a number of files and want to streamline the process of allocating space for that number of file headers. If you do not specify this qualifier, the file system dynamically allocates space as it is needed for new headers on the volume.

/HIGHWATER (default)

/NOHIGHWATER

Affects Files-11 Structure Level 2 disks ONLY. Sets the file highwater mark (FHM) volume attribute, which guarantees that a user cannot read data that he has not written. You cannot specify **/NOHIGHWATER** for magnetic tape.

The **/NOHIGHWATER** qualifier disables FHM for a disk volume.

/INDEX=position

Specifies the location of the index file for the volume's directory structure. Possible positions are as follows:

BEGINNING	Beginning of the volume
MIDDLE	Middle of the volume (default)
END	End of the volume
BLOCK:n	Beginning of the logical block specified by <i>n</i>

/LABEL=option

Defines characteristics for the magnetic tape volume label, as directed by the included option. The available options are as follows:

- **OWNER_IDENTIFIER:**"(14 ANSI characters)"

Allows you to specify the Owner Identifier field in the volume label. The field specified can accept up to 14 ANSI characters.

- **VOLUME_ACCESSIBILITY:**"character"

Specifies the character to be written in the volume accessibility field of the VMS ANSI volume label VOL1 on an ANSI magnetic tape. The character may be any valid ANSI "a" character. This set of characters includes numeric characters, uppercase letters, and any one of the following nonalphanumeric characters:

! " % ' () * + , - . / : ; < = > ?

By default, the VMS operating system provides a routine that checks this field in the following manner.

- If the magnetic tape was created on a version of the VMS operating system that conforms to Version 3 of ANSI, then this option must be used to override any character other than an ASCII space.

DCL-104 DCL Commands
INITIALIZE

- If a VMS protection is specified and the magnetic tape conforms to an ANSI standard that is later than Version 3, then this option must be used to override any character other than an ASCII 1.

If you specify any character other than the default, you must specify the /**OVERWRITE=ACCESSIBILITY** qualifier on the **INITIALIZE** and **MOUNT** commands in order to access the magnetic tape.

/MAXIMUM_FILES=*n*

Restricts the maximum number of files that the volume can contain. The /**MAXIMUM_FILES** qualifier overrides the default value, which is calculated as follows:

$$\frac{\text{volume size in blocks}}{(\text{cluster factor} + 1) * 2}$$

The maximum size you can specify for any volume is as follows:

$$\frac{\text{volume size in blocks}}{(\text{cluster factor} + 1)}$$

The minimum value is 0. Note that the maximum can be increased only by reinitializing the volume.

NOTE: The **MAXIMUM_FILES** qualifier does not reserve or create space for new file headers on a volume. The file system dynamically allocates space as it is needed for new headers.

/OVERWRITE=(*option*[,...])

Requests the **INITIALIZE** command to ignore data on a magnetic tape volume that protects it from being overwritten. You may specify one or more of the following options:

ACCESSIBILITY

(For magnetic tapes only.) If the installation allows, this option overrides any character in the Accessibility Field of the volume. The necessity of this option is defined by the installation. That is, each installation has the option of specifying a routine that the magnetic tape file system will use to process this field. By default, VMS provides a routine that checks this field in the following manner. If the magnetic tape was created on a version of VMS that conforms to Version 3 of ANSI, this option must be used to override any character other than an ASCII space. If a VMS protection is specified and the magnetic tape conforms to an ANSI standard that is later than Version 3, this option must be used to override any character other than an ASCII 1. To use the **ACCESSIBILITY** option, you must have the user privilege **VOLPRO** or be the owner of the volume.

EXPIRATION (For magnetic tapes only.) Allows you to write to a tape that has not yet reached its expiration date. You must have the user privilege VOLPRO to override volume protection, or your UIC must match the UIC written on the volume.

OWNER_IDENTIFIER Allows you to override the processing of the Owner Identifier field of the volume label.

To initialize a volume that was initialized previously with the /PROTECTION qualifier, your UIC must match the UIC written on the volume or you must have VOLPRO privilege.

/OWNER_UIC=uic

Specifies an owner UIC for the volume. The default is your default UIC.

For magnetic tapes, no UIC is written unless protection on the magnetic tape is specified. If protection is specified, but no owner UIC is specified, your current UIC is assigned ownership of the volume.

/PROTECTION=(ownership[:access],...)

Applies the specified protection to the volume. Specify ownership as SYSTEM, OWNER, GROUP, or WORLD and access as R (read), W (write), E (execute), and D (delete). The default is your default protection.

For magnetic tape, the protection code is written to a VMS-specific volume label. The system only applies read and write access restrictions; execute and delete access are meaningless. Moreover, the system and the owner are always given both read and write access to magnetic tapes, regardless of the protection code you specify.

/SHARE (default)

/NOSHARE

Permits all categories of access by all categories of ownership. The /NOSHARE qualifier denies access to group (unless /GROUP is also specified) and world processes.

/STRUCTURE=level

Specifies whether the volume should be formatted in Files-11 Structure Level 1 or Structure Level 2 (the default). Level 1 is incompatible with the /DATA_CHECK and /CLUSTER_SIZE qualifiers. The default protection for a Structure Level 1 disk is full access to system, owner, and group, and R (read) access to all other users.

/SYSTEM

Requires a system UIC or SYSPRV privilege. Defines a system volume. The owner UIC defaults to [1,1]. Protection defaults to complete access by all ownership categories, except that only system processes can create top-level directories.

DCL-106 DCL Commands

INITIALIZE

/USER_NAME=name

Specifies a user name to be associated with the volume. The name must be 1 to 12 alphanumeric characters. The default is your user name.

/VERIFIED

/NOVERIFIED

Indicates whether the disk has bad block data on it. Use the */NOVERIFIED* qualifier to ignore bad block data on the disk. The default is */VERIFIED* for disks with 4096 blocks or more and */NOVERIFIED* for disks with less than 4096 blocks.

/WINDOWS=n

Specifies the number of mapping pointers (used to access data in the file) to be allocated for file windows. The value can be an integer in the range of 7 through 80. The default is 7.

example

```
$ INITIALIZE/USER_NAME=CPA $FLOPPY1 ACCOUNTS
```

Initializes the volume on \$FLOPPY1, labels the volume ACCOUNTS, and gives the volume a user name of CPA.

INITIALIZE/QUEUE

Creates or initializes queues. You use this command to create queues and to assign them names and attributes. The */QUEUE* qualifier is required. The */BATCH* qualifier is required to create a batch queue.

Requires OPER privilege.

format

```
INITIALIZE/QUEUE queue-name[:]
```

parameter

queue-name[:]

Specifies the name of an execution queue or a generic queue. The queue name may be a string of 1 to 31 characters. The character string can include any uppercase and lowercase letters, digits, the dollar sign (\$), and the underscore (_), and must include at least one alphabetic character.

qualifiers

/BASE_PRIORITY=n

Specifies the base process priority at which jobs are initiated from a batch execution queue. By default, if you omit the qualifier, jobs are initiated at the same priority as the base priority established by DEFPRI at system

generation (usually 4). The base priority specifier can be any decimal value from 0 through 15.

You also can specify this qualifier for an output execution queue. In this context the /BASE_PRIORITY qualifier establishes the base priority of the symbiont process when the symbiont process is created.

/BATCH

/NOBATCH (default)

Specifies that you are initializing a batch queue. If you are reinitializing an existing queue, you can use the /BATCH qualifier only if the queue was created as a batch queue. A batch queue is classified as either an execution or generic queue. By default, the /BATCH qualifier initializes an execution queue. To specify a generic batch queue, use the /GENERIC qualifier together with the /BATCH qualifier.

The /[NO]BATCH qualifier of the INITIALIZE/QUEUE command has superseded the /[NO]BATCH qualifier of the START/QUEUE command. Digital recommends that you use the INITIALIZE/QUEUE/[NO]BATCH command to determine queue type. Digital also recommends that you update command procedures that use START/QUEUE/[NO]BATCH. The /BATCH and /DEVICE qualifiers are mutually exclusive; the /NOBATCH and /NODEVICE qualifiers also cannot be used together.

/BLOCK_LIMIT=(*[lowlim,uplim]*)

/NOBLOCK_LIMIT (default)

Limits the size of print jobs that can be processed on an output execution queue. This qualifier allows you to reserve certain printers for certain size jobs. You must specify at least one of the parameters. The lowlim parameter is a decimal number referring to the minimum number of blocks accepted by the queue for a print job. If a print job is submitted that contains fewer blocks than the lowlim value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is decreased sufficiently, the job is processed. The uplim parameter is a decimal number referring to the maximum number of blocks that the queue accepts for a print job. If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is increased sufficiently, the job is processed.

/CHARACTERISTICS=(*characteristic[,...]*)

/NOCHARACTERISTICS (default)

Specifies one or more characteristics for processing jobs on an execution queue. If you specify only one characteristic, you can omit the parentheses. If a queue does not have all the characteristics that have been specified for a job, the job remains pending. Each time you specify /CHARACTERISTICS, all previously set characteristics are cancelled. Only the characteristics specified with the qualifier are established for the queue. Queue characteristics are installation-specific. The characteristic

DCL-108 DCL Commands

INITIALIZE/QUEUE

parameter can be either a value from 0 through 127 or a characteristic name that has been defined by the DEFINE/CHARACTERISTIC command.

/CLOSE

Prevents jobs from being entered in the queue through PRINT or SUBMIT commands or as a result of requeue operations. To allow jobs to be entered, use the /OPEN qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled). When a queue is marked closed, jobs executing continue to execute. Jobs pending in the queue continue to be candidates for execution.

/CPUDEFAULT=time

Defines the default CPU time limit for all jobs in this batch execution queue. You can specify time as delta time, 0, INFINITE, or NONE (default). You can specify up to 497 days of delta time.

If the queue does not have a specified CPUMAXIMUM time limit and the value established in the user authorization file (UAF) has a specified CPU time limit of NONE, either the value 0 or the keyword INFINITE allows unlimited CPU time. If you specify NONE, the CPU time value defaults to the value specified either in the UAF or by the SUBMIT command (if included). CPU time values must be greater than or equal to the number specified by the SYSGEN parameter PQL_MCPULM. The time cannot exceed the CPU time limit set by the /CPUMAXIMUM qualifier.

/CPUMAXIMUM=time

Defines the maximum CPU time limit for all jobs in a batch execution queue. You can specify time as delta time, 0, INFINITE, or NONE (default). You can specify up to 497 days of delta time.

The /CPUMAXIMUM qualifier overrides the time limit specified in the user authorization file (UAF) for any user submitting a job to the queue. Either the value 0 or the keyword INFINITE allows unlimited CPU time. If you specify NONE, the CPU time value defaults to the value specified either in the UAF or by the SUBMIT command (if included). CPU time values must be greater than or equal to the number specified by the SYSGEN parameter PQL_MCPULM.

/DEFAULT=(option[,...])

/NODEFAULT

Establishes defaults for certain options of the PRINT command. Defaults are specified by the list of options. If you specify only one option, you can omit the parentheses. After you set an option for the queue with the /DEFAULT qualifier, you do not have to specify that option in your PRINT commands. If you do specify these options in your PRINT command, the values specified with the PRINT command override the values established for the queue with the /DEFAULT qualifier.

You cannot use the /DEFAULT qualifier with the /GENERIC qualifier.

Possible options are as follows:

[NO]BURST[=keyword]	Controls whether two file flag pages with a burst bar between them are printed preceding output. If you specify the value ALL (default), these flag pages are printed before each file in the job. If you specify the value ONE, these flag pages are printed once before the first file in the job.
[NO]FEED	Controls whether a form feed is inserted automatically at the end of a page.
[NO]FLAG[=keyword]	Controls whether a file flag page is printed preceding output. If you specify the value ALL (default), a file flag page is printed before each file in the job. If you specify the value ONE, a file flag page is printed once before the first file in the job.
FORM=type	Specifies the default form for an output execution queue. If a job is submitted without an explicit form definition, this form is used to process the job. See also /FORM_MOUNTED.
[NO]TRAILER[=keyword]	Controls whether a file trailer page is printed following output. If you specify the value ALL (default), a file trailer page is printed after each file in the job. If you specify the value ONE, a trailer page is printed once after the last file in the job.

When you specify the BURST option for a file, the [NO]FLAG option does not add or subtract a flag page from the two flag pages that are printed preceding the file.

/DESCRIPTION=string
/NODESCRIPTION (default)

A string of up to 255 characters used to provide operator-supplied information about the queue.

Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (").

The /NODESCRIPTION qualifier removes any descriptive text that may be associated with the queue.

/DEVICE[=option]
/NODEVICE

Specifies that you are initializing an output queue of a particular type. If you are reinitializing an existing queue, you can use the /DEVICE qualifier only if the queue was created as an output queue. Possible options are as follows:

DCL-110 DCL Commands

INITIALIZE/QUEUE

PRINTER	Indicates a printer queue.
SERVER	Indicates a server queue. A server queue is controlled by the user-modified or user-written symbiont specified with the /PROCESSOR qualifier.
TERMINAL	Indicates a terminal queue.

If you specify the /DEVICE qualifier without a queue type, /DEVICE=PRINTER is used by default. An output queue is classified as either an execution or generic queue. By default, the /DEVICE qualifier initializes an execution queue of the designated type. To specify a generic printer, server, or terminal queue, use the /GENERIC qualifier with the /DEVICE qualifier. You specify the queue type with the /DEVICE qualifier for informational purposes. When an output execution queue is started, the symbiont associated with the queue determines the actual queue type. The /DEVICE and /BATCH qualifiers are mutually exclusive; the /NODEVICE and /NOBATCH qualifiers also cannot be used together.

/DISABLE_SWAPPING

/NODISABLE_SWAPPING (default)

Controls whether batch jobs executed from a queue can be swapped in and out of memory.

/ENABLE_GENERIC (default)

/NOENABLE_GENERIC

Specifies whether files queued to a generic queue that does not specify explicit queue names with the /GENERIC qualifier can be placed in this execution queue for processing. For more information, see the description of the /GENERIC qualifier.

/FORM_MOUNTED=type

Specifies the mounted form for an output execution queue. If the stock of the mounted form does not match the stock of the default form, as indicated by the /DEFAULT=FORM qualifier, all jobs submitted to this queue without an explicit form definition enter a pending state. If a job is submitted with an explicit form and the stock of the explicit form is not identical to the stock of the mounted form, the job enters a pending state. In both cases, jobs remain pending until the stock of the mounted form of the queue is identical to the stock of the form associated with the job. To specify the form type, use either a numeric value or a form name that has been defined by the DEFINE/FORM command. Form types are installation-specific. You cannot use the /FORM_MOUNTED qualifier with the /GENERIC qualifier.

/GENERIC[=(queue-name[,...])]

/NOGENERIC (default)

Specifies a generic queue. Also specifies that jobs placed in this queue can be moved for processing to compatible execution queues. The /GENERIC qualifier optionally accepts a list of target execution queues that have been previously defined. For a generic batch queue, these target queues

must be batch execution queues. For a generic output queue, these target queues must be output execution queues, but can be of any type (printer, server, or terminal). If you do not specify any target execution queues with the /GENERIC qualifier, jobs can be moved to any execution queue that (1) is initialized with the /ENABLE_GENERIC qualifier, and (2) is the same type (batch or output) as the generic queue. To define the queue as a generic batch or output queue, you use the /GENERIC qualifier with either the /BATCH or /DEVICE qualifiers. If you specify neither /BATCH nor /DEVICE on creation of a generic queue, the queue becomes a generic printer queue by default.

/JOB_LIMIT=n

Indicates the number of batch jobs that can be executed concurrently from the queue. Specify a number in the range 0 through 255. The job limit default value for n is 1.

/LIBRARY=file-name
/NOLIBRARY

Specifies the file name for the device control library. When you initialize an output execution queue, you can use the /LIBRARY qualifier to specify an alternate device control library. The default library is SYS\$LIBRARY:SYSDEVCTL.TLB. You can use only a file name as the parameter of the /LIBRARY qualifier. The system always assumes that the file is located in SYS\$LIBRARY and that the file type is TLB.

/ON=[node::]device[:] (*printer, terminal, server queue*)
/ON=node:: (*batch queue*)

Specifies the node or device, or both, on which this execution queue is located. For batch execution queues, you can specify only the node name. For output execution queues, you can include both the node name and the device name. By default, a queue executes on the same node from which you start the queue. The default device parameter is the same as the queue name.

/OPEN (default)

Allows jobs to be entered in the queue through PRINT or SUBMIT commands or as the result of requeue operations. To prevent jobs from being entered in the queue, use the /CLOSE qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled).

/OWNER_UIC=uic

Enables you to change the user identification code (UIC) of the queue. The default UIC is [1,4].

/PROCESSOR=file-name
/NOPROCESSOR

Allows you to specify your own print symbiont for an output execution queue. You can use any valid file name as a parameter of the

DCL-112 DCL Commands
INITIALIZE/QUEUE

/PROCESSOR qualifier. The system supplies the device and directory name SYS\$SYSTEM and the file type EXE. If you use this qualifier for an output queue, it specifies that the symbiont image to be executed is SYS\$SYSTEM:filename.EXE.

/PROTECTION=(ownership[:access],...)

Specifies the protection of the queue. Ownership categories are SYSTEM, OWNER, GROUP, WORLD; each category can be abbreviated to its first character. Access categories are R (READ), W (WRITE), E (EXECUTE), and D (DELETE); a null access specification means no access. The default protection is: (SYSTEM:E, OWNER:D, GROUP:R, WORLD:W).

/RECORD_BLOCKING (default)

/NORECORD_BLOCKING

Determines whether the symbiont can concatenate (or block together) output records for transmission to the output device. If you specify /NORECORD_BLOCKING, the symbiont sends each formatted record in a separate I/O request to the output device. For the standard VMS print symbiont, record blocking can have a significant performance advantage over single-record mode.

/RETAIN[=option]

/NORETAIN (default)

Holds jobs in the queue in a retained status after they have executed. The /NORETAIN qualifier enables you to reset the queue to the default. Possible options are as follows:

ALL (default)	Holds all jobs in the queue after execution
ERROR	Holds in the queue only jobs that complete unsuccessfully

/SCHEDULE=[NO]SIZE

Specifies whether pending jobs in an output execution queue are scheduled for printing based on the size of the job. When the default, /SCHEDULE=SIZE, is in effect, shorter jobs print before longer ones. When /SCHEDULE=NOSIZE is in effect, jobs are printed in the order they were submitted, regardless of size.

If you enter this command while there are pending jobs in any queue, its effect on future jobs is unpredictable.

/SEPARATE=(option[,...])

/NOSEPARATE (default)

Specifies the mandatory queue attributes, or job separation options, for an output execution queue. Job separation options cannot be overridden by the PRINT command.

You cannot use the /SEPARATE qualifier with the /GENERIC qualifier.

The job separation options are as follows:

[NO]BURST	Specifies whether two job flag pages with a burst bar between them are printed at the beginning of each job.
[NO]FLAG	Specifies whether a job flag page is printed at the beginning of each job.
[NO]TRAILER	Specifies whether a job trailer page is printed at the end of each job.
[NO]RESET=(module[,...])	Specifies one or more device control library modules that contain the job reset sequence for the queue. The specified modules from the queue's device control library (by default SYS\$LIBRARY:SYSDEVCTL) are used to reset the device each time a job reset occurs. The RESET sequence occurs after any file trailer and before any job trailer. Thus, all job separation pages are printed when the device is in its RESET state.

When you specify /SEPARATE=BURST, the [NO]FLAG separation option does not add or subtract a flag page from the two flag pages that are printed preceding the job.

For information on establishing queue attributes that can be overridden, see the description of the /DEFAULT qualifier.

/START

/NOSTART (default)

Starts the queue being initialized by the current INITIALIZE/QUEUE command.

/TERMINAL

/NOTERMINAL (default)

Indicates that the output queue is a terminal queue. The /NOTERMINAL qualifier cancels the effect of a previous /TERMINAL qualifier on the same command. It is supported in this release for compatibility with VMS V4.n. The /[NO]DEVICE qualifier has superseded the /[NO]TERMINAL qualifier. Digital recommends that you use the /[NO]DEVICE qualifier to determine queue type. Digital also recommends that you use this qualifier to update command procedures that use INITIALIZE/QUEUE/[NO]TERMINAL.

/WSDEFAULT=n

Defines for a batch job a working set default, the default number of physical pages that the job can use. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

If you specify 0 or NONE, the working set default value defaults to the value specified in the UAF or by the SUBMIT command (if included).

DCL-114 DCL Commands

INITIALIZE/QUEUE

You also can specify this qualifier for an output execution queue. Used in this context, `/WSDEFAULT` establishes the working set default of the symbiont process for an output execution queue when the symbiont process is created.

`/WSEXTENT=n`

Defines for the batch job a working set extent, the maximum amount of physical memory that the job can use. The job only uses the maximum amount of physical memory when the system has excess free pages. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

If you specify 0 or NONE, the working set extent value defaults to the value specified in the UAF or by the `SUBMIT` command (if included).

You also can specify this qualifier for an output execution queue. Used in this context, `/WSEXTENT` establishes the working set extent of the symbiont process for an output execution queue when the symbiont process is created.

`/WSQUOTA=n`

Defines for a batch job a working set quota, the amount of physical memory that is guaranteed to the job.

The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. If you specify 0 or NONE, the working set quota value defaults to the value specified in the UAF or by the `SUBMIT` command (if included).

You also can specify this qualifier for an output execution queue. Used in this context, `/WSQUOTA` establishes the working set quota of the symbiont process for an output execution queue when the symbiont process is created.

example

```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=3 SYS$BATCH
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=1/WSEXTENT=2000 BIG_BATCH
```

In this example, the first `INITIALIZE/QUEUE` command creates a batch queue called `SYS$BATCH` that can be used for any batch job. The `/JOB_LIMIT` qualifier allows three jobs to execute concurrently. The second `INITIALIZE/QUEUE` command creates a second batch queue called `BIG_BATCH` that is designed for large jobs. Only one job can execute at a time. The working set extent can be as high as 2000.

INQUIRE

Reads a value from SYS\$COMMAND (usually the terminal in interactive mode or the next line in the main command procedure) and assigns it to a symbol.

format

INQUIRE *symbol-name* [*prompt-string*]

parameters

symbol-name

Specifies a 1- through 255-alphanumeric character symbol to be given a value.

prompt-string

Specifies the prompt to be displayed at the terminal when the INQUIRE command is executed. String values are automatically converted to uppercase. Also, any leading and trailing spaces and tabs are removed, and multiple spaces and tabs between characters are compressed to a single space. Enclose the prompt in quotation marks (") if it contains lowercase characters, punctuation, multiple blanks or tabs, or an at sign (@). To denote an actual quotation mark in a prompt-string, enclose the entire string in quotation marks and use two consecutive quotation marks ("") within the string. If you do not specify a prompt string, the command interpreter uses the symbol name to prompt for a value.

qualifiers

/GLOBAL

Specifies that the symbol be placed in the global symbol table. If you do not specify the /GLOBAL qualifier, the symbol is placed in the local symbol table.

/LOCAL (default)

Specifies that the symbol be placed in the local symbol table for the current command procedure.

/PUNCTUATION (default)

/NOPUNCTUATION

Inserts a colon (:) and a space after the prompt when it is displayed on the terminal. To suppress the colon and space, specify /NOPUNCTUATION.

DCL-116 DCL Commands

INQUIRE

example

```
$ INQUIRE CHECK "Enter Y[ES] to continue"  
$ IF .NOT. CHECK THEN EXIT
```

The INQUIRE command displays the following prompting message at the terminal:

```
Enter Y[ES] to continue:
```

The INQUIRE command prompts for a value, which is assigned to the symbol CHECK. The IF command tests the value assigned to the symbol CHECK. If the value assigned to CHECK is true (that is, an odd numeric value, a character string that begins with a T, t, Y, or y, or an odd numeric character string), the procedure continues executing.

If the value assigned to CHECK is false (that is, an even numeric value, a character string that begins with any letter except T, t, Y, or y, or an even numeric character string), the procedure exits.

INSTALL

Invokes the Install Utility, which enhances the performance of selected executable and shareable images by making them "known" to the system and assigning them appropriate attributes. For more information about the Install Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

JOB

Identifies the beginning of a batch job submitted through a card reader. Each batch job submitted through the system card reader must be preceded by a JOB card.

JOB cannot be abbreviated.

format

```
$ JOB user-name
```

parameter

user-name

Identifies the user name under which the job is to be run. Specify the user name as you would during the login procedure.

qualifiers

/AFTER=time

Holds the job until the specified time. If the specified time has already passed, the job is queued for immediate processing. The time can be specified as either an absolute time or a combination of absolute and delta times.

/CHARACTERISTICS=(characteristic[,...])

Specifies one or more characteristics required for processing the job. If you specify only one characteristic, you can omit the parentheses. Codes for characteristics are installation-defined. Use the SHOW QUEUE/CHARACTERISTICS command to see which characteristics are available on your system. All the characteristics specified for the job must also be specified for the queue that will execute the job. If not, the job remains pending in the queue until the queue characteristics are changed or the entry is deleted with the DELETE/ENTRY command. Users need not specify every characteristic of a queue with the JOB command as long as the ones they specify are a subset of the characteristics set for that queue. The job also runs if no characteristics are specified.

/CLI=file-name

Specifies a different command language interpreter (CLI) with which to process the job. The file name specifies that the CLI be SYS\$SYSTEM:filename.EXE. The default CLI is that defined in the user authorization file (UAF).

/CPUIME=n

Specifies a CPU time limit for the batch job. Time can be specified as delta time, 0, NONE, or INFINITE. Specify 0 or INFINITE to request an infinite amount of time. Specify NONE when you want the CPU time to default to your UAF value or the limit specified on the queue. Note that you cannot request more time than permitted by the base queue limits or your UAF.

/DELETE (default)

/NODELETE

Controls whether the batch input file is deleted after the job is processed. If you specify /NODELETE, the file is saved in the user's default directory under the default name INPBATCH.COM. If you specify the /NAME qualifier, the file name of the batch input file is the same as the job name you supply with /NAME.

/HOLD

/NOHOLD (default)

Controls whether or not the job is to be made available for immediate processing.

If you specify `/HOLD`, the job is not released for processing until you specifically release it with the `/NOHOLD` or `/RELEASE` qualifier of the `SET QUEUE/ENTRY` command.

`/KEEP`**`/NOKEEP (default)`**

Controls whether the log file is deleted after it is printed. `/NOKEEP` is the default unless `/NOPRINTER` is specified.

`/LOG_FILE=file-spec`**`/NOLOG_FILE`**

Controls whether a log file with the specified name is created for the job or whether a log file is created. When you use the `/LOG_FILE` qualifier, the system writes the log file to the file you specify. If you use `/NOLOG_FILE`, no log file is created. If you specify neither form of the qualifier, the log file is written to a file in your default directory that has the same file name as the first command file in the job and a file type of `LOG`. Using neither `/LOG_FILE` nor `/NOLOG_FILE` is the default. You can use the `/LOG_FILE` qualifier to specify that the log file be written to a different device.

`/NAME=job-name`

Specifies a file name string to be used as the job name and as the file name for both the batch job log file and the command file. The job name must be 1 to 39 alphanumeric characters and must be a valid file name. The default log file name is `INPBATCH.LOG`; the default command file name is `INPBATCH.COM`.

`/NOTIFY`**`/NONOTIFY (default)`**

Controls whether a message is broadcast to any terminal at which you are logged in, notifying you when your job completes or aborts.

`/PARAMETERS=(parameter[,...])`

Specifies from 1 through 8 optional parameters that can be passed to the command procedure. The parameters define values to be equated to the symbols `P1` through `P8` in the batch job. The symbols are local to the specified command procedure. If you specify only one parameter, you can omit the parentheses. The commas delimit individual parameters. If the parameter contains any spaces, special characters or delimiters, or lowercase characters, enclose it in quotation marks. Individual parameters cannot exceed 255 characters.

`/PRINTER=queue-name`**`/NOPRINTER`**

Controls whether the job log file is queued to the specified queue for printing when the job is complete. The default print queue for the log file is `SY$PRINT`.

/PRIORITY=n

Requires OPER or alter priority (ALTPRI) privilege to raise the priority above the value of the SYSGEN parameter MAXQUEPRI. Specifies the job scheduling priority for the specified job. The value of *n* is an integer from 0 through 255, where 0 is the lowest priority and 255 is the highest. The default value for /PRIORITY is the value of the SYSGEN parameter DEFQUEPRI.

/QUEUE=queue-name[:]

Specifies the name of the batch queue in which the job is to be entered. If you do not specify /QUEUE, the job is placed in the default system batch job queue, SYS\$BATCH.

/RESTART

/NORESTART (default)

Specifies whether the job restarts after a system failure or a STOP/QUEUE/REQUEUE command.

/TRAILING_BLANKS (default)

/NOTRAILING_BLANKS

Controls whether input cards in the card deck are read in card image form or input records are truncated at the last nonblank character. By default, the system does not remove trailing blanks from records read through the card reader.

/WSDEFAULT=n

Defines a working set default for the batch job; the /WSDEFAULT qualifier overrides the working set size specified in the user authorization file (UAF).

N can be any positive integer from 1 to 65,535, 0, or the keyword NONE. A value of 0 or the keyword NONE sets the default value to the value specified either in your UAF or by the working set quota established for the queue. You cannot request a value higher than your default.

/WSEXTENT=n

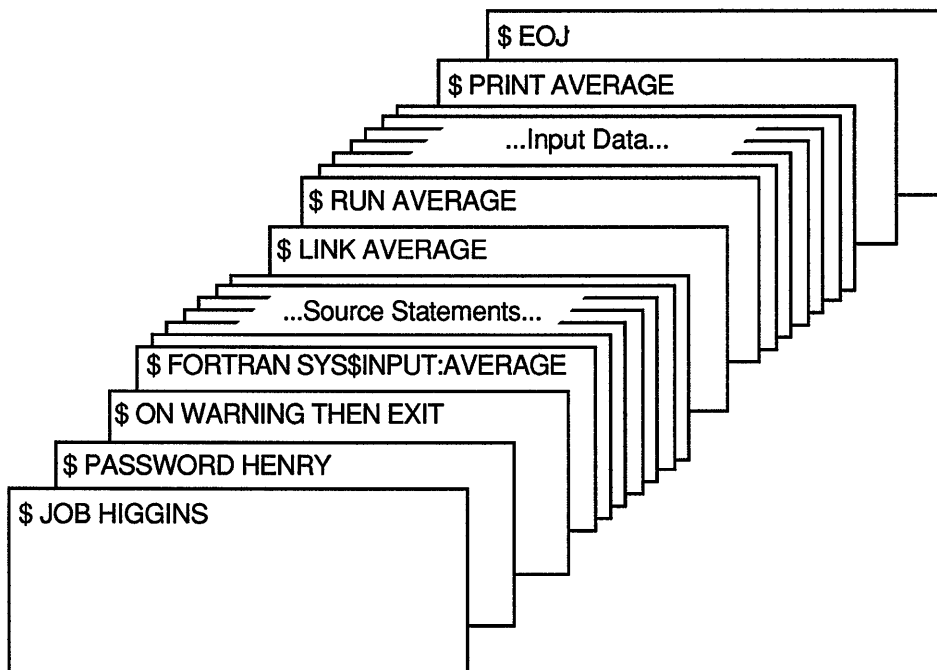
Defines a working set extent for the batch job; the /WSEXTENT qualifier overrides the working set extent in the UAF. *N* can be any positive integer from 1 to 65,535, 0, or the keyword NONE. A value of 0 or the keyword NONE sets the default value either to the value specified in the UAF or working set extent established for the queue. You cannot request a value higher than your default.

/WSQUOTA=n

Defines the maximum working set size (working set quota) for the batch job; the /WSQUOTA qualifier overrides the value in the UAF. *N* can be any positive integer from 1 to 65,535, 0, or the keyword NONE. You cannot request a value higher than your default.

DCL-120 DCL Commands
JOB

example



ZK-0787-GE

The JOB and PASSWORD cards identify and authorize the user HIGGINS to enter batch jobs. The command stream consists of a FORTRAN command and FORTRAN source statements to be compiled. The file name AVERAGE following the device name SYS\$INPUT provides the compiler with a file name for the object and listing files. The output files are cataloged in user HIGGINS's default directory.

If the compilation is successful, the LINK command creates an executable image and the RUN command executes it. Input for the program follows the RUN command in the command stream. The last command in the job prints the program listing. The last card in the deck contains the EOJ (end of job) command.

Lexical Functions

A set of functions that return information about character strings and attributes of the current process.

description

The command language includes constructs, called lexical functions, that return information about the current process and about arithmetic and string expressions. The general format of a lexical function is as follows:

F\$function-name([args,...])

F\$	Indicates that what follows is a lexical function.
function-name	A keyword specifying the function to be evaluated. Function names can be truncated to any unique abbreviation.
()	Enclose function arguments, if any. The parentheses are required for all functions, including functions that do not accept any arguments.
args,...	Specify arguments for the function, if any, using integer or character string expressions.

Table DCL-1 lists each lexical function and briefly describes the information that each function returns. A detailed description of each function, including examples, is given in the following pages.

Table DCL-1: Summary of Lexical Functions

Function	Description
F\$CONTEXT	Specifies selection criteria for use with the F\$PID function.
F\$CVSI	Extracts bit fields from character string data and converts the result, as a signed value, to an integer.
F\$CVTIME	Retrieves information about an absolute, combination, or delta time string.
F\$CVUI	Extracts bit fields from character string data and converts the result, as an unsigned value, to an integer.
F\$DIRECTORY	Returns the current default directory name string.
F\$EDIT	Edits a character string based on the edits specified.

(continued on next page)

Table DCL-1 (Cont.): Summary of Lexical Functions

Function	Description
F\$ELEMENT	Extracts an element from a string in which the elements are separated by a specified delimiter.
F\$ENVIRONMENT	Obtains information about the DCL command environment.
F\$EXTRACT	Extracts a substring from a character string expression.
F\$FAO	Invokes the \$FAO system service to convert the specified control string to a formatted ASCII output string.
F\$FILE_ATTRIBUTES	Returns attribute information for a specified file.
F\$GETDVI	Invokes the \$GETDVI system service to return a specified item of information for a specified device.
F\$GETJPI	Invokes the \$GETJPI system service to return accounting, status, and identification information for a process.
F\$GETQUI	Invokes the \$GETQUI system service to return information about queues, batch and print jobs currently in those queues, form definitions, and characteristic definitions kept in the system job queue file.
F\$GETSYI	Invokes the \$GETSYI system service to return status and identification information about the local system, or about a node in the local cluster, if your system is part of a cluster.
F\$IDENTIFIER	Converts an identifier in named format to its integer equivalent, or vice versa.
F\$INTEGER	Returns the integer equivalent of the result of the specified expression.
F\$LENGTH	Returns the length of a specified string.
F\$LOCATE	Locates a character or character substring within a string and returns its offset within the string.
F\$MESSAGE	Returns the message text associated with a specified system status code value.
F\$MODE	Shows the mode in which a process is executing.
F\$PARSE	Invokes the \$PARSE RMS service to parse a file specification and return either the expanded file specification or the particular file specification field that you request.
F\$PID	For each invocation, returns the next process identification number in sequence.

(continued on next page)

Table DCL-1 (Cont.): Summary of Lexical Functions

Function	Description
F\$PRIVILEGE	Returns a value of "TRUE" or "FALSE" depending on whether your current process privileges match the privileges listed in the argument.
F\$PROCESS	Returns the current process name string.
F\$SEARCH	Invokes the \$SEARCH RMS service to search a directory file, and returns the full file specification for a file you name.
F\$SETPRV	Sets the specified privileges and returns a list of keywords indicating the previous state of these privileges for the current process.
F\$STRING	Returns the string equivalent of the result of the specified expression.
F\$TIME	Returns the current date and time of day, in the format dd-mm-yy hh:mm:ss.cc.
F\$TRNLNM	Translates a logical name and returns the equivalence name string or the requested attributes of the logical name.
F\$TYPE	Determines the data type of a symbol.
F\$USER	Returns the current user identification code (UIC).
F\$VERIFY	Returns the integer 1 if command procedure verification is set on; returns the integer 0 if command procedure verification is set off. The F\$VERIFY function also can set new verification states.

F\$CONTEXT

Specifies selection criteria for use with the F\$PID function. The F\$CONTEXT function enables the F\$PID function to obtain information about processes from any node in a VAXcluster.

format

F\$CONTEXT(*context-type, context-symbol, selection-item, selection-value, value-qualifier*)

arguments

context-type

The type of context to be built. The following table shows the valid context type that is currently available and the lexical function for which a context is built.

Context Type	Description
PROCESS	For use in constructing selection criteria for F\$PID

context-symbol

A symbol that DCL uses to refer to the context memory being constructed by F\$CONTEXT. F\$PID uses this context symbol to process the appropriate list of PIDs.

Specify the context symbol by using a symbol. The first time you use the F\$CONTEXT function in a command procedure, use a symbol that is either undefined or equated to the null string (""). The symbol created will be a local symbol of type "PROCESS_CONTEXT". When the context is no longer valid—that is, when all PIDs have been retrieved by F\$PID calls or an error occurs during one of these calls—the symbol no longer has a type of "PROCESS_CONTEXT". Then you can use the F\$TYPE function in the command procedure to find out if it is necessary to cancel the context.

After setting up the selection criteria, use this context symbol when calling F\$PID.

selection-item

The selection item is a keyword that tells F\$CONTEXT which selection criteria to use. Use only one selection-item keyword per call to F\$CONTEXT.

The following table shows valid selection-item keywords for the PROCESS context-type:

Selection Item	Selection Value	Value Qualifiers	Comments
ACCOUNT	string	EQL, NEQ	Valid account name or list of names. Wildcard characters are allowed.
AUTHPRI	integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Valid authorized base priority (0-31).

Selection Item	Selection Value	Value Qualifiers	Comments
CANCEL			This keyword will cancel the selection criteria for this context.
CURPRIV	keyword	ALL, ANY, EQL, NEQ	Valid privilege name keyword or list of keywords.
GRP	string	GEQ, GTR, LEQ, LSS, EQL, NEQ	Group number or name.
HW_MODEL	integer	EQL, NEQ	Valid hardware model number.
HW_NAME	string	EQL, NEQ	Valid hardware name or a list of keywords. Wildcard characters are allowed.
JOBPRCNT	integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Subprocess count for entire job.
JOBTYP	keyword	EQL, NEQ	Valid job-type keyword. Valid keywords are DETACHED, NETWORK, BATCH, LOCAL, DIALUP, and REMOTE.
MASTER_PID	string	EQL, NEQ	PID of master process.
MEM	string or integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	UIC member number or name.
MODE	keyword	EQL, NEQ	Valid process mode keyword. Valid keywords are OTHER, NETWORK, BATCH, and INTERACTIVE.
NODE_CSID	integer	EQL, NEQ	Node's cluster ID number.
NODENAME	string	EQL, NEQ	Node name or list of node names. Wildcard characters are allowed. The default is your local node. To request all nodes, use the value "*".
OWNER	string	EQL, NEQ	PID of immediate parent process.
PRCNT	integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Subprocess count of process.
PRCNAM	string	EQL, NEQ	Process name or list of process names. Wildcard characters are allowed.
PRI	integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Process priority level number (0-31).

F\$CONTEXT

Selection Item	Selection Value	Value Qualifiers	Comments
PRIB	integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Base process priority level number (0-31).
STATE	keyword	EQL, NEQ	Valid process state keyword.
STS	keyword	EQL, NEQ	Valid process status keyword.
TERMINAL	string	EQL, NEQ	Terminal name or list of names. Wildcard characters are allowed.
UIC	string	EQL, NEQ	UIC identifier (that is, of the form "[group,member]").
USERNAME	string	EQL, NEQ	User name or list of user names. Wildcard characters are allowed.

selection-value

Value of the selection criteria. For example, to process all the processes running on node MYVAX, specify "MYVAX" with the "NODENAME" keyword. For example:

```
$ X = F$CONTEXT("PROCESS", ctx, "NODENAME", "MYVAX", "EQL")
```

Values that are lists are valid with some selection items. The items must be separated by commas. The following example specifies a list of the nodes MYVAX, HERVAX, and HISVAX:

```
$ X = F$CONTEXT("PROCESS", ctx, "NODENAME", "MYVAX,HERVAX,HISVAX", "EQL")
```

You can use wildcard characters for some values. Using wildcard characters for selection items is similar to using wildcard characters for file names. Use the asterisk (*) character to match zero or more characters. Use the percent (%) character to match exactly one character.

value-qualifier

You must qualify selection values. You can qualify a number, for example, by requesting that the selection be based on the process value less than (LSS), less than or equal to (LEQ), greater than (GTR), greater than or equal to (GEQ), equal to (EQL), or not equal to (NEQ) the value specified in the call to F\$PID.

You can qualify some lists with the ALL, ANY, EQL, or NEQ keywords. Such lists are usually masks such as the process privilege mask, which consists of the set of enabled privileges. ALL requires that all items in the list be true for a process; ANY requests that any item in the list be part of the attributes of a process; EQL means that the values must match exactly (that is, values not specified must not be true of the process); and NEQ requires that the value must not match.

F\$CONTEXT

The difference between ALL and EQL is that the values specified with ALL must exist, but other unspecified values can exist also. EQL requires that all values specified must exist, and all others may not. For example, to request those processes whose current privileges include TMPMBX and OPER, but may include other privileges, specify the ALL keyword. To request those processes whose current privileges are TMPMBX and OPER exclusively, specify the EQL keyword.

description

Use the F\$CONTEXT function to set up selection criteria for the F\$PID function.

The F\$CONTEXT function is called as many times as necessary to produce the criteria needed; however, each call can specify only one selection item. Lists of item values are allowed, where appropriate, and more than one context can be operated upon at a time.

After establishing the selection criteria with appropriate calls to F\$CONTEXT, F\$PID is called repeatedly to return all the process identification numbers (PIDs) that meet the criteria specified in the F\$CONTEXT function. When there are no more such processes, the F\$PID function returns a null string.

After the F\$PID function is called, the context symbol is considered "frozen"; F\$CONTEXT cannot be called again with the same context symbol until the associated context selection criteria have been deleted. If you attempt to set up additional selection criteria with the same context symbol, an error message is displayed. However, the context and selection criteria are not affected and calls to F\$PID can continue.

The F\$CONTEXT function uses process memory to store the selection criteria. This memory is deleted under two circumstances. Memory is deleted when F\$PID is called and a null string is returned—that is, when all processes that meet the selection criteria have been returned. Memory also is deleted if the CANCEL selection-item keyword is used in a call to F\$CONTEXT with an established context. This type of call is appropriate for a CTRL/Y or another condition handling routine.

DCL-128 Lexical Functions

F\$CONTEXT

example

```
#!/Establish an error and CTRL/Y handler
$!
$ ON ERROR THEN GOTO error
$ ON CONTROL_Y THEN GOTO error
$!
$ ctx = ""
$ temp = F$CONTEXT ("PROCESS", ctx, "NODENAME", "*", "EQL")
$ temp = F$CONTEXT ("PROCESS", ctx, "USERNAME", "M*,SYSTEM", "EQL")
$ temp = F$CONTEXT ("PROCESS", ctx, "CURPRIV", "SYSPRV,OPER", "ALL")
$!
$!Loop over all processes that meet the selection criteria.
$!Print the PID and the name of the image for each process.
$!
$loop:
$ pid = F$PID(ctx)
$ IF pid .EQS. ""
$ THEN
$     GOTO endloop
$ ELSE
$     image = F$GETJPI(pid,"IMAGENAME")
$     SHOW SYMBOL pid
$     WRITE SYS$OUTPUT image
$     GOTO loop
$ ENDIF
$!The loop over the processes has ended.
$!
$endloop:
$!
$ EXIT
$!
$!Error handler. Clean up the context's memory with the CANCEL selection
$!item keyword.
$!
$error:
$ IF F$TYPE(ctx) .eqs. "PROCESS_CONTEXT" THEN -
-$ temp = F$CONTEXT ("PROCESS", ctx, "CANCEL")
$!
$ EXIT
```

In this example, F\$CONTEXT is called three times to set up selection criteria. The first call requests that the search take place on all nodes in the cluster. The second call requests that only the processes whose user name either starts with an "M" or is "SYSTEM" be processed. The third call restricts the selection to those processes whose current privileges include both SYSPRV and OPER and can have other privileges set.

The command lines inside of *loop* and *endloop* continually call F\$PID to obtain the processes that meet the criteria set up in the F\$CONTEXT calls. After retrieving each PID, F\$GETJPI is called to return the name of the image running in the process. Finally, the procedure displays the name of the image.

In case of error or CTRL/Y, control is passed to *error* and the context is closed if necessary. In this example, note the check for the symbol type `PROCESS_CONTEXT`. If the symbol has this type, selection criteria must be canceled by a call to `F$CONTEXT`. If the symbol is not of the type `PROCESS_CONTEXT`, either selection criteria have not been set up yet in `F$CONTEXT`, or the symbol has been used with `F$PID` until an error occurred or until the end of the process list was reached.

F\$CVSI

Converts the specified bits in the specified character string to a signed number.

format

F\$CVSI(*start-bit,number-of-bits,string*)

arguments

start-bit

The offset of the first bit to be extracted. The low-order (rightmost) bit of a string is position number 0 for determining the offset. Specify the offset as an integer expression.

number-of-bits

The length of the bit string to be extracted, which must be less than or equal to the number of bits in the string.

string

The string from which the bits are taken. Specify the string as a character string expression.

example

```
$ A[0,32] = %X2B
$ SHOW SYMBOL A
A = "+..."
$ X = F$CVSI(0,4,A)
$ SHOW SYMBOL X
X = -5   Hex = FFFFFFFB   Octal = 3777777773
```

This example uses an arithmetic overlay to assign the hexadecimal value 2B to all 32 bits of the symbol A. See the description of the Assignment Statement for more information on arithmetic overlays.

The symbol A has a string value after the overlay because it was previously undefined. (If a symbol is undefined, it has a string value as a result of an arithmetic overlay. If a symbol was previously defined, it retains the same data type after the overlay.) The hexadecimal value 2B corresponds to the ASCII value of the plus sign (+).

Next, the F\$CVSI function extracts the low-order 4 bits from the symbol A; the low-order 4 bits contain the binary representation of the hexadecimal value B. These bits are converted, as a signed value, to an integer. The converted value, -5, is assigned to the symbol X.

F\$CVTIME

Converts an absolute or a combination time string to a string of the form *yyyy-mm-dd hh:mm:ss.cc*. The F\$CVTIME function can also return information about an absolute, combination, or delta time string.

format

F\$CVTIME(*[input_time]* [*output_time_format*] [*output_field*])

arguments

input_time

Specifies a string containing an absolute, combination, or delta time, or TODAY, TOMORROW, or YESTERDAY. Specify the input time string as a character string expression. If the *input_time* argument is omitted or specified as a null string (""), the current system date and time, in absolute format, is used. If parts of the date field are omitted, the missing values default to the current date. If parts of the time field are omitted, the missing values default to zero. If the *input_time* argument is a delta time, you must specify the *output_time_format* argument as DELTA.

output_time_format

Specifies the time format for the information you want returned. Specify the *output_time_format* argument as one of the following character string expressions:

ABSOLUTE	The requested information should be returned in absolute time format, which is <i>dd-mmm-yyyy hh:mm:ss.cc</i> .
COMPARISON (default)	The requested information should be returned in the form <i>yyyy-mm-dd hh:mm:ss.cc</i> ; used for comparing two times.
DELTA	The requested information should be returned in delta format, which is <i>ddd-hh:mm:ss.cc</i> . If the <i>input_time</i> argument is a delta time, the <i>output_time_format</i> argument must be DELTA.

output_field

Specifies a character string expression containing one of the following (do not abbreviate): DATE, MONTH, DATETIME (default), SECOND, DAY, TIME, HOUR, WEEKDAY, HUNDREDTH, YEAR, MINUTE. The information is returned in the time format specified by the *output_time_format* argument. If the *input_time* argument is a delta time and the *output_time_format* argument is DELTA, you cannot specify MONTH, WEEKDAY, or YEAR.

example

```
$ TIME = F$TIME ()
$ SHOW SYMBOL TIME
TIME = "19-APR-1990 10:56:23.10"
$ TIME = F$CVTIME (TIME)
$ SHOW SYMBOL TIME
TIME = "1990-04-19 10:56:23.10"
```

This example uses the **F\$TIME** function to return the system time as a character string and to assign the time to the symbol **TIME**. Then the **F\$CVTIME** function is used to convert the system time to an alternate time format. Note that you do not need to place quotation marks around the argument **TIME** because it is a symbol. Symbols are automatically evaluated when they are used as arguments for lexical functions.

You can use the resultant string to compare two dates (using **.LTS.** and **.GTS.** operators). For example, you can use **F\$CVTIME** to convert two time strings and store the results in the symbols **TIME_1** and **TIME_2**. You can compare the two values, and branch to a label, based on the following results:

```
$ IF TIME_1 .LTS. TIME_2 THEN GOTO FIRST
```

F\$CVUI

Extracts bit fields from character string data and converts the result to an unsigned number.

format

F\$CVUI(*start-bit,number-of-bits,string*)

arguments

start-bit

Specifies the offset of the first bit to be extracted. The low-order (rightmost) bit of a string is position number 0 for determining the offset. Specify the offset as an integer expression.

number-of-bits

Specifies the length of the bit-string to be extracted, which must be less than or equal to the number of bits in the string argument.

string

Specifies the character string to be edited.

example

```
$ A[0,32] = %X2B
$ SHOW SYMBOL A
A = "+..."
$ X = F$CVUI(0,4,A)
$ SHOW SYMBOL X
X = 11 Hex = 0000000B Octal = 0000000013
```

This example uses an arithmetic overlay to assign the hexadecimal value 2B to all 32 bits of the symbol A. The symbol A has a string value after the overlay because it was previously undefined. (If a symbol is undefined, it has a string value as a result of an arithmetic overlay. If a symbol was previously defined, it retains the same data type after the overlay.) The hexadecimal value 2B corresponds to the ASCII character "+".

Next, the F\$CVUI function extracts the low-order 4 bits from the symbol A; the low-order 4 bits contain the binary representation of the hexadecimal value B. These bits are converted, as a signed value, to an integer. The converted value, 11, is assigned to the symbol X.

F\$DIRECTORY

Returns the current default directory name string. The F\$DIRECTORY function has no arguments, but must be followed by parentheses.

format

F\$DIRECTORY()

arguments

None.

example

```
$ SAVE_DIR = F$DIRECTORY()
$ SET DEFAULT [MALCOLM.TESTFILES]
```

.
.
.

```
$ SET DEFAULT 'SAVE_DIR'
```

This example shows an excerpt from a command procedure that uses the F\$DIRECTORY function to save the current default directory setting. The assignment statement equates the symbol SAVE_DIR to the current directory. Then the SET DEFAULT command establishes a new default

directory. Later, the symbol `SAVE_DIR` is used in the `SET DEFAULT` command that restores the original default directory.

Note that you can use the `F$ENVIRONMENT` function with the `DEFAULT` keyword to return the default disk and directory. You should use the `F$ENVIRONMENT` function rather than the `F$DIRECTORY` function in situations involving more than one disk.

F\$EDIT

Edits the character string based on the edits specified in the edit-list.

format

F\$EDIT(*string*, *edit-list*)

arguments

string

A character string to be edited. Quoted sections of the string are not edited.

edit-list

A character string containing one or more of the following keywords that specify the types of edits to be made to the string. If you use a list of keywords, separate them with commas. Do not abbreviate these keywords.

Edit	Action
COLLAPSE	Removes all spaces or tabs
COMPRESS	Replaces multiple spaces or tabs with a single space
LOWERCASE	Changes all uppercase characters to lowercase
TRIM	Removes leading and trailing spaces or tabs
UNCOMMENT	Removes comments
UPCASE	Changes all lowercase characters to uppercase

DCL-134 Lexical Functions

F\$EDIT

example

```
$ LINE = "   THIS   LINE   CONTAINS A "" QUOTED "" WORD"
$ SHOW SYMBOL LINE
LINE = "   THIS   LINE   CONTAINS A " QUOTED " WORD"
$ NEW_LINE = F$EDIT(LINE, "COMPRESS, TRIM")
$ SHOW SYMBOL NEW_LINE
NEW_LINE = "THIS LINE CONTAINS A " QUOTED " WORD"
```

This example uses the F\$EDIT function to compress and trim a string by replacing multiple blanks with a single blank, and by removing leading and trailing blanks. The string LINE contains quotation marks around the word QUOTED. (To enter quotation marks into a character string, use double quotation marks in the assignment statement.)

Note that the F\$EDIT function does not compress the spaces in the quoted section of the string; therefore, the spaces are retained around the word QUOTED.

F\$ELEMENT

Extracts one element from a string of elements.

format

F\$ELEMENT(*element-number*, *delimiter*, *string*)

arguments

element-number

The number of the element to extract (numbering begins with zero). Specify the element-number argument as an integer expression. If the element-number argument exceeds the number of elements in the string, F\$ELEMENT returns the delimiter.

delimiter

A character used to separate the elements in the string. Specify the delimiter as a character string expression.

string

A string containing a delimited list of elements. Specify the string as a character string expression.

example

```
$ DAY_LIST = "MON/TUE/WED/THU/FRI/SAT/SUN"
$ INQUIRE DAY "ENTER DAY (MON TUE WED THU FRI SAT SUN)"
$ NUM = 0
$ LOOP:
$     LABEL = F$ELEMENT(NUM,"/",DAY_LIST)
$     IF LABEL .EQS. "/" THEN GOTO END
$     IF DAY .EQS. LABEL THEN GOTO 'LABEL'
$     NUM = NUM +1
$     GOTO LOOP
$
$
$ MON:
.
.
.
```

This example sets up a *loop* to test an input value against the elements in a list of values. If the value for DAY matches one of the elements in DAY_LIST, control is passed to the corresponding label. If the value returned by the F\$ELEMENT function matches the delimiter, the value DAY was not present in the DAY_LIST, and control is passed to the label END.

F\$ENVIRONMENT

Returns information about the current DCL command environment.

format

F\$ENVIRONMENT(*item*)

argument

item

A keyword, specified as a character string, that specifies the type of information to be returned. Do not abbreviate these keywords. Specify one of the following keywords:

DCL-136 Lexical Functions
F\$ENVIRONMENT

Item	Data Type	Information Returned
CAPTIVE	String	TRUE if you are logged in to a captive account.
CONTROL	String	Control characters currently enabled with SET CONTROL. Multiple characters are separated by commas; if no control characters are enabled, the null string ("") is returned.
DEFAULT	String	Current default device and directory name. The returned string is the same as SHOW DEFAULT output.
DEPTH	Integer	Current command procedure depth.
INTERACTIVE	String	TRUE if the process is executing interactively.
KEY_STATE	String	Current locked keypad state. See the description of the DEFINE/KEY command for more information on keypad states.
MAX_DEPTH	Integer	Maximum allowable command procedure depth.
MESSAGE	String	Current setting of SET MESSAGE qualifiers. Each qualifier in the string is prefaced by a slash; therefore, the output from F\$ENVIRONMENT("MESSAGE") can be appended to the SET MESSAGE command to form a valid DCL command line.
NOCONTROL	String	Control characters currently disabled with SET NOCONTROL. Multiple characters are separated by commas; if no control characters are disabled, the null string is returned.
ON_CONTROL_Y	String	If issued from a command procedure, returns TRUE if ON_CONTROL_Y is set. ON_CONTROL_Y always returns FALSE at DCL command level.
ON_SEVERITY	String	If issued from a command procedure, returns the severity level at which the action specified with the ON command is performed. ON_SEVERITY returns "NONE" when SET NOON is in effect or at DCL command level.
OUTPUT_RATE	String	Delta time string containing the default output rate, which indicates how often data is written to the batch job log file while the batch job is executing. OUTPUT_RATE returns a null string if used interactively.
PROCEDURE	String	File specification of the current command procedure. PROCEDURE returns a null string if used interactively.

Item	Data Type	Information Returned
PROMPT	String	Current DCL prompt.
PROMPT_CONTROL	String	TRUE if a carriage return and line feed precede the prompt.
PROTECTION	String	Current default file protection.
SYMBOL_SCOPE	String	[NO]LOCAL,[NO]GLOBAL to indicate the current symbol scoping state.
VERIFY_IMAGE	String	TRUE if image verification (SET VERIFY=IMAGE) is in effect.
VERIFY_PROCEDURE	String	TRUE if procedure verification (SET VERIFY=PROCEDURE) is in effect.

example

```
$ SAVE_MESSAGE = F$ENVIRONMENT("MESSAGE")
$ SET MESSAGE/NOFACILITY/NOIDENTIFICATION
```

.
.
.

```
$ SET MESSAGE'SAVE_MESSAGE'
```

This example uses the F\$ENVIRONMENT function to save the current message setting before changing the setting. At the end of the command procedure, the original message setting is restored. The apostrophes surrounding the symbol SAVE_MESSAGE indicate that the value for the symbol should be substituted.

F\$EXTRACT

Extracts the specified characters from the specified string.

format

```
F$EXTRACT(start,length,string)
```

arguments

start

Specifies the offset of the starting character of the string you want to extract. Specify the start argument as an integer expression that is greater than or equal to 0.

DCL-138 Lexical Functions

F\$EXTRACT

length

Specifies the number of characters you want to extract; must be less than or equal to the size of the string. Specify the length as an integer expression that is greater than or equal to 0.

string

Specifies the character string to be edited. Specify the string as a character string expression.

example

```
$ IF F$EXTRACT(12,2,F$TIME()) .GES. "12" THEN GOTO AFTERNOON
$ MORNING:
$ WRITE SYS$OUTPUT "Good morning!"
$ EXIT
$ AFTERNOON:
$ WRITE SYS$OUTPUT "Good afternoon!"
$ EXIT
```

This example shows a procedure that displays a different message, depending on whether the current time is morning or afternoon. It first obtains the current time of day by using the F\$TIME function. The F\$TIME function returns a character string, which is the string argument for the F\$EXTRACT function. The F\$TIME function is automatically evaluated when it is used as an argument, so you do not need to use quotation marks.

Next, the F\$EXTRACT function extracts the hours from the date and time string returned by F\$TIME. The string returned by F\$TIME always contains the hours field beginning at an offset of 12 characters from the start of the string.

The F\$EXTRACT function extracts two characters from the string, beginning at this offset, and compares the string value extracted with the string value 12. If the comparison is true, then the procedure writes "Good afternoon!". Otherwise, it writes "Good morning!".

Note that you can also use the F\$CVTIME function to extract the hour field from a time specification. This method is easier than the one shown in the above example.

F\$FAO

Invokes the \$FAO system service to convert character and numeric input to character strings. (FAO stands for formatted ASCII output.) By specifying formatting instructions, you can use the F\$FAO function to convert integer values to character strings, insert carriage returns and form feeds, insert text, and so on.

format

F\$FAO(*control-string*[,*arg1, arg2...arg15*])

arguments

control-string

Specifies the fixed text of the output string, consisting of text and any number of FAO directives. The control string may be any length. Specify the control string as a character string expression.

Table DCL-2 lists the FAO directives you can specify in a control string.

arg1, arg2...arg15

Specifies the arguments required by the FAO directives used in the control string. Specify the arguments *arg1, arg2...arg15* as integer or character string expressions. Table DCL-2 lists the argument types required by each FAO directive.

If you specify an argument whose type (integer or string) does not match that of the corresponding directive, unpredictable results are returned. You can use the F\$INTEGER and F\$STRING lexical functions to convert arguments to the proper type.

description

Specify an FAO directive using any one of the following formats:

Format	Function
!DD	One directive
!n(DD)	A directive repeated a specified number of times
!lengthDD	A directive that places its output in a field of a specified length
!n(lengthDD)	A directive that is repeated a specified number of times and generates output fields of a specified length

The exclamation point (!) indicates that the following character or characters are to be interpreted as an FAO directive. DD represents a one- or two-character uppercase code indicating the action that F\$FAO is to perform. When specifying repeat counts, n is a decimal value specifying

DCL-140 Lexical Functions
F\$FAO

the number of times the directive is to be repeated. The length value is a decimal value that instructs F\$FAO to generate an output field of “length” characters.

The FAO directives are grouped in the following categories:

- Character string insertion
- Zero-filled numeric conversion
- Blank-filled numeric conversion
- Special formatting
- Parameter interpretation

Table DCL-2 summarizes the FAO directives and shows the required argument types.

Table DCL-2: Summary of FAO Directives

Directive	Argument Type	Description
Character string insertion:		
!AS	String	Inserts a character string as is
Zero-filled numeric conversion:		
!OB	Integer	Converts a byte to octal notation
!OW	Integer	Converts a word to octal notation
!OL	Integer	Converts a longword to octal notation
!XB	Integer	Converts a byte to hexadecimal notation
!XW	Integer	Converts a word to hexadecimal notation
!XL	Integer	Converts a longword to hexadecimal notation
!ZB	Integer	Converts a byte to decimal notation
!ZW	Integer	Converts a word to decimal notation
!ZL	Integer	Converts a longword to decimal notation
Blank-filled numeric conversion:		
!UB	Integer	Converts a byte to decimal notation without adjusting for negative numbers
!UW	Integer	Converts a word to decimal notation without adjusting for negative numbers
!UL	Integer	Converts a longword to decimal notation without adjusting for negative numbers

(continued on next page)

Table DCL-2 (Cont.): Summary of FAO Directives

Directive	Argument Type	Description
!SB	Integer	Converts a byte to decimal notation with negative numbers converted properly
!SW	Integer	Converts a word to decimal notation with negative numbers converted properly
!SL	Integer	Converts a longword to decimal notation with negative numbers converted properly
Special formatting:		
!/	None	Inserts a carriage return and a line feed
!_	None	Inserts a tab
!^	None	Inserts a form feed
!!	None	Inserts an exclamation mark
!%I	Integer	Converts a longword integer to a named UIC in the format [group-identifier,member-identifier]
!%S	None	Inserts an "s" if the most recently converted number is not 1 (Not recommended for use with multilingual products.)
!%U	Integer	Converts a longword integer to a numeric UIC in the format [g,m], where <i>g</i> is the group number and <i>m</i> is the member number The directive inserts the brackets and the comma.
!n<...!>	None	Left-justifies and blank-fills all data represented by the instructions ... in fields <i>n</i> characters wide
!n*c	None	Repeats the character represented by <i>c</i> for <i>n</i> times
!n%C	String	Inserts a character string when the most recently evaluated argument has the value <i>n</i> (Recommended for use with multilingual products.)
!%E	String	Inserts a character string when the value of the most recently evaluated argument does not match any preceding !n%C directives (Recommended for use with multilingual products.)
!%F	None	Marks the end of a plurals statement

(continued on next page)

Table DCL-2 (Cont.): Summary of FAO Directives

Directive	Argument Type	Description
!%T	Integer equal to 0	Inserts the current time
!%D	Integer equal to 0	Inserts the current date/time
Argument interpretation:		
!-	None	Reuses the last argument
!+	None	Skips the next argument

Output Strings from Character String Insertion

The !AS directive inserts a character string (specified as an argument for the directive) into the control string. The field length of the character string when it is inserted into the control string defaults to the length of the character string. If the default length is shorter than an explicitly stated field length, the string is left-justified and blank-filled. If the default length is longer than an explicitly stated field length, the string is truncated on the right.

Output Strings from Zero-Filled Numeric Conversion

Directives for zero-filled numeric conversion convert an integer (specified as an argument for the directive) to decimal, octal, or hexadecimal notation. The ASCII representation of the integer is inserted into the control string. Default output field lengths for the converted argument are determined as follows.

Directives that convert arguments to octal notation return 3 digits for byte conversion, 6 digits for word conversion, and 11 digits for longword conversion. Numbers are right-justified and zero-filled on the left. Explicit-length fields longer than the default are blank-filled on the left. Explicit-length fields shorter than the default are truncated on the left.

Directives that convert arguments to hexadecimal notation return 2 digits for byte conversion, 4 digits for word conversion, and 8 digits for longword conversion. Numbers are right-justified and zero-filled on the left. Explicit-length fields longer than the default are blank-filled on the left. Explicit-length fields shorter than the default are truncated on the left.

Directives that convert arguments to decimal notation return the required number of characters for the decimal number. Explicit-length fields longer than the default are zero-filled on the left. If an explicit-length field is shorter than the number of characters required for the decimal number, the output field is completely filled with asterisks (*).

For byte conversion, only the low-order 8 bits of the binary representation of the argument are used. For word conversion, only the low-order 16 bits of the binary representation of the argument are used. For longword conversion, the entire 32-bit binary representation of the argument is used.

Output Strings from Blank-Filled Numeric Conversion

Directives for blank-filled numeric conversion convert an integer (specified as an argument for the directive) to decimal notation. These directives can convert the integer as a signed or unsigned number. The ASCII representation of the integer is inserted into the control string.

Output field lengths for the converted argument default to the required number of characters. Values shorter than explicit-length fields are right-justified and blank-filled; values longer than explicit-length fields cause the field to be filled with asterisks.

For byte conversion, only the low-order 8 bits of the binary representation of the argument are used. For word conversion, only the low-order 16 bits of the binary representation of the argument are used. For longword conversion, the entire 32-bit binary representation of the argument is used.

Output Strings from Special Formatting Directives

The !*n*%C and !%E directives insert an ASCII string (based on the value of the most recently evaluated argument) into the output string. These directives are useful for inserting irregular plural nouns and verbs.

If the most recently evaluated argument equals *n*, the text between one directive and the next is inserted into the output string. If the most recently evaluated argument does not equal *n*, the next !*n*%C directive is processed.

If *n* must be a negative number, you must specify it as an argument and use the number sign (#).

You can specify the !*n*%C and !%E directives with repeat counts. If you specify repeat counts, the text between one directive and the next is copied to the output string the specified number of times.

The %F directive marks the end of a plurals statement.

example

```
$ COUNT = 57
$ REPORT = F$FAO("NUMBER OF FORMS = !SL",COUNT)
$ SHOW SYMBOL REPORT
$ REPORT = "NUMBER OF FORMS = 57"
```

In this command procedure, the FAO directive !SL is used in a control string to convert the number equated to the symbol COUNT to a character string. The converted string is inserted into the control string.

Note that COUNT is assigned an integer value of 57. The F\$FAO function returns the ASCII string, "NUMBER OF FORMS = 57", and assigns the string to the symbol REPORT.

F\$FILE_ATTRIBUTES

Returns attribute information for a specified file.

format

F\$FILE_ATTRIBUTES(*file-spec,item*)

arguments

file-spec

Specifies the name of the file about which you are requesting information. You must specify the file name as a character string expression.

You may specify only one file name. Wildcard characters are not allowed.

item

Indicates which attribute of the file is to be returned. The item argument must be specified as a character string expression, and can be any one of the VMS RMS field names listed in Table DCL-3.

Table DCL-3: F\$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
AI	String	"TRUE" if AI journaling is enabled; returns "TRUE" or "FALSE"
ALQ	Integer	Allocation quantity
BDT	String	Backup date/time

(continued on next page)

Table DCL-3 (Cont.): F\$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
BI	String	"TRUE" if BI journaling is enabled; returns "TRUE" or "FALSE"
BKS	Integer	Bucket size
BLS	Integer	Block size
CBT	String	"TRUE" if contiguous-best-try; returns "TRUE" or "FALSE"
CDT	String	Creation date/time
CTG	String	"TRUE" if contiguous; returns "TRUE" or "FALSE"
DEQ	Integer	Default extension quantity
DID	String	Directory ID string
DVI	String	Device name string
EDT	String	Expiration date/time
EOF	Integer	Number of blocks used
ERASE	String	"TRUE" if a file's contents are erased before a file is deleted; returns "TRUE" or "FALSE"
FFB	Integer	First free byte
FID	String	File ID string
FSZ	Integer	Fixed control area size
GBC	Integer	Global buffer count
GRP	Integer	Owner group number
JOURNAL_FILE	String	"TRUE" if the file is a journal file; returns "TRUE" or "FALSE"
KNOWN	String	Known file; returns "TRUE" or "FALSE" to indicate whether file is installed with the Install Utility
LOCKED	String	"TRUE" if a file is deaccessed-locked; returns "TRUE" or "FALSE"
MBM	Integer	Owner member number
MRN	Integer	Maximum record number
MRS	Integer	Maximum record size
NOA	Integer	Number of areas
NOK	Integer	Number of keys
ORG	String	File organization; returns "SEQ", "REL", "IDX"
PRO	String	File protection string
PVN	Integer	Prolog version number

(continued on next page)

DCL-146 Lexical Functions

F\$FILE_ATTRIBUTES

Table DCL-3 (Cont.): F\$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
RAT	String	Record attributes; returns "CR", "PRN", "FTN", ""
RCK	String	TRUE if read check; returns "TRUE", "FALSE"
RDT	String	Revision date/time
RFM	String	Record format string; returns the values "VAR", "FIX", "VFC", "UDF", "STM", "STMLF", "STMCR"
RU	String	"TRUE" if RU journaling is enabled; returns "TRUE" or "FALSE"
RVN	Integer	Revision number
STORED_SEMANTICS	String	ASCII string that represents stored semantics
UIC	String	Owner UIC string
WCK	String	"TRUE" if write check; returns "TRUE", "FALSE"

example

```
$ FILE_ORG = F$FILE_ATTRIBUTES("QUEST.DAT", "ORG")
$ SHOW SYMBOL FILE_ORG
FILE_ORG = "SEQ"
```

This example uses the F\$FILE_ATTRIBUTES function to assign the value of the file organization type to the symbol FILE_ORG. The F\$FILE_ATTRIBUTES function returns the character string "SEQ" to show that QUEST.DAT is a sequential file.

The QUEST.DAT and ORG arguments for the F\$FILE_ATTRIBUTES function are string literals and must be enclosed in quotation marks when used in expressions.

F\$GETDVI

Invokes the \$GETDVI system service to return a specified item of information for a specified device.

format

F\$GETDVI(*device-name,item*)

arguments

device-name

Specifies a physical device name or a logical name equated to a physical device name. Specify the device name as a character string expression.

item

Specifies the type of device information to be returned. The item argument must be specified as a character string expression and may be any one of the items listed in Table DCL-4.

description

The F\$GETDVI function returns information on all items that can be specified with the \$GETDVI system service. In addition to the items that the \$GETDVI system service allows, the F\$GETDVI function allows you to specify the item EXISTS.

Table DCL-4 lists the items you can specify with the F\$GETDVI function, the type of information returned, and the data types of the return values. Table DCL-5 lists the values returned by the DEVCLASS item. Table DCL-6 lists the values returned by the DEVTYPE item.

Table DCL-4: F\$GETDVI Items

Item	Return Type	Information Returned
ACPPID	String	ACP process ID.
ACPTYPE	String	ACP type code, as one of the following strings: "F11V1", "F11V2", "JNL", "MTA", "NET", "REM", or "ILLEGAL" if the device is not mounted or is mounted FOREIGN.
ALL	String	"TRUE" or "FALSE" to indicate whether the device is allocated.
ALLDEVNAM	String	Allocation class device name.
ALLOCLASS	Longword integer between 0 and 255	Allocation class of the host.
ALT_HOST_AVAIL	String	"TRUE" or "FALSE" to indicate whether the host serving the alternate path is available.
ALT_HOST_NAME	String	Name of the host serving the alternate path.
ALT_HOST_TYPE	String	Hardware type of the host serving the alternate path.

(continued on next page)

**DCL-148 Lexical Functions
F\$GETDVI**

Table DCL-4 (Cont.): F\$GETDVI Items

Item	Return Type	Information Returned
AVL	String	"TRUE" or "FALSE" to indicate whether the device is available for use.
CCL	String	"TRUE" or "FALSE" to indicate whether the device is a carriage control device.
CLUSTER	Integer	Volume cluster size.
CONCEALED	String	"TRUE" or "FALSE" to indicate whether the logical device name translates to a concealed device.
CYLINDERS	Integer	Number of cylinders on the volume (disk).
DEVBUFSIZ	Integer	Device buffer size.
DEVCHAR	Integer	Device characteristics.
DEVCHAR2	Integer	Additional device characteristics.
DEVCLASS	Integer	Device class. See Table DCL-5 for a list of the values returned.
DEVDEPEND	Integer	Device-dependent information.
DEVDEPEND2	Integer	Additional device-dependent information.
DEVLOCKNAM	String	A unique lock name for the device.
DEVNAM	String	Device name.
DEVSTS	Integer	Device-dependent status information.
DEVTYPE	Integer	Device type. See Table DCL-6 for a list of the values returned.
DIR	String	"TRUE" or "FALSE" to indicate whether the device is directory structured.
DMT	String	"TRUE" or "FALSE" to indicate whether the device is marked for dismount.
DUA	String	"TRUE" or "FALSE" to indicate whether the device is a generic device.
ELG	String	"TRUE" or "FALSE" to indicate whether the device has error logging enabled.
ERRCNT	Integer	Error count.
EXISTS	String	"TRUE" or "FALSE" to indicate whether the device exists on the system.
FOD	String	"TRUE" or "FALSE" to indicate whether the device is a files-oriented device.

(continued on next page)

Table DCL-4 (Cont.): F\$GETDVI Items

Item	Return Type	Information Returned
FOR	String	"TRUE" or "FALSE" to indicate whether the device is mounted foreign.
FREEBLOCKS	Integer	Number of free blocks on the volume (disk).
FULLDEVNAM	String	Fully qualified device name.
GEN	String	"TRUE" or "FALSE" to indicate whether the device is a generic device.
HOST_AVAIL	String	"TRUE" or "FALSE" to indicate whether the host serving the primary path is available.
HOST_COUNT	Integer	Number of hosts that make the device available to other nodes in the VAXcluster.
HOST_NAME	String	Name of the host serving the primary path.
HOST_TYPE	String	Hardware type of the host serving the primary path.
IDV	String	"TRUE" or "FALSE" to indicate whether the device is capable of providing input.
LOCKID	Integer	Clusterwide lock identification.
LOGVOLNAM	String	Logical volume name.
MAXBLOCK	Integer	Number of logical blocks on the volume.
MAXFILES	Integer	Maximum number of files on the volume. This item code is applicable only to disks.
MBX	String	"TRUE" or "FALSE" to indicate whether the device is a mailbox.
MEDIA_ID	String	Nondecoded media ID.
MEDIA_NAME	String	Either the name of the disk or the tape type.
MEDIA_TYPE	String	Device name prefix.
MNT	String	"TRUE" or "FALSE" to indicate whether the device is mounted.
MOUNTCNT	Integer	Mount count.
NET	String	"TRUE" or "FALSE" to indicate whether the device is a network device.
NEXTDEVNAM	String	Device name of the next volume in a volume set. This item applies only to disks.
ODV	String	"TRUE" or "FALSE" to indicate whether the device is capable of providing output.
OPCNT	Integer	Operation count.

(continued on next page)

DCL-150 Lexical Functions
F\$GETDVI

Table DCL-4 (Cont.): F\$GETDVI Items

Item	Return Type	Information Returned
OPR	String	"TRUE" or "FALSE" to indicate whether the device is an operator.
OWNUIC	String	UIC of the device owner.
PID	String	Process identification of the device owner.
RCK	String	"TRUE" or "FALSE" to indicate whether the device has read checking enabled.
RCT	String	"TRUE" or "FALSE" to indicate whether the disk contains RCT.
REC	String	"TRUE" or "FALSE" to indicate whether the device is record oriented.
RECSIZ	Integer	Blocked record size.
REFCNT	Integer	Reference count of processes using the device.
REMOTE_DEVICE	String	"TRUE" or "FALSE" to indicate whether the device is a remote device.
RND	String	"TRUE" or "FALSE" to indicate whether the device allows random access.
ROOTDEVNAM	String	Device name of the root volume in a volume set. This item applies only to disks.
RTM	String	"TRUE" or "FALSE" to indicate whether the device is real-time.
SDI	String	"TRUE" or "FALSE" to indicate whether the device is single-directory structured.
SECTORS	Integer	Number of sectors per track. This item applies only to disks.
SERIALNUM	Integer	Volume serial number. This item applies only to disks.
SERVED_DEVICE	String	"TRUE" or "FALSE" to indicate whether the device is a served device.
SHR	String	"TRUE" or "FALSE" to indicate whether the device is shareable.
SPL	String	"TRUE" or "FALSE" to indicate whether the device is being spooled.
SPLDEVNAM	String	Name of the device being spooled.
SQD	String	"TRUE" or "FALSE" to indicate whether the device is sequential block-oriented (that is, magnetic tape).

(continued on next page)

Table DCL-4 (Cont.): F\$GETDVI Items

Item	Return Type	Information Returned
STS	Integer	Status information.
SWL	String	"TRUE" or "FALSE" to indicate whether the device is software write-locked.
TRACKS	Integer	Number of tracks per cylinder. This item applies only to disks.
TRANSCNT	Integer	Volume transaction count.
TRM	String	"TRUE" or "FALSE" to indicate whether the device is a terminal.
TT_ACCPORNAM	String	The terminal server name and port name.
TT_ALTYPEAHD	String	"TRUE" or "FALSE" to indicate whether the terminal has an alternate type-ahead buffer (terminals only).
TT_ANSICRT	String	"TRUE" or "FALSE" to indicate whether the terminal is an ANSI CRT terminal (terminals only).
TT_APP_KEYPAD	String	"TRUE" or "FALSE" to indicate whether the keypad is in applications mode (terminals only).
TT_AUTOBAUD	String	"TRUE" or "FALSE" to indicate whether the terminal has automatic baud rate detection (terminals only).
TT_AVO	String	"TRUE" or "FALSE" to indicate whether the terminal has a VT100-family terminal display (terminals only).
TT_BLOCK	String	"TRUE" or "FALSE" to indicate whether the terminal has block mode capability (terminals only).
TT_BRDCSTMBX	String	"TRUE" or "FALSE" to indicate whether the terminal uses mailbox broadcast messages (terminals only).
TT_CRFILL	String	"TRUE" or "FALSE" to indicate whether the terminal requires fill after RET (terminals only).
TT_DECCRT	String	"TRUE" or "FALSE" to indicate whether the terminal is a Digital CRT terminal (terminals only).
TT_DECCRT2	String	"TRUE" or "FALSE" to indicate whether the terminal is a Digital CRT2 terminal (terminals only).
TT_DECCRT3	String	"TRUE" or "FALSE" to indicate whether the terminal is a Digital CRT3 terminal (terminals only).
TT_DIALUP	String	"TRUE" or "FALSE" to indicate whether the terminal is connected to dialup (terminals only).
TT_DISCONNECT	String	"TRUE" or "FALSE" to indicate whether the terminal can be disconnected (terminals only).

(continued on next page)

DCL-152 Lexical Functions
F\$GETDVI

Table DCL-4 (Cont.): F\$GETDVI Items

Item	Return Type	Information Returned
TT_DMA	String	"TRUE" or "FALSE" to indicate whether the terminal has DMA mode (terminals only).
TT_DRCS	String	"TRUE" or "FALSE" to indicate whether the terminal supports loadable character fonts (terminals only).
TT_EDIT	String	"TRUE" or "FALSE" to indicate whether the edit characteristic is set.
TT_EDITING	String	"TRUE" or "FALSE" to indicate whether advanced editing is enabled (terminals only).
TT_EIGHTBIT	String	"TRUE" or "FALSE" to indicate whether the terminal uses the 8-bit ASCII character set (terminals only).
TT_ESCAPE	String	"TRUE" or "FALSE" to indicate whether the terminal generates escape sequences (terminals only).
TT_FALLBACK	String	"TRUE" or "FALSE" to indicate whether the terminal uses the multinational fallback option (terminals only).
TT_HALFDUP	String	"TRUE" or "FALSE" to indicate whether the terminal is in half-duplex mode (terminals only).
TT_HANGUP	String	"TRUE" or "FALSE" to indicate whether the hangup characteristic is set (terminals only).
TT_HOSTSYNC	String	"TRUE" or "FALSE" to indicate whether the terminal has host/terminal communication (terminals only).
TT_INSERT	String	"TRUE" or "FALSE" to indicate whether insert-mode is the default line editing mode (terminals only).
TT_LFFILL	String	"TRUE" or "FALSE" to indicate whether the terminal requires fill after LF (terminals only).
TT_LOCALECHO	String	"TRUE" or "FALSE" to indicate whether the local echo characteristic is set (terminals only).
TT_LOWER	String	"TRUE" or "FALSE" to indicate whether the terminal has the lowercase characters set.
TT_MBXDSABL	String	"TRUE" or "FALSE" to indicate whether mailboxes associated with the terminal will receive unsolicited input notification or input notification (terminals only).
TT_MECHFORM	String	"TRUE" or "FALSE" to indicate whether the terminal has mechanical form feed (terminals only).
TT_MECHTAB	String	"TRUE" or "FALSE" to indicate whether the terminal has mechanical tabs and is capable of tab expansion (terminals only).

(continued on next page)

Table DCL-4 (Cont.): F\$GETDVI Items

Item	Return Type	Information Returned
TT_MODEM	String	"TRUE" or "FALSE" to indicate whether the terminal is connected to a modem (terminals only).
TT_MODHANGUP	String	"TRUE" or "FALSE" to indicate whether the modify hang-up characteristic is set (terminals only).
TT_NOBRDCST	String	"TRUE" or "FALSE" to indicate whether the terminal will receive broadcast messages (terminals only).
TT_NOECHO	String	"TRUE" or "FALSE" to indicate whether the input characters are echoed.
TT_NOTYPEAHD	String	"TRUE" or "FALSE" to indicate whether data must be solicited by a read operation.
TT_OPER	String	"TRUE" or "FALSE" to indicate whether the terminal is an operator terminal (terminals only).
TT_PAGE	Integer	Terminal page length (terminals only).
TT_PASTHRU	String	"TRUE" or "FALSE" to indicate whether there is passall with flow control (terminals only).
TT_PHYDEVNAM	String	Physical device name associated with a channel number or virtual terminal.
TT_PRINTER	String	"TRUE" or "FALSE" to indicate whether there is a printer port available (terminals only).
TT_READSYNC	String	"TRUE" or "FALSE" to indicate whether the terminal has read synchronization (terminals only).
TT_REGIS	String	"TRUE" or "FALSE" to indicate whether the terminal has ReGIS graphics (terminals only).
TT_REMOTE	String	"TRUE" or "FALSE" to indicate whether the terminal has established modem control (terminals only).
TT_SCOPE	String	"TRUE" or "FALSE" to indicate whether the terminal is a video screen display (terminals only).
TT_SECURE	String	"TRUE" or "FALSE" to indicate whether the terminal can recognize the secure server (terminals only).
TT_SETSPEED	String	"TRUE" or "FALSE" to indicate whether you can set the speed on the terminal line (terminals only).
TT_SIXEL	String	"TRUE" or "FALSE" to indicate whether the sixel is supported (terminals only).
TT_TTSYNC	String	"TRUE" or "FALSE" to indicate whether there is terminal/host synchronization (terminals only).

(continued on next page)

DCL-154 Lexical Functions
F\$GETDVI

Table DCL-4 (Cont.): F\$GETDVI Items

Item	Return Type	Information Returned
TT_SYSPWD	String	"TRUE" or "FALSE" to indicate whether the system password is enabled for a particular terminal.
TT_WRAP	String	"TRUE" or "FALSE" to indicate whether a new line should be inserted if the cursor moves beyond the right margin.
UNIT	Integer	The unit number.
VOLCOUNT	Integer	The count of volumes in a volume set. This item applies only to disks.
VOLNAM	String	The volume name.
VOLNUMBER	Integer	Number of the current volume in a volume set. This item applies only to disks.
VOLSETMEM	String	"TRUE" or "FALSE" to indicate whether the device is a volume set (disks only).
VPRO	String	The volume protection mask.
WCK	String	"TRUE" or "FALSE" to indicate whether the device has write checking enabled.

Table DCL-5: Values Returned by the DEVCLASS Item

Device Class	Value	Symbolic Name
Disk device	1	DC\$_DISK
Tape device	2	DC\$_TAPE
Synchronous	32	DC\$_SCOM
Communications device		
Card reader	65	DC\$_CARD
Terminal	66	DC\$_TERM
Line printer	67	DC\$_LP
Real-time	96	DC\$_REALTIME
Bus	128	DC\$_BUS
Mailbox	160	DC\$_MAILBOX
Journal	161	DC\$_JOURNAL
Miscellaneous device	200	DC\$_MISC

Table DCL-6: Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
Device Class: DC\$_DISK			
RK06	1	RD54	32
RK07	2	CRX50	33
RP04	3	RX50	33
RP05	4	RRD50	34
RP06	5	GENERIC_DU	35
RMO3	6	RX33	36
RP07	7	RX18	37
RP07HT	8	RA70	38
RL01	9	RA90	39
RL02	10	RD32	40
RX02	11	DISK9	41
RX04	12	RX35	42
RM80	13	RF30	43
TU58	14	RF70	44
RM05	15	RD33	45
RX01	16	ESE20	46
ML11	17	TU56	47
RB02	18	RZ22	48
RB80	19	RZ23	49
RA80	20	RZ24	50
RA81	21	RZ55	51
RA60	22	RRD40	52
RZ01	23	GENERIC_DK	54
RZF01	24	RX23	55
RD51	25	FD1	129
RX50	26	FD2	130
RC25	23	FD3	131

(continued on next page)

DCL-156 Lexical Functions
F\$GETDVI

Table DCL-6 (Cont.): Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
Device Class: DC\$_DISK			
RD52	27	FD4	132
RD53	28	FD5	133
RD26	29	FD6	134
RA82	30	FD7	135
RD31	31	FD8	136
Device Class: DC\$_TAPE			
TE16	1	MW_TSV05	14
TU45	2	TK70	15
TU77	3	RV20	16
TS11	4	RV80	16
TU78	5	TK60	17
TA78	6	GENERIC_TU	18
TU80	7	TA79	19
TU81	8	TAPE9	20
TA81	9	TA90	21
TK50	10	TF30	22
MR_TU70	11	TF70	23
MR_TU72	12	RV60	24
MW_TSU05	13		
Device Class: DC\$_SCOM			
DMC11	1	YQ_3271	18
DMR11	2	YR_DDCMP	19
XK_3271	3	YS_SDLG	20
XJ_2780	4	UK_KTC32	21
NW_X25	5	DEQNA	22
NV_X29	6	DMV11	23

(continued on next page)

Table DCL-6 (Cont.): Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
Device Class: DC\$_SCOM			
SB_ISB11	7	ES_LANCE	24
MX_MUX200	8	DELUA	25
DMP11	9	NQ_3271	26
DMF32	10	DMB32	27
XV_3271	11	YI_KMS11K	28
CI	12	ET_DEBNT	29
NI	13	ET_DEBNA	29
UNA11	14	SJ_DSV11	30
DEUNA	14	SL_DSB32	31
YN_X25	15	ZS_DST32	32
YO_X25	16	XQ_DELQA	33
YP_ADCCP	17		
Device Class: DC\$_CARD			
CR11	1		
Device Class: DC\$_TERM			
TTYUNKN	0	LA24	37
VT05	1	LA100	37
VK100	2	LQP02	38
VT173	3	LA210	40
VT5X	64	LN03	41
TEK401X	10	LN01K	42
FT1	16	LA80	43
FT2	17	VT52	64
FT3	18	VT55	65
FT4	19	VT100	96
FT5	20	VT101	97
FT6	21	VT102	98

(continued on next page)

DCL-158 Lexical Functions
F\$GETDVI

Table DCL-6 (Cont.): Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
Device Class: DC\$_TERM			
FT7	22	VT105	99
FT8	23	VT125	100
LAX	32	VT131	101
LA36	32	VT132	102
LA120	33	VT200_Series	110
LA34	34	Pro_Series	111
LA38	35	VT300_Series	112
LA12	36		
Device Class: DC\$_LP			
LP11	1	LC_DMF32	4
LA11	2	LI_DMB32	5
LA180	3	PRTR9	6
Device Class: DC\$_REALTIME			
LPA11	1	XP_PCL11B	9
DR780	2	IX_IEX11	10
DR750	3	FP_FEPCM	11
DR11W	4	TK_FCM	12
PCL11R	5	XI_DR11C	13
PCL11T	6	XA_DRV11WA	14
DR11C	7	DRB32	15
XI_DR11C	8	HX_DRQ3B	16

(continued on next page)

Table DCL-6 (Cont.): Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
Device Class: DC\$_BUS			
CI780	1	BCI750	15
CI750	2	BCA	16
UQPORT	3	RQDX3	17
UDA50	3	NISCA	18
UDA50A	4	AIO	19
LESI	5	AIE	20
TU81P	6	DEBNT	20
RDRX	7	BSA	21
TK50P	8	KSB50	21
RUX50P	9	TK70P	22
RC26P	10	RV20P	23
QDA50	11	RV80P	23
KDA50	11	TK60P	24
BDA50	12	SII	25
KDB50	12	KFSQSA	26
RRD50P	13	SHAC	27
QDA25	14	CIXCA	28
KDA25	14	CIXCB	29

Device Class: DC\$_MAILBOX

MBX	1
SHRMBX	2
NULL	3

Device Class: DC\$_JOURNAL

UNKNJNL	0	RUJNL	1
BIJNL	2	AIJNL	3
ATJNL	4	CLJNL	5

(continued on next page)

DCL-160 Lexical Functions

F\$GETDVI

Table DCL-6 (Cont.): Values Returned by the DEVTYPE Item

Device Type	Value	Device Type	Value
Device Class: DC\$_MISC			
DN11	1	SFUN9	3
PV	2	USER9	4

example

```
$ ERR = F$GETDVI("_DQA0", "ERRCNT")
$ SHOW SYMBOL ERR
ERR = 0 Hex = 00000000 Octal = 000000
```

This example shows how to use the F\$GETDVI function to return an error count for the device DQA0. You must place quotation marks around the device name DQA0 and the item ERRCNT because they are string literals.

F\$GETJPI

Invokes the \$GETJPI system service to return accounting, status, and identification information on the specified process.

Requires GROUP privilege to obtain information on other processes in the same group. Requires WORLD privilege to obtain information on any other processes in the system.

format

F\$GETJPI(*pid,item*)

arguments

pid

Specifies the identification number of the process for which information is being reported. Specify the pid argument as a character string expression. You can omit the leading zeros. If you specify a null string (""), the current process identification number is used.

item

Indicates the type of process information to be returned. *Item* must be specified as a character string expression and may be any one of the items listed in Table DCL-7.

description

The F\$GETJPI function returns information on all items that can be specified with the \$GETJPI system service.

Table DCL-7: F\$GETJPI Items

Item	Return Type	Information Returned
ACCOUNT	String	Account name string (8 characters filled with trailing blanks)
APTCNT	Integer	Active page table count
ASTACT	Integer	Access modes with active ASTs
ASTCNT	Integer	Remaining AST quota
ASTEN	Integer	Access modes with ASTs enabled
ASTLM	Integer	AST limit quota
AUTHPRI	Integer	Maximum priority that a process without the ALTPRI privilege can achieve with the \$SETPRI system service
AUTHPRIV	String	Privileges that a process is authorized to enable
BIOCNT	Integer	Remaining buffered I/O quota
BIOLM	Integer	Buffered I/O limit quota
BUFIO	Integer	Count of process buffered I/O operations
BYTCNT	Integer	Remaining buffered I/O byte count quota
BYTLM	Integer	Buffered I/O byte count limit quota
CLINAME	String	Current command language interpreter; always returns "DCL"
CPULIM	Integer	Limit on process CPU time
CPUTIM	Integer	CPU time used in hundredths of a second
CURPRIV	String	Current process privileges
CREPRC_FLAGS	Integer	Flags specified by the stsf argument in the \$CREPRC call that created the process
DFPFC	Integer	Default page fault cluster size
DFWSCNT	Integer	Default working set size
DIOCNT	Integer	Remaining direct I/O quota
DIOLM	Integer	Direct I/O limit quota
DIRIO	Integer	Count of direct I/O operations for the process
EFCS	Integer	Local event flags 0 through 31

(continued on next page)

DCL-162 Lexical Functions
F\$GETJPI

Table DCL-7 (Cont.): F\$GETJPI Items

Item	Return Type	Information Returned
EFCU	Integer	Local event flags 32 through 63
EFWM	Integer	Event flag wait mask
ENQCNT	Integer	Lock request quota remaining
ENQLM	Integer	Lock request quota limit
EXCVEC	Integer	Address of a list of exception vectors
FILCNT	Integer	Remaining open file quota
FILLM	Integer	Open file quota
FINALEXC	Integer	Address of a list of final exception vectors
FREPOVA	Integer	First free page at end of program region (PO space). Irrelevant if no image is running.
FREP1VA	Integer	First free page at end of control region (P1 space)
FREPTECNT	Integer	Number of pages available for virtual memory expansion
GPGCNT	Integer	Global page count in working set
GRP	Integer	Group number of the UIC
IMAGECOUNT	Integer	Number of images that have been run down for the process
IMAGNAME	String	File name of the current image
IMAGPRIV	String	Privileges with which the current image was installed
JOBPRCNT	Integer	Number of subprocesses owned by the process
JOBTYPE	Integer	Execution mode of the process at the root of the job tree
LOGINTIM	String	Process creation time
MASTER_PID	String	Returns the process identification of the process at the top of the current job's process tree
MAXDETACH	Integer	Maximum number of detached processes allowed the user who owns the process
MAXJOBS	Integer	Maximum number of active processes allowed for the user who owns the process
MEM	Integer	Member number of the UIC
MODE	String	Current process mode ("BATCH", "INTERACTIVE", "NETWORK", or "OTHER")
MSGMASK	Integer	Default message mask
OWNER	String	Process identification number of process owner

(continued on next page)

Table DCL-7 (Cont.): F\$GETJPI Items

Item	Return Type	Information Returned
PAGEFLTS	Integer	Count of page faults
PAGFILCNT	Integer	Remaining paging file quota
PAGFILLOC	Integer	Location of the paging file
PGFLQUOTA	Integer	Paging file quota (maximum virtual page count)
PHDFLAGS	Integer	Flags word
PID	String	Process identification number
PPGCNT	Integer	Process page count
PRCCNT	Integer	Count of subprocesses
PRCLM	Integer	Subprocess quota
PRCNAM	String	Process name
PRI	Integer	Process's current priority
PRIB	Integer	Process's base priority
PROC_INDEX	Integer	Process's index number
PROCPRIV	Integer	Process's default privileges
SHRFILLM	Integer	Maximum number of open shared files allowed for the job to which the process belongs
SITESPEC	Integer	Per-process site-specific longword
STATE	String	Process state
STS	Integer	Process status flags
SWPFILLOC	Integer	Location of the swap file
TABLENAME	String	File specification of the process's current command language interpreter (CLI) table
TERMINAL	String	Login terminal name for interactive users (1-7 characters)
TMBU	Integer	Termination mailbox unit number
TQCNT	Integer	Remaining timer queue entry quota
TQLM	Integer	Timer queue entry quota
UAF_FLAGS	Integer	User Authorization File (UAF) flags from the UAF record of the user who owns the process
UIC	String	Process's UIC
USERNAME	String	User name string (12 characters filled with trailing blanks)

(continued on next page)

Table DCL-7 (Cont.): F\$GETJPI Items

Item	Return Type	Information Returned
VIRTPEAK	Integer	Peak virtual address size
VOLUMES	Integer	Count of currently mounted volumes
WSAUTH	Integer	Maximum authorized working set size
WSAUTHEXT	Integer	Maximum authorized working set extent
WSEXTENT	Integer	Current working set extent
WSPEAK	Integer	Working set peak
WSQUOTA	Integer	Working set size quota
WSSIZE	Integer	Process's current working set size

example

```
$ NAME = F$GETJPI("3B0018", "USERNAME")  
$ SHOW SYMBOL NAME  
NAME = "JANE"
```

This example shows how to use the F\$GETJPI function to return the username for the process number 3B0018. The username is assigned to the symbol NAME.

F\$GETQUI

Invokes the \$GETQUI system service to return information about queues, batch and print jobs currently in those queues, form definitions, and characteristic definitions kept in the system job queue file.

Requires READ access to the job or SYSPRV or OPER privilege to obtain job and file information.

format

F\$GETQUI(*function*, [*item*], [*object-id*], [*flags*])

arguments

function

Specifies the action that the F\$GETQUI lexical function is to perform. F\$GETQUI supports all functions that can be specified with the \$GETQUI system service.

Function	Description
CANCEL_OPERATION	Terminates any wildcard operation that may have been initiated by a previous call to F\$GETQUI.
DISPLAY_CHARACTERISTIC	Returns information about a specific characteristic definition or the next characteristic definition in a wildcard operation.
DISPLAY_ENTRY	Returns information about a specific job entry or the next job entry that matches the selection criteria in a wildcard operation. The DISPLAY_ENTRY function code is similar to the DISPLAY_JOB function code in that both return job information. DISPLAY_JOB, however, requires that a call be made to establish queue context; DISPLAY_ENTRY does not require that queue context be established.
DISPLAY_FILE	Returns information about the next file defined for the current job context. Before you make a call to F\$GETQUI to request file information, you must make a call to display queue and job information (with the DISPLAY_QUEUE and DISPLAY_JOB function codes) or display entry information (with the DISPLAY_ENTRY function code).
DISPLAY_FORM	Returns information about a specific form definition or the next form definition in a wildcard operation.
DISPLAY_JOB	Returns information about the next job defined for the current queue context. Before you make a call to F\$GETQUI to request job information, you must make a call to display queue information (with the DISPLAY_QUEUE function code). The DISPLAY_JOB function code is similar to the DISPLAY_ENTRY function code in that both return job information. DISPLAY_JOB, however, requires that a call be made to establish queue context; DISPLAY_ENTRY does not require that queue context be established.
DISPLAY_QUEUE	Returns information about a specific queue definition or the next queue definition in a wildcard operation.
TRANSLATE_QUEUE	Translates a logical name for a queue to the equivalence name for the queue.

Some function arguments cannot be specified with the item-code, object-id, or flags argument. The following table lists each function argument and corresponding format line to show whether the item-code, object-id, and flags arguments are required, optional, or not applicable for that specific function. In the following format lines, brackets denote an optional argument. An omitted argument means the argument is not applicable for that function. Note that two commas must be used as placeholders to denote an omitted (whether optional or not applicable) argument.

DCL-166 Lexical Functions

F\$GETQUI

Function	Format Line
CANCEL_OPERATION	F\$GETQUI("CANCEL_OPERATION") or F\$GETQUI(" ")
DISPLAY_CHARACTERISTIC	F\$GETQUI("DISPLAY_CHARACTERISTIC",[item],object-id,[flags])
DISPLAY_ENTRY	F\$GETQUI("DISPLAY_ENTRY",[item],[object-id],[flags])
DISPLAY_FILE	F\$GETQUI("DISPLAY_FILE",[item],[flags])
DISPLAY_FORM	F\$GETQUI("DISPLAY_FORM",[item],object-id,[flags])
DISPLAY_JOB	F\$GETQUI("DISPLAY_JOB",[item],[flags])
DISPLAY_QUEUE	F\$GETQUI("DISPLAY_QUEUE",[item],object-id,[flags])
TRANSLATE_QUEUE	F\$GETQUI("TRANSLATE_QUEUE",[item],object-id)

item

Corresponds to a \$GETQUI system service output item code. *Item* specifies the kind of information you want returned about a particular queue, job, file, form, or characteristic. Table DCL-8 lists each item code and the data type of the value returned for each item code.

object-id

Corresponds to the \$GETQUI system service search-name and search-number input item codes. *Object-id* specifies either the name or number of an object (for example, a specific queue name or form number) about which F\$GETQUI is to return information. Wildcard names are allowed for the following functions:

- DISPLAY_CHARACTERISTIC
- DISPLAY_ENTRY
- DISPLAY_FORM
- DISPLAY_QUEUE

By specifying a wildcard as the object-id argument on successive calls, you can get status information about one or more jobs in a specific queue or about files within jobs in a specific queue. When a wildcard name is used, each call returns information for the next object (queue, form, and so on) in the list. A null string (" ") is returned when the end of the list is reached. A wildcard can represent only object names, not object numbers.

flags

Specifies a list of keywords, separated by commas, that corresponds to the flags defined for the \$GETQUI system service search-flags input item code. (These flags are used to define the scope of the object search specified in the call to the \$GETQUI system service.) Note that these keywords can be used only with certain function codes:

Lexical Functions DCL-167
F\$GETQUI

Keyword	Valid Function Code	Description
ALL_JOBS	DISPLAY_JOB	Requests that F\$GETQUI search all jobs included in the established queue context. If you do not specify this flag, F\$GETQUI returns information only about jobs that have the same user name as the caller.
BATCH	DISPLAY_QUEUE DISPLAY_ENTRY	Selects batch queues.
EXECUTING_JOBS	DISPLAY_ENTRY DISPLAY_JOB	Selects executing jobs.
FREEZE_CONTEXT	DISPLAY_CHARACTERISTIC	When in wildcard mode, prevents advance of wildcard context to the next object. If you do not specify this flag, the context is advanced to the next object.
	DISPLAY_ENTRY DISPLAY_FILE DISPLAY_FORM DISPLAY_JOB DISPLAY_QUEUE	
GENERIC	DISPLAY_QUEUE	Selects generic queues for searching.
HOLDING_JOBS	DISPLAY_ENTRY DISPLAY_JOB	Selects jobs on unconditional hold.
PENDING_JOBS	DISPLAY_ENTRY DISPLAY_JOB	Selects pending jobs.
PRINTER	DISPLAY_QUEUE DISPLAY_ENTRY	Selects printer queues.
RETAINED_JOBS	DISPLAY_ENTRY DISPLAY_JOB	Selects jobs being retained.
SERVER	DISPLAY_QUEUE DISPLAY_ENTRY	Selects server queues.
SYMBIONT	DISPLAY_QUEUE	Selects all output queues. Equivalent to specifying "PRINTER,SERVER,TERMINAL".
	DISPLAY_ENTRY	
TERMINAL	DISPLAY_QUEUE DISPLAY_ENTRY	Selects terminal queues.

DCL-168 Lexical Functions
F\$GETQUI

Keyword	Valid Function Code	Description
THIS_JOB	DISPLAY_FILE	Selects all job file information about the calling batch job, the command file being executed, or the queue associated with the calling batch job.
	DISPLAY_JOB	
	DISPLAY_QUEUE	
TIMED_RELEASE_JOBS	DISPLAY_ENTRY	Selects jobs on hold until a specified time.
	DISPLAY_JOB	
WILDCARD	DISPLAY_CHARACTERISTIC	Establishes and saves a context. Because the context is saved, the next operation can be performed based on that context.
	DISPLAY_ENTRY	
	DISPLAY_FORM	
	DISPLAY_QUEUE	

description

The F\$GETQUI lexical function provides all the features of the \$GETQUI system service, including wildcard and nested wildcard operations.

The F\$GETQUI function returns information on all items that can be specified with the \$GETQUI system service. Table DCL-8 lists the items you can specify with the F\$GETQUI function, the information returned, and the data type of this information.

Table DCL-8: F\$GETQUI Items

Item	Return Type	Information Returned
ACCOUNT_NAME	String	The account name of the owner of the specified job.
AFTER_TIME	String	The system time at or after which the specified job can execute.
ASSIGNED_QUEUE_NAME	String	The name of the execution queue to which the logical queue specified in the call to F\$GETQUI is assigned.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
BASE_PRIORITY	Integer	The priority at which batch jobs are initiated from a batch execution queue or the priority of a symbiont process that controls output execution queues.
CHARACTERISTICS	String	The characteristics associated with the specified queue or job.
CHARACTERISTIC_NAME	String	The name of the specified characteristic.
CHARACTERISTIC_NUMBER	Integer	The number of the specified characteristic.
CHECKPOINT_DATA	String	The value of the DCL symbol BATCH\$RESTART when the specified batch job is restarted.
CLI	String	The name of the command language interpreter used to execute the specified batch job. The file specification returned assumes the device name SYS\$SYSTEM and the file type EXE.
COMPLETED_BLOCKS	Integer	The number of blocks that the symbiont has processed for the specified print job. This item code is applicable only to print jobs.
CONDITION_VECTOR	Integer	The completion status of the specified job.
CPU_DEFAULT	String	The default CPU time limit specified for the queue in 10-millisecond units. This item code is applicable only to batch execution queues.
CPU_LIMIT	String	The maximum CPU time limit specified for the specified job or queue in 10-millisecond units. This item code is applicable only to batch jobs and batch execution queues.
DEFAULT_FORM_NAME	String	The name of the default form associated with the specified output queue.
DEFAULT_FORM_STOCK	String	The name of the paper stock on which the specified default form is to be printed.
DEVICE_NAME	String	The node and device (or both) on which the specified execution queue is located. For batch execution queues, only the node name is returned. For output execution queues, only the device name is returned. The node name is used only in VAXcluster systems. The node name is specified by the SYSGEN parameter SCSNODE for the processor on which the queue executes.
ENTRY_NUMBER	Integer	The queue entry number of the specified job.

(continued on next page)

DCL-170 Lexical Functions
F\$GETQUI

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
EXECUTING_JOB_COUNT	Integer	The number of jobs in the queue that are currently executing.
FILE_BURST	String	"TRUE" or "FALSE" to indicate whether burst and flag pages are to be printed preceding a file.
FILE_CHECKPOINTED	String	"TRUE" or "FALSE" to indicate whether file is checkpointed.
FILE_COPIES	Integer	The number of times the specified file is to be processed. This item code is applicable only to output execution queues.
FILE_COPIES_DONE	Integer	The number of times the specified file has been processed. This item code is applicable only to output execution queues.
FILE_DELETE	String	"TRUE" or "FALSE" to indicate whether file is to be deleted after execution of request.
FILE_DOUBLE_SPACE	String	"TRUE" or "FALSE" to indicate whether the symbiont formats the file with double spacing.
FILE_EXECUTING	String	"TRUE" or "FALSE" to indicate whether file is being processed.
FILE_FLAG	String	"TRUE" or "FALSE" to indicate whether a flag page is to be printed preceding a file.
FILE_FLAGS	Integer	The processing options that have been selected for the specified file.
FILE_IDENTIFICATION	String	The internal file-identification value that uniquely identifies the selected file.
FILE_PAGE_HEADER	String	"TRUE" or "FALSE" to indicate whether page header is to be printed on each page of output.
FILE_PAGINATE	String	"TRUE" or "FALSE" to indicate whether symbiont paginates output by inserting a form feed whenever output reaches the bottom margin of the form.
FILE_PASSALL	String	"TRUE" or "FALSE" to indicate whether symbiont prints the file in PASSALL mode.
FILE_SETUP_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before the specified file is printed. This item code is meaningful only for output execution queues.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
FILE_SPECIFICATION	String	The fully qualified RMS file specification of the file about which F\$GETQUI is returning information.
FILE_STATUS	Integer	File status information.
FILE_TRAILER	String	"TRUE" or "FALSE" to indicate whether trailer page is to be printed following a file.
FIRST_PAGE	Integer	The page number at which the printing of the specified file is to begin. This item code is applicable only to output execution queues.
FORM_DESCRIPTION	String	The text string that describes the specified form to users and operators.
FORM_FLAGS	Integer	The processing options that have been selected for the specified form.
FORM_LENGTH	Integer	The physical length of the specified form in lines. This item code is applicable only to output execution queues.
FORM_MARGIN_BOTTOM	Integer	The bottom margin of the specified form in lines.
FORM_MARGIN_LEFT	Integer	The left margin of the specified form in characters.
FORM_MARGIN_RIGHT	Integer	The right margin of the specified form in characters.
FORM_MARGIN_TOP	Integer	The top margin of the specified form in lines.
FORM_NAME	String	The name of the specified form or the mounted form associated with the specified job or queue.
FORM_NUMBER	Integer	The number of the specified form.
FORM_SETUP_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before a file is printed on the specified form. This item code is meaningful only for output execution queues.
FORM_SHEET_FEED	String	"TRUE" or "FALSE" to indicate whether symbiont pauses at the end of each physical page so that another sheet of paper can be inserted.
FORM_STOCK	String	The name of the paper stock on which the specified form is to be printed.
FORM_TRUNCATE	String	"TRUE" or "FALSE" to indicate whether printer discards any characters that exceed the specified right margin.
FORM_WIDTH	Integer	The width of the specified form.

(continued on next page)

DCL-172 Lexical Functions
F\$GETQUI

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
FORM_WRAP	String	"TRUE" or "FALSE" to indicate whether the printer prints any characters that exceed the specified right margin on the following line.
GENERIC_TARGET	String	The names of the execution queues that are enabled to accept work from the specified generic queue. This item code is meaningful only for generic queues.
HOLDING_JOB_COUNT	Integer	The number of jobs in the queue being held until explicitly released.
INTERVENING_BLOCKS	Integer	The number of blocks to be processed before the specified job can begin to execute. This item code is meaningful only for output execution queues.
INTERVENING_JOBS	Integer	The number of jobs that are to be processed before the specified job can begin to execute. This item code is meaningful only for output execution queues.
JOB_ABORTING	String	"TRUE" or "FALSE" to indicate whether system is attempting to abort execution of job.
JOB_COPIES	Integer	The number of times the specified print job is to be repeated.
JOB_COPIES_DONE	Integer	The number of times that the specified print job has been repeated.
JOB_CPU_LIMIT	String	"TRUE" or "FALSE" to indicate whether a CPU time limit is specified for the job.
JOB_EXECUTING	String	"TRUE" or "FALSE" to indicate whether job is executing or printing.
JOB_FILE_BURST	String	"TRUE" or "FALSE" to indicate whether a burst page option is explicitly specified for the job.
JOB_FILE_BURST_ONE	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede only the first copy of the first file in the job.
JOB_FILE_FLAG	String	"TRUE" or "FALSE" to indicate whether flag page precedes each file in the job.
JOB_FILE_FLAG_ONE	String	"TRUE" or "FALSE" to indicate whether flag page precedes only the first copy of the first file in the job.
JOB_FILE_PAGINATE	String	"TRUE" or "FALSE" to indicate whether a paginate option is explicitly specified for the job.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
JOB_FILE_TRAILER	String	"TRUE" or "FALSE" to indicate whether trailer page follows each file in the job.
JOB_FILE_TRAILER_ONE	String	"TRUE" or "FALSE" to indicate whether trailer page follows only the last copy of the last file in the job.
JOB_FLAGS	Integer	The processing options that have been selected for the specified job. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of JOB_FLAGS: JOB_CPU_LIMIT, JOB_FILE_BURST, JOB_FILE_BURST_ONE, JOB_FILE_FLAG, JOB_FILE_FLAG_ONE, JOB_FILE_PAGINATE, JOB_FILE_TRAILER, JOB_FILE_TRAILER_ONE, JOB_LOG_DELETE, JOB_LOG_NULL, JOB_LOG_SPOOL, JOB_LOWERCASE, JOB_NOTIFY, JOB_RESTART, JOB_WSDEFAULT, JOB_WSEXTENT, and JOB_WSQUOTA.
JOB_HOLDING	String	"TRUE" or "FALSE" to indicate whether job will be held until it is explicitly released.
JOB_INACCESSIBLE	String	"TRUE" or "FALSE" to indicate whether caller does not have READ access to the specific job and file information in the system queue file. When FALSE, the DISPLAY_JOB and DISPLAY_FILE operations can return information for only the following output value item codes: AFTER_TIME COMPLETED_BLOCKS ENTRY_NUMBER INTERVENING_BLOCKS INTERVENING_JOBS JOB_SIZE JOB_STATUS
JOB_LIMIT	Integer	The number of jobs that can execute simultaneously on the specified queue. This item code is applicable only to batch execution queues.
JOB_LOG_DELETE	String	"TRUE" or "FALSE" to indicate whether the log file is deleted after it is printed.
JOB_LOG_NULL	String	"TRUE" or "FALSE" to indicate whether no log file is created.

(continued on next page)

DCL-174 Lexical Functions
F\$GETQUI

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
JOB_LOG_SPOOL	String	"TRUE" or "FALSE" to indicate whether job log file is queued for printing when job is complete.
JOB_LOWERCASE	String	"TRUE" or "FALSE" to indicate whether job is to be printed on printer that can print both uppercase and lowercase letters.
JOB_NAME	String	The name of the specified job.
JOB_NOTIFY	String	"TRUE" or "FALSE" to indicate whether message is broadcast to terminal when job completes or aborts.
JOB_PENDING	String	"TRUE" or "FALSE" to indicate whether job is pending.
JOB_PID	String	The process identification (PID) of the executing batch job.
JOB_REFUSED	String	"TRUE" or "FALSE" to indicate whether job was refused by symbiont and is waiting for symbiont to accept it for processing.
JOB_RESET_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before each job in the specified queue is printed. This item code is meaningful only for output execution queues.
JOB_RESTART	String	"TRUE" or "FALSE" to indicate whether job will restart after a system failure or can be requeued during execution.
JOB_RETAINED	String	"TRUE" or "FALSE" to indicate whether job has completed, but is being retained in the queue.
JOB_SIZE	Integer	The total number of blocks in the specified print job.
JOB_SIZE_MAXIMUM	Integer	The maximum number of blocks that a print job initiated from the specified queue can contain. This item code is applicable only to output execution queues.
JOB_SIZE_MINIMUM	Integer	The minimum number of blocks that a print job initiated from the specified queue can contain. This item code is applicable only to output execution queues.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
JOB_STARTING	String	"TRUE" or "FALSE" to indicate whether job controller is starting to process the job and has begun communicating with an output symbiont or a job controller on another node.
JOB_STATUS	Integer	The specified job's status flags.
JOB_SUSPENDED	String	"TRUE" or "FALSE" to indicate whether job is suspended.
JOB_TIMED_RELEASE	String	"TRUE" or "FALSE" to indicate whether job is waiting for specified time to execute.
JOB_WSDEFAULT	String	"TRUE" or "FALSE" to indicate whether default working set size is specified for the job.
JOB_WSEXTENT	String	"TRUE" or "FALSE" to indicate whether working set extent is specified for the job.
JOB_WSQUOTA	String	"TRUE" or "FALSE" to indicate whether working set quota is specified for the job.
LAST_PAGE	Integer	The page number at which the printing of the specified file should end. This item code is applicable only to output execution queues.
LIBRARY_SPECIFICATION	String	The name of the device control library for the specified queue. The library specification assumes the device and directory name SYS\$LIBRARY and a file type of TLB. This item code is meaningful only for output execution queues.
LOG_QUEUE	String	The name of the queue into which the log file produced for the specified batch job is to be entered for printing. This item code is applicable only to batch jobs.
LOG_SPECIFICATION	String	The name of the log file to be produced for the specified job. This item code is meaningful only for batch jobs.
NOTE	String	The note that is to be printed on the job flag and file flag pages of the specified job. This item code is meaningful only for output execution queues.
OPERATOR_REQUEST	String	The message that is to be sent to the queue operator before the specified job begins to execute. This item code is meaningful only for output execution queues.
OWNER_UIC	String	The owner UIC of the specified queue.

(continued on next page)

DCL-176 Lexical Functions
F\$GETQUI

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
PAGE_SETUP_MODULES	String	The names of the text modules to be extracted from the device control library and copied to the printer before each page of the specified form is printed. This item code is meaningful only for output execution queues.
PARAMETER_1 through PARAMETER_8	String	The value of the user-defined parameters that in batch jobs become the value of the DCL symbol P1 through P8, respectively.
PENDING_JOB_BLOCK_COUNT	Integer	The total number of blocks for all pending jobs in the queue (valid only for output execution queues).
PENDING_JOB_COUNT	Integer	The number of jobs in the queue in a pending state.
PENDING_JOB_REASON	Integer	The reason that the job is in a pending state.
PEND_CHAR_MISMATCH	String	"TRUE" or "FALSE" to indicate whether job requires characteristics that are not available on the execution queue.
PEND_JOB_SIZE_MAX	String	"TRUE" or "FALSE" to indicate whether block size of job exceeds the upper block limit of the execution queue.
PEND_JOB_SIZE_MIN	String	"TRUE" or "FALSE" to indicate whether block size of job is less than the lower limit of the execution queue.
PEND_LOWERCASE_MISMATCH	String	"TRUE" or "FALSE" to indicate whether job requires lowercase printer.
PEND_NO_ACCESS	String	"TRUE" or "FALSE" to indicate whether owner of job does not have access to the execution queue.
PEND_QUEUE_BUSY	String	"TRUE" or "FALSE" to indicate whether job is pending because the number of jobs currently executing on the queue equals the job limit for the queue.
PEND_QUEUE_STATE	String	"TRUE" or "FALSE" to indicate whether job is pending because the execution queue is not in a running, open state.
PEND_STOCK_MISMATCH	String	"TRUE" or "FALSE" to indicate whether the stock type required by the job's form does not match the stock type of the form mounted on the execution queue.
PRIORITY	Integer	The scheduling priority of the specified job.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
PROCESSOR	String	The name of the symbiont image that executes print jobs initiated from the specified queue.
PROTECTION	String	The specified queue's protection mask.
QUEUE_ACL_SPECIFIED	String	"TRUE" or "FALSE" to indicate whether an access control list has been specified for the queue.
QUEUE_ALIGNING	String	"TRUE" or "FALSE" to indicate whether queue prints a specified amount of output so that paper can be properly aligned.
QUEUE_BATCH	String	"TRUE" or "FALSE" to indicate whether queue is a batch queue or a generic batch queue.
QUEUE_CLOSED	String	"TRUE" or "FALSE" to indicate whether queue is closed and will not accept new jobs until the queue is put in an open state.
QUEUE_CPU_DEFAULT	String	"TRUE" or "FALSE" to indicate whether a default CPU time limit has been specified for all jobs in the queue.
QUEUE_CPU_LIMIT	String	"TRUE" or "FALSE" to indicate whether a maximum CPU time limit has been specified for all jobs in the queue.
QUEUE_FILE_BURST	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede each file in each job initiated from the queue.
QUEUE_FILE_BURST_ONE	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede only the first copy of the first file in each job initiated from the queue.
QUEUE_FILE_FLAG	String	"TRUE" or "FALSE" to indicate whether flag page precedes each file in each job initiated from the queue.
QUEUE_FILE_FLAG_ONE	String	"TRUE" or "FALSE" to indicate whether flag page precedes only the first copy of the first file in each job initiated from the queue.
QUEUE_FILE_PAGINATE	String	"TRUE" or "FALSE" to indicate whether output symbiont paginates output for each job initiated from this queue. The output symbiont paginates output by inserting a form feed whenever output reaches the bottom margin of the form.

(continued on next page)

DCL-178 Lexical Functions
F\$GETQUI

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
QUEUE_FILE_TRAILER	String	"TRUE" or "FALSE" to indicate whether trailer page follows each file in each job initiated from the queue.
QUEUE_FILE_TRAILER_ONE	String	"TRUE" or "FALSE" to indicate whether trailer page follows only the last copy of the last file in each job initiated from the queue.
QUEUE_FLAGS	Integer	The processing options that have been selected for the specified queue.
QUEUE_GENERIC	String	"TRUE" or "FALSE" to indicate whether the queue is a generic queue.
QUEUE_GENERIC_SELECTION	String	"TRUE" or "FALSE" to indicate whether the queue is an execution queue that can accept work from a generic queue.
QUEUE_IDLE	String	"TRUE" or "FALSE" to indicate whether queue prints a specified amount of output so that paper can be properly aligned.
QUEUE_JOB_BURST	String	"TRUE" or "FALSE" to indicate whether burst and flag pages precede each job initiated from the queue.
QUEUE_JOB_FLAG	String	"TRUE" or "FALSE" to indicate whether a flag page precedes each job initiated from the queue.
QUEUE_JOB_SIZE_SCHED	String	"TRUE" or "FALSE" to indicate whether jobs initiated from the queue are scheduled according to size, with the smallest job of a given priority processed first. (Meaningful only for output queues.)
QUEUE_JOB_TRAILER	String	"TRUE" or "FALSE" to indicate whether a trailer page follows each job initiated from the queue.
QUEUE_LOWERCASE	String	"TRUE" or "FALSE" to indicate whether queue is associated with a printer that can print both uppercase and lowercase characters.
QUEUE_NAME	String	The name of the specified queue or the name of the queue that contains the specified job.
QUEUE_PAUSED	String	"TRUE" or "FALSE" to indicate whether execution of all current jobs in the queue is temporarily halted.
QUEUE_PAUSING	String	"TRUE" or "FALSE" to indicate whether queue is temporarily halting execution. Currently executing jobs are completing; temporarily, no new jobs can begin executing.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
QUEUE_PRINTER	String	"TRUE" or "FALSE" to indicate whether the queue is a printer queue.
QUEUE_RECORD_BLOCKING	String	"TRUE" or "FALSE" to indicate whether the symbiont is permitted to concatenate, or block together, the output records it sends to the output device.
QUEUE_REMOTE	String	"TRUE" or "FALSE" to indicate whether queue is assigned to a physical device that is not connected to the local node.
QUEUE_RESETTING	String	"TRUE" or "FALSE" to indicate whether queue is resetting and stopping.
QUEUE_RESUMING	String	"TRUE" or "FALSE" to indicate whether queue is restarting after pausing.
QUEUE_RETAIN_ALL	String	"TRUE" or "FALSE" to indicate whether all jobs initiated from the queue remain in the queue after they finish executing. Completed jobs are marked with a completion status.
QUEUE_RETAIN_ERROR	String	"TRUE" or "FALSE" to indicate whether only jobs that do not complete successfully are retained in the queue.
QUEUE_SERVER	String	"TRUE" or "FALSE" to indicate whether queue processing is directed to a server symbiont.
QUEUE_STALLED	String	"TRUE" or "FALSE" to indicate whether physical device to which queue is assigned is stalled; that is, the device has not completed the last I/O request submitted to it.
QUEUE_STARTING	String	"TRUE" or "FALSE" to indicate whether queue is starting.
QUEUE_STATUS	Integer	The specified queue's status flags.
QUEUE_STOPPED	String	"TRUE" or "FALSE" to indicate whether queue is stopped.
QUEUE_STOPPING	String	"TRUE" or "FALSE" to indicate whether queue is stopping.
QUEUE_SWAP	String	"TRUE" or "FALSE" to indicate whether jobs initiated from the queue can be swapped.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
QUEUE_TERMINAL	String	"TRUE" or "FALSE" to indicate whether the queue is a generic queue that can place jobs only in terminal queues.
QUEUE_UNAVAILABLE	String	"TRUE" or "FALSE" to indicate whether physical device to which queue is assigned is not available.
QUEUE_WSDEFAULT	String	"TRUE" or "FALSE" to indicate whether default working set size is specified for each job initiated from the queue.
QUEUE_WSEXTENT	String	"TRUE" or "FALSE" to indicate whether working set extent is specified for each job initiated from the queue.
QUEUE_WSQUOTA	String	"TRUE" or "FALSE" to indicate whether working set quota is specified for each job initiated from the queue.
REQUEUE_QUEUE_NAME	String	The name of the queue to which the specified job is reassigned.
RESTART_QUEUE_NAME	String	The name of the queue in which the job will be placed if the job is restarted.
RETAINED_JOB_COUNT	Integer	The number of jobs in the queue retained after successful completion plus those retained on error.
SCSNODE_NAME	String	The 6-byte name of the VAX node on which jobs initiated from the specified queue execute. The node name matches the value of the SYSGEN parameter SCSNODE for the target node.
SUBMISSION_TIME	String	The time at which the specified job was submitted to the queue.
TIMED_RELEASE_JOB_COUNT	Integer	The number of jobs in the queue on hold until a specified time.
UIC	String	The UIC of the owner of the specified job.
USERNAME	String	The user name of the owner of the specified job.

(continued on next page)

Table DCL-8 (Cont.): F\$GETQUI Items

Item	Return Type	Information Returned
WSDEFAULT	Integer	The default working set size specified for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.
WSEXTENT	Integer	The working set extent specified for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.
WSQUOTA	Integer	The working set quota for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.

example

```
$ BLOCKS = F$GETQUI("DISPLAY_ENTRY", "JOB_SIZE", 1347)
```

In this example, the F\$GETQUI lexical function is used to obtain the size in blocks of print job 1347. The value returned reflects the total number of blocks occupied by the files associated with the job.

F\$GETSYI

Invokes the \$GETSYI system service to return status and identification information about the local system (or about a node in the local VAXcluster, if your system is part of a VAXcluster).

format

F\$GETSYI(*item* [,*node*])

arguments

item

Indicates the type of information to be reported about the local node (or about another node in your VAXcluster, if your system is part of a VAXcluster). Specify the item as a character string expression. You can specify the items in Table DCL-9 only for your local node; you cannot specify the node argument with these items. You can specify these items whether or not you are in a VAXcluster.

You can specify the items in Table DCL-10 for either your local node or for another node in your VAXcluster. The information in this table is returned for your local node if you do not specify the node argument; the information is returned for the specified node if you include the node argument. Your system must be a member of a VAXcluster in order to specify the items in this table, except for CLUSTER_MEMBER.

DCL-182 Lexical Functions

F\$GETSYI

node

Specifies the node in your VAXcluster for which information is to be returned. Specify the node as a character string expression. (This argument can be specified only if your system is part of a VAXcluster.)

You can request information about another node in your VAXcluster only when you specify an item from Table DCL-10. If you do not specify a node, the default is the current node. You cannot use wildcards to specify the node argument with the F\$GETSYI function (as you can with the \$GETSYI system service).

description

The F\$GETSYI returns information on the items that can be specified with the \$GETSYI system service.

Table DCL-9 lists the items you can specify with the F\$GETSYI lexical function to get information about your local node. Table DCL-10 lists the items you can specify to get information about either your local node, or another node in your VAXcluster.

Table DCL-9: F\$GETSYI Items for the Local Node Only

Item	Return Type	Information Returned
ACTIVECPU_CNT	Integer	The count of CPUs actively participating in the current boot of the SMP system.
AVAILCPU_CNT	Integer	The count of CPUs recognized in the system.
ARCHFLAG	String	Architecture flags for the system.
BOOTTIME	String	The time the system was booted.
CHARACTER_EMULATED	String	"TRUE" or "FALSE" to indicate whether the character string instructions are emulated on the CPU.
CPU	Integer	The processor type, as represented in the processor's SID register. For example, the integer 1 represents a VAX-11/780 and the integer 6 represents a VAX 8530, VAX 8550, VAX 8700 and VAX 8800.
DECIMAL_EMULATED	String	"TRUE" or "FALSE" to indicate whether the decimal string instructions are emulated on the CPU.

(continued on next page)

Table DCL-9 (Cont.): F\$GETSYI Items for the Local Node Only

Item	Return Type	Information Returned
D_FLOAT_EMULATED	String	"TRUE" or "FALSE" to indicate whether the D_floating instructions are emulated on the CPU.
ERRORLOGBUFFERS	Integer	Number of system pages in use as buffers for error logging.
F_FLOAT_EMULATED	String	"TRUE" or "FALSE" to indicate whether the F_floating instructions are emulated on the CPU.
G_FLOAT_EMULATED	String	"TRUE" or "FALSE" to indicate whether the G_floating instructions are emulated on the CPU.
PAGEFILE_FREE	Integer	Number of free pages in the currently installed paging files.
PAGEFILE_PAGE	Integer	Number of pages in the currently installed paging files.
SID	Integer	System identification register.
SWAPFILE_FREE	Integer	Number of free pages in the currently installed swapping files.
SWAPFILE_PAGE	Integer	Number of pages in the currently installed swapping files.
VERSION	String	Version of VMS in use (8-character string filled with trailing blanks).

Table DCL-10: F\$GETSYI Items for the Local Node or for Other Nodes in the VAXCluster

Item	Return Type	Information Returned
CLUSTER_FSYSID	String	System identification number for first node to boot in the VAXcluster (the founding node). This number is returned as a character string containing a hexadecimal number.
CLUSTER_FTIME	String	The time when the first node in the VAXcluster was booted.

(continued on next page)

Table DCL-10 (Cont.): F\$GETSYI Items for the Local Node or for Other Nodes in the VAXCluster

Item	Return Type	Information Returned
CLUSTER_MEMBER	String	"TRUE" or "FALSE" if the node is a member of the local VAXcluster.
CLUSTER_NODES	Integer	Total number of nodes in the VAXcluster, as an integer.
CLUSTER_QUORUM	Integer	Total quorum for the VAXcluster.
CLUSTER_VOTES	Integer	Total number of votes in the VAXcluster.
CONTIG_GBLPAGES	Integer	Total number of free, contiguous global pages.
FREE_GBLPAGES	Integer	Current count of free global pages.
FREE_GBLSECTS	Integer	Current count of free global section table entries.
HW_MODEL	Integer	An integer that identifies the node's VAX model type.
HW_NAME	String	The VAX model name.
NODENAME	String	Node name.
NODE_AREA	Integer	The VAX DECnet area for the target node.
NODE_CSID	String	The CSID of the specified node, as a string containing a hexadecimal number. The CSID is a form of system identification.
NODE_HWTYPE	String	Hardware type of the specified node.
NODE_HWVERS	String	Hardware version of the specified node.
NODE_NUMBER	Integer	The VAX DECnet number for the specified node.
NODE_QUORUM	Integer	Quorum that the node has.
NODE_SWINCARN	String	Software incarnation number for the specified node. This number is returned as a string containing a hexadecimal number.
NODE_SWTYPE	String	Type of operating system software used by the specified node.

(continued on next page)

Table DCL-10 (Cont.): F\$GETSYI Items for the Local Node or for Other Nodes in the VAXcluster

Item	Return Type	Information Returned
NODE_SWVERS	String	Software version of the specified node.
NODE_SYSTEMID	String	System identification number for the specified node. This number is returned as a string containing a hexadecimal number.
NODE_VOTES	Integer	Number of votes that the node has.
SCS_EXISTS	String	"TRUE" or "FALSE" to indicate whether the system communication subsystem (SCS) is currently loaded on a VAX node.

example

```
$ MEM = F$GETSYI("CLUSTER_MEMBER", "LONDON")
$ SHOW SYMBOL MEM
MEM = "TRUE"
```

This example uses the F\$GETSYI function to determine whether the node LONDON is a member of the local VAXcluster. The "TRUE" indicates that the remote node LONDON is a member of the VAXcluster.

F\$IDENTIFIER

Converts an alphanumeric identifier to its integer equivalent, or converts an integer identifier to its alphanumeric equivalent. An identifier is a name or number that identifies a category of users of a data resource. The system uses identifiers to determine a user's access to a resource.

format

F\$IDENTIFIER(*identifier,conversion-type*)

arguments

identifier

Specifies the identifier to be converted. Specify the identifier as an integer expression if you are converting an integer to a name. Specify the identifier as a character string expression if you are converting a name to an integer. The F\$IDENTIFIER function does not convert letters in the identifier to uppercase. Therefore, you must specify the identifier the same way it is defined in the rights database.

conversion-type

Indicates the type of conversion to be performed. If the identifier argument is alphanumeric, specify the translation argument as a character string containing "NAME_TO_NUMBER". If the identifier argument is numeric, specify the translation argument as a character string containing "NUMBER_TO_NAME".

example

```
$ UIC_INT= F$IDENTIFIER("SLOANE", "NAME_TO_NUMBER")
$ SHOW SYMBOL UIC_INT
UIC_INT = 15728665 Hex = 00F00019 Octal = 00074000031
$ UIC = F$FAO("!%U", UIC_INT)
$ SHOW SYMBOL UIC
UIC = [360,031]
```

This example uses the F\$IDENTIFIER to convert the member identifier from the UIC [MANAGERS,SLOANE] to an integer. The F\$IDENTIFIER function shows that the member identifier SLOANE is equivalent to the integer 15728665. Note that you must specify the identifier SLOANE using uppercase letters.

To convert this octal number to a standard numeric UIC, use the F\$FAO function with the !%U directive. (This directive converts a longword to a UIC in named format.) In this example, the member identifier SLOANE is equivalent to the numeric UIC [360,031].

F\$INTEGER

Returns the integer equivalent of the result of the specified expression.

format

F\$INTEGER(*expression*)

argument

expression

Specifies the expression to be evaluated. Specify either an integer or a character string expression. If you specify an integer expression, the F\$INTEGER function evaluates the expression and returns the result. If you specify a string expression, the F\$INTEGER function evaluates the expression, converts the resulting string to an integer, and returns the result. If the string contains characters that do not form a valid integer, the F\$INTEGER function returns the integer 1 if the string begins with T, t, Y, or y. The function returns the integer 0 if the string begins with any other character.

example

```
$ A = "23"
$ B = F$INTEGER("-9" + A)
$ SHOW SYMBOL B
  B = -923 Hex=FFFFFC65 Octal=176145
```

This example shows how to use the F\$INTEGER function to equate a symbol to the integer value returned by the function.

The F\$INTEGER function in the above example returns the integer equivalent of the string expression (“-9” + A). First, the F\$INTEGER function evaluates the string expression by concatenating the string literal “-9” with the string literal “23”. Note that the value of the symbol A is automatically substituted in a string expression. Also note that the plus sign (+) is a string concatenation operator since both arguments are string literals.

After the string expression is evaluated, the F\$INTEGER function converts the resulting character string (“-923”) to an integer, and returns the value -923. This integer value is assigned to the symbol B.

F\$LENGTH

Returns the length of the specified character string.

format

F\$LENGTH(*string*)

argument

string

Specifies the character string whose length is being determined. Specify the string argument as a character string expression.

example

```
$ MESSAGE = F$MESSAGE(%X1C)
$ SHOW SYMBOL MESSAGE
  MESSAGE = "%SYSTEM-F-EXQUOTA, exceeded quota"
$ STRING_LENGTH = F$LENGTH(MESSAGE)
$ SHOW SYMBOL STRING_LENGTH
  STRING_LENGTH = 33 Hex = 00000021 Octal = 000041
```

The first assignment statement uses the F\$MESSAGE function to return the message that corresponds to the hexadecimal value 1C. The message is returned as a character string and is assigned to the symbol MESSAGE.

The F\$LENGTH function is then used to return the length of the character string assigned to the symbol MESSAGE. You do not need to use quotation marks when you use the symbol MESSAGE as an argument for the F\$LENGTH function. (Quotation marks are not used around symbols in character string expressions.)

The F\$LENGTH function returns the length of the character string and assigns it to the symbol STRING_LENGTH. At the end of the example, the symbol STRING_LENGTH has a value equal to the number of characters in the value of the symbol named MESSAGE, that is, 33.

F\$LOCATE

Locates a specified portion of a character string and returns as an integer the offset of the first character. (An offset is the position of a character or a substring relative to the beginning of the string. The first character in a string is always offset position 0 from the beginning of the string.) If the substring is not found, F\$LOCATE returns the length (the offset of the last character in the character string plus one) of the searched string.

format

F\$LOCATE(*substring*,*string*)

arguments

substring

The character string, specified in the string argument, that you want to locate within the string.

string

The character string to be edited by F\$LOCATE.

example

```
$ INQUIRE TIME "Enter time"  
$ IF F$LOCATE(":",TIME) .EQ. F$LENGTH(TIME) THEN -  
GOTO NO_COLON
```

This section of a command procedure compares the results of the F\$LOCATE and F\$LENGTH functions to see if they are equal. This technique is commonly used to determine whether a character or substring is contained in a string.

In the example, the INQUIRE command prompts for a time value and assigns the user-supplied time to the symbol TIME. The IF command checks for the presence of a colon in the string entered in response to the prompt. If the value returned by the F\$LOCATE function equals the value returned by the F\$LENGTH function, the colon is not present. You use the .EQ. operator (rather than .EQS.) because the F\$LOCATE and F\$LENGTH functions return integer values.

Note that quotation marks are used around the substring argument, the colon, because it is a string literal. However, the symbol `TIME` does not require quotation marks because it is automatically evaluated as a string expression.

F\$MESSAGE

Returns as a character string the facility, severity, identification, and text associated with the specified system status code.

format

F\$MESSAGE(*status-code*)

argument

status-code

The status code for which you are requesting error message text. You must specify the status code as an integer expression.

example

```
$ ERROR_TEXT = F$MESSAGE(%X1C)
$ SHOW SYMBOL ERROR_TEXT
ERROR_TEXT = "%SYSTEM-F-EXQUOTA, exceeded quota"
```

This example shows how to use the `F$MESSAGE` function to determine the message associated with the status code `%X1C`. The `F$MESSAGE` function returns the message string, which is assigned to the symbol `ERROR_TEXT`.

F\$MODE

Returns a character string showing the mode in which a process is executing. Returns the character string "INTERACTIVE" for interactive processes. If the process is noninteractive, the character string "BATCH", "NETWORK" or "OTHER" is returned. The return string always contains uppercase letters. The `F$MODE` function has no arguments, but must be followed by parentheses.

format

F\$MODE()

arguments

None.

DCL-190 Lexical Functions

F\$MODE

example

```
$ IF F$MODE() .NES. "INTERACTIVE" THEN GOTO NON_INT_DEF
$ INTDEF:          ! Commands for interactive terminal sessions

.
.
.

$ EXIT
$ NON_INT_DEF:    !Commands for non-interactive processes

.
.
.
```

This example shows the beginning of a login.com file that has two sets of initialization commands: one for interactive mode and one for noninteractive mode (including batch and network jobs). The IF command compares the character string returned by F\$MODE with the character string INTERACTIVE; if they are not equal, control branches to the label NON_INT_DEF. If the character strings are equal, the statements following the label INTDEF are executed and the procedure exits before the statements at NON_INT_DEF.

F\$PARSE

Invokes the \$PARSE RMS service to parse a file specification and return as a character string either the expanded file specification or the particular file specification field that you request.

format

F\$PARSE(*file-spec* [,*default-spec*] [,*related-spec*] [,*field*] [,*parse-type*])

arguments

file-spec

Specifies a character string containing the file specification to be parsed. The file specification can contain wildcard characters. If you use a wildcard character, the file specification returned by the F\$PARSE function contains the wildcard.

default-spec

Specifies a character string containing the default file specification. The fields in the default file specification are substituted in the output string if a particular field in the file-spec argument is missing. You can make further substitutions in the file-spec argument by using the related-spec argument.

related-spec

Specifies a character string containing the related file specification. The fields in the related file specification are substituted in the output string if a particular field is missing from both the file-spec and default-spec arguments.

field

Specifies a character string containing the name of a field in a file specification. Specifying the field argument causes F\$PARSE to return a specific portion of a file specification.

Specify one of the following field names (do not abbreviate):

NODE	Node name
DEVICE	Device name
DIRECTORY	Directory name
NAME	File name
TYPE	File type
VERSION	File version number

parse-type

The type of parsing to be performed. By default, the F\$PARSE function verifies that the directory in the file specification exists on the device in the file specification. Note that the device and directory can be explicitly given in one of the arguments, or can be provided by default. Also, by default the F\$PARSE function translates logical names if they are provided in any of the arguments. You can change how the F\$PARSE function parses a file specification by using one of the following keywords:

NO_CONCEAL	Ignores the "conceal" attribute in the translation of a logical name as part of the file specification; that is, logical name translation does not end when a concealed logical name is encountered.
SYNTAX_ONLY	The syntax of the file specification is checked without verifying that the specified directory exists on the specified device.

description

When you use the F\$PARSE function, you can omit those optional arguments to the right of the last argument you specify. However, you must include commas as placeholders if you omit optional arguments to the left of the last argument you specify. If you omit the device and directory names in the file-spec argument, the F\$PARSE function supplies defaults, first from the default-spec argument and second from the related-spec argument. If names are not provided by these arguments, the F\$PARSE function uses your current default disk and directory. If you omit the file name, file type, or version number, the F\$PARSE function supplies defaults, first from the default-spec argument and second from the related-spec argument. If names are not provided by these arguments, the F\$PARSE function returns a null specification for these fields.

example

```
§ SET DEF DISK2:[FIRST]
§ SPEC = F$PARSE("JAMES.MAR","[ROOT]",,, "SYNTAX_ONLY")
§ SHOW SYMBOL SPEC
SPEC = "DISK2:[ROOT] JAMES.MAR;"
```

In this example, the F\$PARSE function returns the expanded file specification for the file JAMES.MAR. The example uses the SYNTAX_ONLY keyword to request that F\$PARSE should check the syntax, but should not verify that the [ROOT] directory exists on DISK2.

The default device and directory are DISK2:[FIRST]. Because the directory name [ROOT] is specified as the default-spec argument in the assignment statement, it is used as the directory name in the output string. Note that the default device returned in the output string is DISK2, and the default version number for the file is null. You must place quotation marks around the arguments JAMES.MAR and ROOT because they are string literals.

If you had not specified syntax-only parsing, and [ROOT] were not on DISK2, a null string would have been returned.

F\$PID

The F\$PID function returns a process identification (PID) number as a character string and updates the context symbol to point to the current position in the system's process list. You can step through all the processes on a system, or use the lexical F\$CONTEXT to specify selection criteria. The lexical F\$CONTEXT is not required.

The PIDs returned by the F\$PID function depend on the privilege of your process. If you have GROUP privilege, the F\$PID function returns PIDs of processes in your group. If you have WORLD privilege, the F\$PID function returns PIDs of all processes on the system. If you lack GROUP or WORLD privileges, the F\$PID function returns only those processes that you own.

After the first call initializes the context symbol, each subsequent F\$PID function call returns the next process in sequence, using the selection criteria set up by the F\$CONTEXT function, if any. After the last process in the list is returned, the F\$PID function returns a null string.

The F\$CONTEXT function enables the F\$PID function to retrieve processes from any node in a VAXcluster.

format

F\$PID(*context-symbol*)

argument

context-symbol

Specifies a symbol that DCL uses to store a pointer into the system's list of processes. The F\$PID function uses this pointer to return a PID. Specify the context symbol by using a symbol. The first time you use the F\$PID function in a command procedure, you should use a symbol that is either undefined or equated to the null string ("") or a context symbol that has been created by the F\$CONTEXT function.

example

```
$ CONTEXT = ""
$ START:
$   PID = F$PID(CONTEXT)
$   IF PID .EQS. "" THEN EXIT
$   SHOW SYMBOL PID
$   GOTO START
```

This command procedure uses the F\$PID function to display a list of PIDs. The assignment statement declares the symbol CONTEXT, which is used as the context-symbol argument for the F\$PID function. Because CONTEXT is equated to a null string, the F\$PID function returns the first PID in the process list that it has the privilege to access.

The PIDs displayed by this command procedure depend on the privilege of your process. When run with GROUP privilege, the PIDs of users in your group are displayed. When run with WORLD privilege, the PIDs of all users on the system are displayed. Without GROUP or WORLD privilege, only those processes that you own are displayed.

F\$PRIVILEGE

Returns a value of either "TRUE" or "FALSE", depending on whether your current process privileges match those specified in the argument. You can specify either the positive or negative version of a privilege.

format

F\$PRIVILEGE(*priv-states*)

argument

priv-states

A character string containing a privilege or a list of privileges separated by commas. Specify any one of the process privileges except [NO]ALL.

description

Use the F\$PRIVILEGE function to identify your current process privileges.

If "NO" precedes the privilege, the privilege must be disabled in order for the function to return a value of "TRUE". The F\$PRIVILEGE function checks each of the keywords in the specified list, and if the result for any one is false, the string "FALSE" is returned.

example

```
$ PROCPRIV = F$PRIVILEGE ("OPER, GROUP, TMPMBX, NONETMBX")
$ SHOW SYMBOL PROCPRIV
PROCPRIV = "FALSE"
```

The F\$PRIVILEGE function is used to test whether the process has OPER, USER, TMPMBX, and NETMBX privileges.

The process in this example has OPER, GROUP, TMPMBX, and NETMBX privileges. Therefore, a value of "FALSE" is returned because the process has NETMBX privilege, but NONETMBX was specified in the priv-states list. Although the Boolean result for the other three keywords is true, the entire expression is declared false because the result for NONETMBX was false.

F\$PROCESS

Obtains the current process name string. The F\$PROCESS function has no arguments, but must be followed by parentheses.

format

F\$PROCESS()

arguments

None.

example

```
$ NAME = F$PROCESS ()
$ SHOW SYMBOL NAME
NAME = "MARTIN"
```

In this example, the F\$PROCESS function returns the current process name and assigns it to the symbol NAME.

F\$SEARCH

Invokes the \$SEARCH RMS service to search a directory file and return the full file specification for a file you specify.

format

F\$SEARCH(*file-spec* [, *stream-id*])

arguments

file-spec

Specifies a character string containing the file specification to be searched for. If the device or directory names are omitted, the defaults from your current default disk and directory are used. The F\$SEARCH function does not supply defaults for a file name or type. If the version is omitted, the specification for the file with the highest version number is returned. If the file-spec argument contains wildcards, each time F\$SEARCH is called, the next file specification that agrees with the file-spec argument is returned. A null string is returned after the last file specification that agrees with the file-spec argument.

stream-id

A positive integer representing the search stream identification number. The search stream identification number is used to maintain separate search contexts when you use the F\$SEARCH function more than once and when you supply different file-spec arguments. If you omit *stream-id*, the F\$SEARCH function assumes an implicit single search stream. That is, the F\$SEARCH function starts searching at the beginning of the directory file each time you specify a different file-spec argument.

DCL-196 Lexical Functions

F\$SEARCH

example

```
$ START:
$   COM = F$SEARCH ("*.COM;*") ,1)
$   DAT = F$SEARCH ("*.DAT;*") ,2)
$   SHOW SYMBOL COM
$   SHOW SYMBOL DAT
$   IF (COM.EQS. "") .AND. (DAT.EQS. "") THEN EXIT
$   GOTO START
```

This command procedure searches the default disk and directory for both COM and DAT files. Note that the stream-id is specified for each F\$SEARCH function so that the context for each search is maintained.

The first F\$SEARCH function starts searching from the top of the directory file for a file with a type of COM. When it finds a COM file, a pointer is set to maintain the search context. When the F\$SEARCH function is used the second time, it again starts searching from the top of the directory file for a file with a type of DAT. When the procedure loops back to the label START, the stream-id argument allows each F\$SEARCH function to start searching in the correct place in the directory file. After all versions of COM and DAT files are returned, the procedure exits.

F\$SETPRV

Invokes the \$SETPRV system service to enable or disable specified user privileges. The F\$SETPRV function returns a list of keywords indicating user privileges; this list shows the status of the specified privileges before F\$SETPRV was executed. Your process must be authorized to set the specified privilege.

format

F\$SETPRV (*priv-states*)

argument

priv-states

A character string defining a privilege or a list of privileges separated by commas.

example

```
$ OLDPRIV = F$SETPRV ("OPER,NOTMPMBX")
$ SHOW SYMBOL OLDPRIV
OLDPRIV = "NOOPER,TMPMBX"
```

In this example, the process is authorized to change the OPER and TMPMBX privileges. The F\$SETPRV function enables the OPER privilege and disables the TMPMBX privilege. In addition, the F\$SETPRV function returns the keywords NOOPER and TMPMBX, showing the state of these privileges before they were changed.

You must place quotation marks around the list of privilege keywords because it is a string literal.

F\$STRING

Returns the string that is equivalent to the specified expression.

format

F\$STRING(*expression*)

argument

expression

The integer or string expression to be evaluated. If you specify an integer expression, the F\$STRING expression evaluates the expression, converts the resulting integer to a string, and returns the result. If you specify a string expression, the F\$STRING expression evaluates the expression and returns the result. When converting an integer to a string, the F\$STRING function uses decimal representation and omits leading zeroes. When converting a negative integer, the F\$STRING function places a minus sign at the beginning string representation of the integer.

example

```
$ A = 5
$ B = F$STRING(-2 + A)
$ SHOW SYMBOL B
B = "3"
```

The F\$STRING function in this example converts the result of the integer expression $(-2 + A)$ to the numeric string, "3". First, the F\$STRING function evaluates the expression $(-2 + A)$. Note that 5, the value of symbol A, is automatically substituted when the integer expression is evaluated.

After the integer expression is evaluated, the F\$STRING function converts the resulting integer, 3, to the string "3". This string is assigned to the symbol B.

F\$TIME

Returns as a character string the current date and time in absolute time format. The returned string has the following fixed, 23-character format:

dd-mmm-yyyy hh:mm:ss.cc

The F\$TIME function has no arguments, but must be followed by parentheses.

format

F\$TIME()

arguments

None.

example

```
$ OPEN/WRITE OUTFILE DATA.DAT
$ TIME_STAMP = F$TIME()
$ WRITE OUTFILE TIME_STAMP
```

This example shows how to use the F\$TIME function to time-stamp a file that you create from a command procedure. OUTFILE is the logical name for the file DATA.DAT, which is opened for writing. The F\$TIME function returns the current date and time string, and assigns this string to the symbol TIME_STAMP. The WRITE command writes the date and time string to OUTFILE.

F\$TRNLNM

Translates a logical name and returns the equivalence name string, or the requested attributes of the logical name specified. The return value can be a character string or an integer, depending on the arguments you specify with the F\$TRNLNM function. The translation is not iterative; the equivalence string is not checked to determine whether it is a logical name.

format

F\$TRNLNM(*logical-name* [,*table*] [,*index*] [,*mode*] [,*case*] [,*item*])

arguments

logical-name

Specifies a character string containing the logical name to be translated.

table

Specifies a character string containing the logical name table or tables that the F\$TRNLNM function should search to translate the logical name. The table argument must be a logical name that translates to a logical name table or to a list of table names. If you do not specify a table, the default value is LNM\$DCL_LOGICAL. Unless LNM\$DCL_LOGICAL has been redefined for your process, the F\$TRNLNM function searches the process, job, group, and system logical name tables, in that order, and returns the equivalence name for the first match found.

index

Specifies the number of the equivalence name to be returned if the logical name has more than one translation. If you do not specify the index argument, the default is 0.

mode

Specifies a character string containing one of the following access modes for the translation: USER (default), SUPERVISOR, EXECUTIVE, or KERNEL. The F\$TRNLNM function starts by searching for a logical name created with the access mode specified in the mode argument. If it does not find a match, the F\$TRNLNM function searches for the name created with each inner access mode and returns the first match found.

case

Specifies the type of case translation to be performed. Specify the case argument as either of the following character strings: CASE_BLIND (default) or CASE_SENSITIVE. If the translation is case blind, the F\$TRNLNM function first searches for a logical name with characters of the same case as the name argument. If no match is found, the F\$TRNLNM function searches for an uppercase version of the name argument and the logical names it is searching. The result of the first successful translation is returned. If the translation is case sensitive, the F\$TRNLNM function searches only for a logical name with characters of the same case as the name argument. The F\$TRNLNM function returns a null string if no exact match is found.

item

A character string containing the type of information that F\$TRNLNM should return about the specified logical name. Specify one of the following items:

DCL-200 Lexical Functions
F\$TRNLNM

Item	Return Type	Information Returned
ACCESS_MODE	String	One of the following access modes associated with the logical name: "USER", "SUPERVISOR", "EXECUTIVE", "KERNEL".
CONCEALED	String	Either "TRUE" or "FALSE" to indicate whether the CONCEALED attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created.
CONFINED	String	Either "TRUE" or "FALSE" to indicate whether the logical name is confined. If the logical name is confined (TRUE), then the name is not copied to subprocesses. If the logical name is not confined (FALSE), then the name is copied to subprocesses.
CRELOG	String	"TRUE" or "FALSE" to indicate whether the logical name was created with the \$CRELOG system service or with the \$CRELNM system service, using the CRELOG attribute.
LENGTH	Integer	Length of the equivalence name associated with the specified logical name. If the logical name has more than one equivalence name, the F\$TRNLNM function returns the length of the name specified by the index argument.
MAX_INDEX	Integer	The largest index defined for the logical name. The index shows how many equivalence names are associated with a logical name. The index is zero based; that is, the index 0 refers to the first name in a list of equivalence names.
NO_ALIAS	String	Either "TRUE" or "FALSE" to indicate whether the logical name has the NO_ALIAS attribute. The NO_ALIAS attribute means that a logical name must be unique within outer access mode.
TABLE	String	Either "TRUE" or "FALSE" to indicate whether the logical name is the name of a logical name table.
TABLE_NAME	String	Name of the table where the logical name was found.
TERMINAL	String	Either "TRUE" or "FALSE" to indicate whether the TERMINAL attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created. The TERMINAL attribute indicates that the logical name is not a candidate for iterative translation.
VALUE	String	Default. The equivalence name associated with the specified logical name. If the logical name has more than one equivalence name, the F\$TRNLNM function returns the name specified by the index argument.

example

```
$ DEFINE/TABLE=LNMSGROUP TERMINAL 'F$TRNLNM("SYS$OUTPUT")'
```

This example shows a line from a command procedure that (1) uses the F\$TRNLNM function to determine the name of the current output device and (2) creates a group logical name table entry based on the equivalence string.

You must enclose the argument SYS\$OUTPUT in quotation marks because it is a character string.

Also, in this example you must enclose the F\$TRNLNM function in single quotes to force the lexical function to be evaluated. Otherwise, the DEFINE command does not automatically evaluate the lexical function.

F\$TYPE

Returns the data type of a symbol. The string "INTEGER" is returned if the symbol is equated to an integer, or if the symbol is equated to a string whose characters form a valid integer.

The string "STRING" is returned if the symbol is equated to a character string whose characters do not form a valid integer.

If the symbol is undefined, a null string is returned.

format

F\$TYPE(*symbol-name*)

argument

symbol

The name of the symbol that is evaluated.

example

```
$ NUM = "52"  
$ TYPE = F$TYPE(NUM)  
$ SHOW SYMBOL TYPE  
TYPE = "INTEGER"
```

This example uses the F\$TYPE function to determine the data type of the symbol NUM. NUM is equated to the character string "52". Because the characters in the string form a valid integer, the F\$TYPE function returns the string "INTEGER".

F\$USER

Returns the current user identification code (UIC) in named format as a character string. The F\$USER function has no arguments, but must be followed by parentheses.

format

F\$USER()

arguments

None.

example

```
$ UIC = F$USER()  
$ SHOW SYMBOL UIC  
UIC = "[GROUP6, JENNIFER]"
```

In this example the F\$USER function returns the current user identification code and assigns it to the symbol UIC.

F\$VERIFY

Returns an integer value indicating whether the procedure verification setting is currently on or off. If used with arguments, the F\$VERIFY function can turn the procedure and image verification settings on or off. You must include the parentheses after the F\$VERIFY function whether or not you specify arguments.

format

F\$VERIFY(*[procedure-value]* [*,image-value*])

arguments

procedure-value

An integer expression with a value of 1 to turn procedure verification on, or 0 to turn procedure verification off. When procedure verification is on, each DCL command line in the command procedure is displayed on the output device. Procedure verification allows you to verify that each command is executing correctly.

image-value

An integer expression with a value of 1 to turn image verification on, or 0 to turn image verification off. When image verification is on, data lines in the command procedure are displayed on the output device.

description

Using the F\$VERIFY function in command procedures allows you to test the current procedure verification setting. For example, a command procedure can save the current procedure verification setting before changing it and then later restore the setting.

If you do not specify any arguments, neither of the verification settings is changed. If you specify only the procedure-value argument, both procedure and image verification are turned on (if the value is 1) or off (if the value is 0). If you specify both arguments, procedure and image verification are turned on or off independently. If you specify the image-value argument alone, only image verification is turned on or off. If you specify the image-value argument alone, you must precede the argument with a comma.

DCL performs the F\$VERIFY function if it appears after a comment character and if it is enclosed in single quotes. This is the only processing that DCL performs within a comment.

example

```
$ SAVE_PROC_VERIFY = F$ENVIRONMENT("VERIFY_PROCEDURE")
$ SAVE_IMAGE_VERIFY = F$ENVIRONMENT("VERIFY_IMAGE")
$ SET NOVERIFY
```

```
.
.
.
```

```
$ TEMP = F$VERIFY(SAVE_PROC_VERIFY, SAVE_IMAGE_VERIFY)
```

This example shows an excerpt from a command procedure. The first assignment statement assigns the current procedure verification setting to the symbol SAVE_PROC_VERIFY. The second assignment statement assigns the current image verification setting to the symbol SAVE_IMAGE_VERIFY.

Then, the SET NOVERIFY command disables procedure and image verification. Later, the F\$VERIFY function resets the verification settings, using the original values (equated to the symbols SAVE_PROC_VERIFY and SAVE_IMAGE_VERIFY). The symbol TEMP contains the procedure verification before it is changed with the F\$VERIFY function. (In this example the value of TEMP is not used.)

LIBRARY

Invokes the Librarian Utility to create, modify, or describe an object, macro, help, text, or shareable image library.

format

LIBRARY *library-file-spec* [*input-file-spec*,...]

LICENSE

Invokes the License Management Utility (LICENSE), used to manage software licenses on the VMS operating system. For a complete description of this utility, see the *VMS License Management Utility Manual*, part of the VMS Base Documentation Set.

LINK

Invokes the VMS Linker to link one or more object modules into a program image and defines execution characteristics of the image.

format

LINK *file-spec*,...]

parameter

***file-spec*,...]**

Specifies one or more input files (wildcard characters not allowed). Input files may be object modules, libraries to be searched for external references or from which specific modules are to be included, shareable images to be included in the output image or option files to be read by the linker. Separate multiple input file specifications with commas (,) or plus signs (+). In either case, the linker creates a single image file.

If you omit the file type in an input file specification, the linker supplies default file types, based on the nature of the file. For object modules, file type OBJ is assumed.

qualifiers

/BRIEF

Requests the linker to produce a brief map file; valid only with the */MAP* qualifier.

/CONTIGUOUS

/NOCONTIGUOUS (*default*)

Controls whether the output image file is contiguous.

/CROSS_REFERENCE
/NOCROSS_REFERENCE (default)

Controls whether the map contains a symbol cross-reference list with entries for each global symbol referenced in the image, its value, and all modules in the image that refer to it.

/DEBUG[=file-spec]
/NODEBUG

If the object module contains local symbol table or traceback information, you can specify */DEBUG* to include the information in the image as well. If the object module does not contain local symbol table or traceback information, only global symbols are available for symbolic debugging.

If you specify the */DEBUG* qualifier, by default, the VAX Symbolic Debugger is linked with the image. However, you can use the file-spec option to specify an alternate debugger (wildcard characters are not allowed).

/EXECUTABLE[=file-spec]
/NOEXECUTABLE

Permits you to specify whether or not the linker creates an executable image. By default the linker creates an executable image with the same file name as the first input file and a file type of EXE but this qualifier gives you the option of assigning the image a file specification (wildcard characters not allowed). The placement of the command qualifier determines the output file specification defaults.

/FULL

Requests a full map listing; valid only with */MAP* qualifier.

/HEADER

Provides a system image header when used with the */SYSTEM* qualifier.

/INCLUDE=(module-name[,...])

Positional qualifier. Selects modules from the associated object module library or image library as input to the linking operation. No wildcard characters are allowed in the module name specifications. If you specify several modules, separate them with commas and enclose the list in parentheses.

/LIBRARY

Positional qualifier. Indicates that the associated input file is a library (default file type OLB) whose modules should be searched to resolve undefined symbols. You are not permitted to specify a library as the first input file unless you also specify the */INCLUDE* qualifier to indicate which modules in the library are to be included in the input.

/MAP[=file-spec]
/NOMAP

Permits you to specify whether or not a memory allocation listing (map) is produced and gives you the option of assigning it a file specification. In interactive mode, the default is */NOMAP*; in batch mode, the default is */MAP*. You can specify the map's contents using either the */BRIEF*, */FULL*, or */CROSS_REFERENCE* qualifiers.

/OPTIONS

Positional qualifier. Indicates that the associated input file (default file type *OPT*) contains a list of linking options.

/POIMAGE

Directs the linker to create an image in P0 address space together with the stack and the VMS RMS buffers that usually go in P1 address space.

/PROTECT

When used with the */SHAREABLE* qualifier, directs the linker to create a protected shareable image that can execute privileged change mode instructions even when it is linked into a nonprivileged executable image.

/SELECTIVE_SEARCH

Positional qualifier. Use this qualifier when you want the linker to omit from the output image symbol table, all symbols from the associated input object module that are not needed to resolve outstanding references.

/SHAREABLE[=file-spec]
/NOSHAREABLE

Command qualifier. By default, the linker creates an executable image. If you specify the */SHAREABLE* qualifier, the linker creates a shareable image file instead. Optionally, you may designate a name for the output file; however, wildcard characters are not permitted. To specify an input shareable image, the */SHAREABLE* qualifier must be used as an input file qualifier in an options file.

/SHAREABLE
/SHAREABLE=NOCOPY

Positional qualifier. Use this positional qualifier in the context of an options file only to identify an input file as a shareable image file. The keyword *NOCOPY* tells the linker not to bind a private copy of the shareable image to the executable image.

/SYMBOL_TABLE[=file-spec]
/NOSYMBOL_TABLE

The default is */NOSYMBOL_TABLE* (do not create a symbol table). Use the */SYMBOL_TABLE* qualifier when you want the linker to create a symbol table object module file (default file type *STB*) that contains symbol definitions for all global symbols in the image being linked. The symbol table file can be subsequently specified in *LINK* commands to provide the symbol definitions to other images.

When you specify `/SYMBOL_TABLE`, you can control the defaults applied to the output file specification. Optionally, you may designate a name for the symbol table file, but you may not use wildcard characters.

`/SYSLIB`
`/NOSYSLIB`

The default is `/SYSLIB` (search the system libraries). Use the `/NOSYSLIB` qualifier to prevent the linking operation from automatically searching the default system libraries, `SY$LIBRARY:IMAGELIB.OLB` and then `SY$LIBRARY:STARLET.OLB`, for unresolved references in the input files.

`/SYSSHR`
`/NOSYSSHR`

The default is `/SYSSHR` (search the default system shareable image library). Use the `/NOSYSSHR` qualifier to prevent the linking operation from automatically searching the default system shareable image library, `SY$LIBRARY:IMAGELIB.OLB`, for unresolved references.

`/SYSTEM[=base-address]`
`/NOSYSTEM`

The default is `/NOSYSTEM` (do not produce a system image). Use the `/SYSTEM` qualifier to produce a system image and optionally assign it a base address. You cannot use the `/SYSTEM` qualifier with either the `/SHAREABLE` qualifier or the `/DEBUG` qualifier. The base address specifies where the image is to be loaded in virtual memory. It can be expressed in decimal, hexadecimal, or octal format, using the radix specifiers `%D`, `%X`, or `%O`, respectively. The default base address is `%X80000000`.

`/TRACEBACK (default)`
`/NOTRACEBACK`

Default is `/TRACEBACK` (include traceback information in the image file to help the system trace the call stack when an error occurs). Use the `/NOTRACEBACK` qualifier to prevent the linker from including traceback information.

If you specify `/DEBUG`, `/TRACEBACK` is assumed.

`/USERLIBRARY[=(table[,...])]`
`/USERLIBRARY=ALL`

You use this qualifier to specify which user-defined default libraries (process, group, system or, by default, all three) the linker searches after it has searched any specified user libraries. The `/NOUSERLIBRARY` qualifier tells the linker not to search any user-defined default libraries.

example

```
$ LINK/MAP/FULL DRACO,CYGNUS,LYRA
```

The LINK command in this example links the modules DRACO.OBJ, CYGNUS.OBJ, and LYRA.OBJ and creates an executable image named DRACO.EXE. The /MAP and /FULL qualifiers request a full map of the image, with descriptions of each program section, lists of global symbols by name and by value, and a summary of the image characteristics. The map file is named DRACO.MAP.

LOGIN Procedure

Initiates an interactive terminal session.

format

CTRL/C

CTRL/Y

RET

qualifiers

/CLI=command-language-interpreter

Specifies the name of an alternate command language interpreter (CLI) to override the default CLI listed in the user authorization file. The CLI you specify must be located in SYS\$SYSTEM and have the file type EXE.

If you do not specify a command interpreter using the /CLI qualifier and do not have a default CLI listed in the user authorization file, the system supplies a default of /CLI=DCL.

/COMMAND[=file-spec]

/NOCOMMAND

Controls whether to execute your default login command procedure when you log in. Use the /COMMAND qualifier to specify the name of an alternate login command procedure. If you specify a file name without a file type, the default file type COM is used. If you specify /COMMAND and omit the file specification, your default login command procedure is executed. By default, /COMMAND is assumed.

Use the /NOCOMMAND qualifier if you do not want your default login command procedure to be executed.

/DISK=device-name[:]

Specifies the name of a disk device to be associated with the logical device SYS\$DISK for the terminal session. This specification overrides the default SYS\$DISK device established in the authorization file.

/NEW_PASSWORD

If you specify this qualifier when you log in to the system, you must change the account password before logging in (as if the password had expired). Use this qualifier as a shortcut if you had intended to change your password after login, or if you suspect that your password has been detected.

/TABLES=(command-table[,...])

Specifies the name of an alternate CLI table to override the default listed in the user authorization file (UAF). This table name is considered a file specification. The default device and directory is SYS\$SHARE and the default file type is EXE.

If a logical name is used, the table name specification must be defined in the system logical name table.

If the */CLI* qualifier is set to DCL or MCR, the */TABLES* qualifier defaults to the correct value. If the */TABLES* qualifier is specified without the */CLI* qualifier, the CLI specified in the user's UAF will be used.

The default is */TABLES=DCLTABLES*.

example

RET

```
Username: JONES
Password:
User authorization failure
```

RET

```
Username: JONES
Password:
Welcome to VAX/VMS Version 5.00 on node JUPITER
Last interactive login on Tuesday, 19-APR-1990 09:16:47.08
Last non-interactive login on Monday, 19-APR-1990 17:32:34.27
1 failure since last successful login.
```

\$

This example shows the "User authorization failure" message, which indicates that the password has been entered incorrectly. After successfully logging in, a message is displayed showing the number of login failures since your last successful login. This message is displayed only if one or more login failures have occurred.

RET

```
Username: MIHALY/NEW_PASSWORD
Password:
Password:
Welcome to VAX/VMS Version V5.2 on node JUPITER
Last interactive login on Tuesday, 19-APR-1990 09:16:47.08
Last non-interactive login on Monday, 19-APR-1990 17:32:34.27
Your password has expired; you must set a new password to log in.

Old password:

New password:
```

In this example, the user enters the `/NEW_PASSWORD` qualifier after the username `MIHALY`. The system then forces the user to set a new password immediately after login. The prompts are the same as those provided when you enter the DCL command `SET PASSWORD` from the command line.

LOGOUT

Terminates an interactive terminal session.

format

LOGOUT

qualifiers

/BRIEF

Prints a brief logout message (process name, date, and time) or a full logout message (a brief message plus accounting statistics).

/FULL

Requests the long form of the logout message. When you specify `/FULL`, the command interpreter displays a summary of accounting information for the terminal session. The default for a batch job is `/FULL`.

/HANGUP

/NOHANGUP

For dialup terminals, determines whether or not the phone hangs up whenever you log out. By default, the `/HANGUP` setting of your terminal port determines whether the line is disconnected. Your system manager determines whether you are permitted to use this qualifier.

example

```
$ LOGOUT/FULL
HIGGINS   logged out at 19-APR-1990 14:23:45.30
Accounting information:
Buffered I/O count:      22      Peak working set size:      90
Direct I/O count:       10      Peak virtual size:          69
Page faults:            68      Mounted volumes:             0
Charged CPU time: 0 00:01:30.50  Elapsed time:           0 04:59:02.63
```

In this example, the `LOGOUT` command with the `/FULL` qualifier displays a summary of accounting statistics for the terminal session.

MACRO

Invokes the VAX MACRO assembler to assemble one or more assembly language source files.

See the qualifier descriptions for restrictions.

format

MACRO *file-spec-list*

parameter

file-spec-list

Requests the assembly of one or more VAX MACRO assembly language source files. The *file-spec-list* parameter consists of one or more file specifications. For each file specification, the MACRO command supplies a default file type of MAR. You cannot include a wildcard character in a file specification.

File specifications separated by commas cause the MACRO assembler to produce an object file (and, if indicated, a listing file) for each specified file. File specifications separated by plus signs are concatenated into one input file and produce a single object file (and listing file). The MACRO assembler creates output files of one version higher than the highest version existing in the target directory.

qualifiers

/ANALYSIS_DATA[=*file-spec*]
/NOANALYSIS_DATA (*default*)

Controls whether the assembler creates an analysis data file for the VAX Source Code Analyzer (SCA), and optionally provides the file specification.

By default, the assembler does not create an analysis data file. If you specify **/ANALYSIS_DATA** without a file specification, the assembler creates a file with the same file name as the first input file for the MACRO command. The default file type for analysis data files is ANA. When you specify **/ANALYSIS_DATA**, you can control the defaults applied to the output file specification by the placement of the qualifier in the command.

/CROSS_REFERENCE[=(*function*[,...])]
/NOCROSS_REFERENCE (*default*)

Controls whether a cross-reference listing of the locations in the source file where the specified function (or functions) is defined or referenced. If you specify more than one function, separate each with a comma and enclose the entire list in parentheses. You can specify the following functions:

DCL-212 DCL Commands

MACRO

ALL	Cross-references directives, macros, operation codes, registers, and symbols
DIRECTIVES	Cross-references directives
MACROS	Cross-references macros
OPCODES	Cross-references operation codes
REGISTERS	Cross-references registers
SYMBOLS	Cross-references symbols

Because the assembler writes the cross-references to the listing file, you must specify the `/LIST` qualifier with the `/CROSS_REFERENCE` qualifier. If you specify no functions in the `/CROSS_REFERENCE` qualifier, the assembler assumes the default value of `/CROSS_REFERENCE=(MACROS,SYMBOLS)`. The `/NOCROSS_REFERENCE` qualifier excludes the cross-reference listing.

/DEBUG[=option] ***/NODEBUG (default)***

Includes or excludes local symbols in the symbol table or traceback information in the object module. You can replace `/ENABLE` and `/DISABLE` with `/DEBUG` and `/NODEBUG` when you use the appropriate `DEBUG` and `TRACEBACK` options. `/DEBUG` or `/NODEBUG` override debugging characteristics set with the `.ENABLE` or `.DISABLE` assembler directives. You can specify one or more of the following options:

ALL	Includes in the object module all local symbols in the symbol table, and provides all traceback information for the debugger. This qualifier is equivalent to <code>/ENABLE=(DEBUG,TRACEBACK)</code> .
NONE	Makes local symbols and traceback information in the object module unavailable to the debugger. This qualifier is equivalent to <code>/DISABLE=(DEBUG,TRACEBACK)</code> .
SYMBOLS	Makes all local symbols in the object module available to the debugger. Makes traceback information unavailable to the debugger. This qualifier is equivalent to <code>/ENABLE=DEBUG</code> and <code>/DISABLE=TRACEBACK</code> together.
TRACEBACK	Makes traceback information in the object module available to the debugger and local symbols unavailable to the debugger. This qualifier is equivalent to <code>/ENABLE=TRACEBACK</code> and <code>/DISABLE=DEBUG</code> together.

If you specify no options to the `/DEBUG` qualifier, it assumes the default value of `/DEBUG=ALL`.

/DIAGNOSTICS[=file-spec] ***NODIAGNOSTICS (default)***

Creates a file containing assembler messages and diagnostic information. If you omit the file specification, the default file name is the same as the source program; the default file type is `DIA`.

No wildcard characters are allowed in the file specification.

The diagnostics file is reserved for use with DIGITAL layered products, such as, but not limited to, the VAX Language-Sensitive Editor (LSE).

/DISABLE=(function[,...])
/NODISABLE

Provides initial settings for the functions disabled by the **.DISABLE** assembler directive. You can specify one or more of the following functions:

ABSOLUTE	Assembles relative addresses as absolute addresses
DEBUG	Includes local symbol table information in the object file for use with the debugger
TRUNCATION	Truncates floating-point numbers (if truncation is disabled, numbers are rounded)
GLOBAL	Assumes undefined symbols to be external symbols
SUPPRESSION	Suppresses listing of unreferenced symbols in the symbol table
TRACEBACK	Provides traceback information to the debugger

If you specify more than one function, separate each with a comma and enclose the list with parentheses. If you specify no functions in the **/DISABLE** qualifier, it assumes the default value of **/DISABLE=(ABSOLUTE,DEBUG, TRUNCATION)**. The **/NODISABLE** qualifier has the same effect as not specifying the **/DISABLE** qualifier, or negates the effects of any **/DISABLE** qualifiers specified earlier on the command line.

/ENABLE=(function[,...])
/NOENABLE

Provides initial settings for the functions controlled by the **.ENABLE** assembler directive. The **/NOENABLE** qualifier has the same effect as not specifying the **/ENABLE** qualifier, or negates the effects of any **/ENABLE** qualifiers specified earlier on the command line. You can specify one or more of the functions as listed in the description of the **/DISABLE** qualifier, separating each with a comma and enclosing the list in parentheses. If you specify no functions in the **/DISABLE** qualifier, it assumes the default value of **/ENABLE=(GLOBAL,TRACEBACK,SUPPRESSION)**.

/LIBRARY
/NOLIBRARY

Positional qualifier. The **/LIBRARY qualifier cannot be used with the **/UPDATE** qualifier.** The associated input file to the **/LIBRARY** qualifier must be a macro library. The default file type is **MLB**. The **/NOLIBRARY** qualifier has the same effect as not specifying the **/LIBRARY** qualifier, or negates the effects of any **/LIBRARY** qualifiers specified earlier on the command line.

DCL-214 DCL Commands
MACRO

The assembler can search up to 16 libraries, one of which is always STARLET.MLB. This number applies to a particular assembly, not necessarily to a particular MACRO command. If you enter the MACRO command so that more than one source file is assembled, but the source files are assembled *separately*, you can specify up to 16 macro libraries for each separate assembly. More than one macro library in an assembly causes the libraries to be searched in reverse order of their specification.

A macro call in a source program causes the assembler to begin the following sequence of searches:

1. An initial search of the libraries specified with the .LIBRARY directive. The assembler searches these libraries in the reverse order of that in which they were declared.
2. If the macro definition is not found in any of the libraries specified with the .LIBRARY directive, a search of the libraries specified in the DCL MACRO command line (in the reverse order in which they were specified).
3. If the macro definition is not found in any of the libraries specified in the command line, a search of STARLET.MLB.

/LIST[=file-spec]
/NOLIST

Creates or omits an output listing, and optionally provides an output file specification for it. The default file type for the listing file is LIS. No wildcard characters are allowed in the file specification.

An interactive MACRO command does not produce a listing file by default. /NOLIST, present either explicitly or by default, causes errors to be reported on the current output device.

/LIST is the default for a MACRO command in a batch job. /LIST allows you to control the defaults applied to the output file specification by the placement of the qualifier in the command.

/OBJECT[=file-spec]
/NOBJECT

Creates or omits an object module. It also defines the file specification. By default, the assembler creates an object module with the same file name as the first input file. The default file type for object files is .OBJ. No wildcard characters are allowed in the file specification.

/OBJECT controls the defaults applied to the output file specification by the placement of the qualifier in the command.

***/SHOW[(function[,...])
/NOSHOW[(function[,...])]***

Provides initial settings for the functions controlled by the assembler directives `.SHOW` and `.NOSHOW`. You can specify one or more of the following functions:

<code>CONDITIONALS</code>	Lists unsatisfied conditional code associated with <code>.IF</code> and <code>.ENDC MACRO</code> directives
<code>CALLS</code>	Lists macro calls and repeat range expansions
<code>DEFINITIONS</code>	Lists macro definitions
<code>EXPANSIONS</code>	Lists macro expansions
<code>BINARY</code>	Lists binary code generated by the expansion of macro calls

If you specify more than one function, separate each with a comma and enclose the list in parentheses. If you specify no functions in the `/SHOW` qualifier, it increments the listing level count; the `/NOSHOW` qualifier decrements the count in similar circumstances. Because these qualifiers contribute to the listing file, you must also specify the `/LIST` qualifier when you use them. Updates the input file it qualifies using the SLP batch editor and the specified update file or files. By default, the assembler assumes that the update file has the same file name as the input source file and a file type of `UPD`. You cannot include a wildcard character in the file specifications. If you specify more than one update file specification, separating each with a comma and enclosing the list in parentheses, the assembler merges the contents into a single list of updates before applying the updates to the source file.

The `/NOUPDATE` qualifier has the same effect as not specifying the `/UPDATE` qualifier, or negates any `/UPDATE` qualifiers specified earlier on the command line. The input source file and update files are not changed by the update operation. The effects of the update appear in the compiled output. If you specify the `/LIST` qualifier with the `/UPDATE` qualifier, the assembler writes an audit trail of the changes to the listing file.

example

```
$ MACRO/LIST CYGNUS, LYRA/OBJECT=LYRAN + MYLIB/LIBRARY
```

In this example, the `MACRO` command requests two separate assemblies. Using `.MAR` as the default, `MACRO` assembles `CYGNUS.MAR` to produce `CYGNUS.LIS` and `CYGNUS.OBJ`. Then it assembles `LYRA.MAR` and creates a listing file named `LYRA.LIS` and an object module named `LYRAN.OBJ`. The default output file type for a listing is `.LIS`.

The command requests the search of the `MYLIB` library file in the current directory for macro definitions.

MAIL

Invokes the Mail Utility (MAIL), which is used to send messages to other users of the system. For a complete description of the Mail Utility, including more information about the MAIL command and its qualifiers, see the Reference Section.

format

MAIL *[file-spec] [recipient-name]*

MERGE

Invokes the Sort/Merge Utility to combine two through ten similarly sorted input files and create a single output file. Note that input files to be merged must be in sorted order. For a complete description of the Sort/Merge Utility, including more information about the MERGE command and its qualifiers, see the Reference Section.

format

MERGE *input-file-spec1, input-file-spec2[,...] output-file-spec*

MESSAGE

Invokes the Message Utility (MESSAGE) to compile one or more files of message definitions.

format

MESSAGE *file-spec[,...]*

MONITOR

Invokes the Monitor Utility (MONITOR) to monitor classes of systemwide performance data at a specified interval.

format

MONITOR *[class-name[,...]]*

MOUNT

Invokes the Mount Utility (MOUNT) to make a disk or magnetic tape volume available for processing. For more information about the Mount Utility, see the *VMS System Manager's Manual* in the VMS Base Documentation Set.

NCS

Invokes the VMS National Character Set Utility to provide a convenient method of implementing alternative (non-ASCII) string collating sequences, typically using subsets of the DEC Multinational Character Set. NCS also facilitates the implementation of string conversion functions.

format

NCS *[file-spec,...]*

ON

Performs a specified action when a command or program executed within a command procedure encounters an error condition or is interrupted by CTRL/Y. The specified actions are performed only if the command interpreter is enabled for error checking or CTRL/Y interrupts (the default conditions). Use the ON command only in a command procedure.

format

ON *condition THEN [\$] command*

parameters

condition

Either the severity level of an error or a CTRL/Y interrupt. Specify one of the following keywords, which may be abbreviated to one or more characters:

WARNING	Return status of warning occurs (\$SEVERITY equals 0)
ERROR	Return status of error occurs (\$SEVERITY equals 2)
SEVERE_ERROR	Return status of error occurs (\$SEVERITY equals 4)
CONTROL_Y	CTRL/Y character occurs on SYS\$INPUT

The default error condition is ON ERROR THEN EXIT.

DCL-218 DCL Commands

ON

To specify a CRTL/Y interrupt, use the following keyword:

CONTROL/Y

command

The DCL command line to be executed. You can optionally precede the command line with a dollar sign (\$). If you specified an error condition as the condition parameter, the action is taken when errors equal to or greater than the specified level of error occur.

example

```
$ ON WARNING THEN EXIT
```

```
.  
. .  
. .
```

```
$ SET NOON  
$ RUN [SSTEST]LIBRA  
$ SET ON
```

```
.  
. .  
. .
```

The ON command requests that the procedure exit when any warning, error, or severe error occurs. Later, the SET NOON command disables error checking before executing the RUN command. Regardless of any status code returned by the program LIBRA.EXE, the procedure continues. The next command, SET ON, reenables error checking and reestablishes the most recent ON condition.

OPEN

Opens a file for reading, writing, or both, assigns a logical name to a file, and places the name in the process logical name table.

See the qualifier descriptions for restrictions.

format

OPEN *logical-name[:]* *file-spec*

parameters***logical-name[:]***

Specifies the logical name and a character string to be assigned to the file.

file-spec

Specifies the name of the file or device being opened for input or output. The file type defaults to DAT. Wildcard characters are not allowed. To create a new, sequential file, specify the /WRITE qualifier.

qualifiers***/APPEND***

Opens an existing file for writing and positions the record pointer at the end-of-file. New records are added to the end of the file. Use the /APPEND qualifier only to add records to an existing file. The /APPEND and the /WRITE qualifiers are mutually exclusive.

/ERROR=label

Transfers control to the location specified by the label keyword (in a command procedure) if the OPEN operation results in an error. The error routine specified for this qualifier overrides any ON condition action specified. If /ERROR is not specified, the current ON condition action is taken.

/READ (default)

Opens the file for reading. If you specify the /READ qualifier without the /WRITE qualifier, you must specify an existing file.

/SHARE[=option]

Opens the specified file as a shareable file to allow other users read or write access. If you specify /SHARE=READ, users are allowed read access to the file. If you specify /SHARE=WRITE or omit the option, users are allowed read and write access to the specified file.

/WRITE

Opens the file for writing. The following restrictions apply to the /WRITE qualifier:

- Use the /WRITE qualifier to open and create a new, sequential file.
- Use the /READ qualifier with the /WRITE qualifier to open an existing file. You cannot use OPEN/READ/WRITE to create a new file.
- The /WRITE and the /APPEND qualifiers are mutually exclusive.

DCL-220 DCL Commands

OPEN

example

```
$ OPEN/WRITE/ERROR=OPEN_ERROR OUTPUT_FILE TEMP.OUT
$ COUNT = 0
$ WRITE_LOOP:
$ COUNT = COUNT +1
$ IF COUNT .EQ. 11 THEN GOTO ENDIT
$ WRITE OUTPUT_FILE "Count is ''COUNT'.'"
.
.
.

$ GOTO WRITE_LOOP
$ ENDIT:
$ CLOSE OUTPUT_FILE
$ EXIT
$
$ OPEN_ERROR:
$ WRITE SYS$OUTPUT "Cannot open file TEMP.OUT"
$ EXIT
```

The OPEN command with the /WRITE qualifier creates the file TEMP.OUT and assigns it the logical name OUTPUT_FILE. TEMP.OUT is a sequential file.

The /ERROR qualifier specifies that if any error occurs while opening the file, the command interpreter should transfer control to the line at the label OPEN_ERROR. The command procedure writes records to the file TEMP.OUT until the symbol COUNT equals 11.

PASSWORD

When submitting a batch job through a card reader, provides the password associated with the user name that is specified with the JOB card. Although the PASSWORD card is required, the password on the card is optional if the account has a null password.

The PASSWORD command is valid only in a batch job submitted through a card reader and requires that a dollar sign precede the PASSWORD command on the card.

format

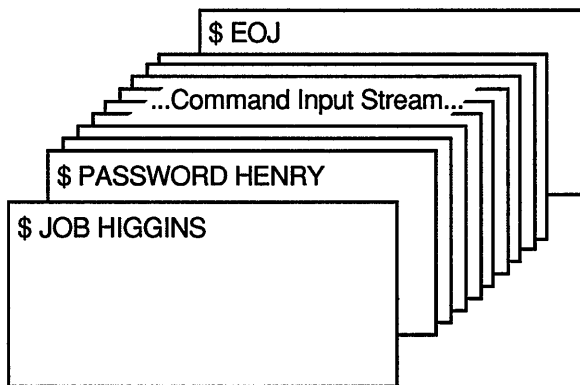
```
$ PASSWORD [password]
```


parameter

password

Specifies the password associated with the user name specified with the JOB command. *Password* can be 1 to 31 characters long. If you are submitting the job from an account with a null password, omit the password specifier on the PASSWORD card.

example



ZK-0786-GE

The JOB and PASSWORD commands precede a batch job submitted from the card reader. An EOJ command marks the end of the job.

PATCH

Invokes the Patch Utility (PATCH) to patch an executable image, shareable image, or device driver image.

format

PATCH *file-spec*

PHONE

Invokes the Phone Utility that allows you to communicate with other users on your system or any other VMS system connected to your system by DECnet-VAX.

PHONE can be used only on video terminals that are supported by the VMS screen package. These terminals include all terminals such as the VT100-, VT200-, and VT300-series that support ANSI terminal escape sequences and VT52 terminals.

format

PHONE [*phone-command*]

PRINT

Queues one or more files for printing.

Requires OPER privilege, EXECUTE (E) access to the queue, or WRITE (W) access to the queue.

format

PRINT *file-spec[,...]*

parameter

file-spec[,...]

Specifies one or more files to be printed. Wildcard characters are allowed. The default file type is that of the preceding file. If no previous file specification contains an explicit file type, the default file type is LIS.

If you specify two or more files, separate the file specifications with either commas or plus signs.

If you specify a node name, you must use the /REMOTE qualifier.

qualifiers

/AFTER=*time*

/NOAFTER

Holds the job until the specified time. The time can be specified as an absolute time or a combination of absolute and delta times. If the specified time has passed, the job is queued for printing immediately.

/BACKUP

/NOBACKUP

Modifies the interpretation of the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other

qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]
/NOBEFORE

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BURST[=keyword]
/NOBURST

Positional qualifier. Controls whether two file flag pages with a burst bar between them are printed preceding a file. If the /BURST qualifier is specified between the PRINT command and the file specifications, it can take either of two keywords:

ALL Prints the flag pages and a burst bar before each file in the job
ONE Prints the flag pages and a burst bar before the first file in the job

If you want the /BURST qualifier to apply to individual files in a multifile job, place the qualifier directly after each file that you want to have the flag pages and a burst bar.

/BY_OWNER[=uic]
/NOBY_OWNER

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CHARACTERISTICS=(characteristic[,...])

Specifies the name or number of one or more characteristics to be associated with the job. Characteristics can refer to such things as color of ink. If you specify only one characteristic, you can omit the parentheses.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each file is queued for printing to confirm that the operation should be performed on that file. The following responses are valid:

DCL-224 DCL Commands

PRINT

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	<code><RET></code>	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and `<RET>`. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/COPIES=*n*

Positional qualifier. Specifies the number of copies to print. The value of *n* can be from 1 to 255 and defaults to 1. If you place the /COPIES qualifier after the PRINT command name, each file in the parameter list is printed the specified number of times. If you specify /COPIES following a file specification, only that file is printed the specified number of times.

/CREATED (default)

/NOCREATED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/DELETE

/NODELETE (default)

Positional qualifier. Controls whether files are deleted after printing. If you place the /DELETE qualifier after the PRINT command name, all specified files are deleted. If you specify /DELETE after a file specification, only that file is deleted after it is printed.

/DEVICE=*queue-name*[:]

Places the print job in the specified queue (rather than the default queue SYS\$PRINT). This qualifier is synonymous with /QUEUE, except that the /DEVICE qualifier is reserved for special use by Digital. Its usage, therefore, is not recommended.

/EXCLUDE=(*file-spec*[,...])

/NOEXCLUDE

Excludes the specified files from the print operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative

version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED
/NOEXPIRED

Modifies the interpretation of the time value specified with the */BEFORE* or */SINCE* qualifier. */EXPIRED* selects files according to their expiration dates. (The expiration date is set with the *SET FILE/EXPIRATION_DATE* command.) The */EXPIRED* qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

/FEED
/NOFEED

Positional qualifier. Controls whether form feeds are inserted into the print job when the printer reaches the bottom margin of the form in use. The */[NO]FEED* qualifier does not affect user-formatted files.

/FLAG[=keyword]
/NOFLAG

Positional qualifier. Controls whether a file flag page is printed preceding a file. The flag page contains the name of the user submitting the job, the job entry number, and other information about the file being printed. If the */FLAG* qualifier is positioned between the *PRINT* command and the file specifications, it can take either of two keywords:

ALL Prints a file flag page before each file in the job
ONE Prints a file flag page before the first file in the job

If you want the */FLAG* qualifier to apply to individual files in a multifile job, place the qualifier directly after each file that you want to have a flag page.

/FORM=form

Specifies the name or number of the form to be associated with the print job. If you omit the */FORM* qualifier, the default form for the execution queue is associated with the job. To see which forms have been defined for your system, use the *SHOW QUEUE/FORM* command.

/HEADER
/NOHEADER (default)

Positional qualifier. Controls whether a heading line is printed at the top of each page.

/HOLD
/NOHOLD (default)

Controls whether a job is available for printing immediately. The */HOLD* qualifier holds the job until released by a *SET ENTRY/RELEASE* or *SET ENTRY/NOHOLD* command.

DCL-226 DCL Commands
PRINT

/IDENTIFY (default)

/NOIDENTIFY

Displays the job name, queue name, entry number, and status of the job when it is queued.

/JOB_COUNT=n

Prints the job *n* times. The value of *n* can be from 1 through 255 and defaults to 1.

/LOWERCASE

/NOLOWERCASE (default)

Indicates whether the print job must be printed on a printer that can print both lowercase and uppercase letters. The */NOLOWERCASE* qualifier means that files can be printed on printers supporting only uppercase letters. If all available printers can print both uppercase and lowercase letters, you do not need to specify */LOWERCASE*.

/MODIFIED

/NOMODIFIED

Modifies the interpretation of the time value specified with the */BEFORE* or */SINCE* qualifier. The qualifier */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/NAME=job-name

Names the job. The name consists of 1 through 39 alphanumeric characters. If characters other than alphanumerics, underscores, or dollar signs are used in the name, enclose the name in quotation marks. The default is the name of the first (or only) file in the job.

/NOTE=string

Specifies a message string of up to 255 characters to appear on the flag page of the job.

/NOTIFY

/NONOTIFY (default)

Controls whether a message is broadcast to your terminal session when the job is printed or aborted.

/OPERATOR=string

Specifies a message of up to 255 characters to be sent to the operator when the job begins to print.

/PAGES=(*[lowlim,]uplim*)

Positional qualifier. Specifies the number of pages to print for the specified job. The *lowlim* specifier refers to the first page in the group of pages that you want printed for that file. If you omit the *lowlim* specifier, the printing starts on the first page of the file. The *uplim* specifier refers

to the last page of the file that you want printed. If you want to print to the end of the file, but do not know how many pages the file contains, use two consecutive quotation marks (" ") as the uplim specifier.

/PARAMETERS=(parameter[,...])

Specifies from one to eight optional parameters to be passed to the job; each parameter can contain up to 255 characters. Enclose parameters containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (" ").

/PASSALL

/NOPASSALL (default)

Positional qualifier. Specifies whether the symbiont bypasses all formatting and sends the output QIO to the driver with format suppressed. All qualifiers affecting formatting, as well as the /HEADER, /PAGES, and /SETUP qualifiers, are ignored.

/PRIORITY=n

Requires OPER or ALTPRI privilege to raise the priority above the SYSGEN parameter MAXQUEPRI. Specifies the job-scheduling priority of the print job. The value of *n* can be from 0 through 255, where 0 is the lowest priority and 255 is the highest. The default value of *n* is the value of the SYSGEN parameter DEFQUEPRI. No privilege is needed to set the priority lower than the MAXQUEPRI value.

/QUEUE=queue-name[:]

Queues the job to the specified output queue. The default queue is SYS\$PRINT. This qualifier is synonymous with /DEVICE.

/REMOTE

Queues the job to SYS\$PRINT on the remote node specified in the file specification; the file *must* exist on the remote node. When you use /REMOTE, you *must* include the node name in the file specification.

You can only specify the following qualifiers with /REMOTE: /BACKUP, /BEFORE, /BY_OWNER, /CONFIRM, /CREATED, /EXCLUDE, /EXPIRED, /MODIFIED, and /SINCE.

/RESTART (default)

/NORESTART

Indicates whether a job restarts after a system failure or after a STOP/QUEUE/REQUEUE command.

/SETUP=module[,...]

Extracts the specified modules from the device control library and copies the modules to the printer before a file is printed. By default, no device control modules are copied.

DCL-228 DCL Commands
PRINT

/SINCE[=time]
/NOSINCE

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /SINCE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/SPACE
/NOSPACE (default)

Positional qualifier. Controls whether print job output is double-spaced. The default is single-spaced output.

/TRAILER[=keyword]
/NOTRAILER

Positional qualifier. Controls whether a file trailer page is printed at the end of a file. The trailer page displays the job entry number as well as information about the user submitting the job and the files being printed. If the /TRAILER qualifier is positioned between the PRINT command and the file specifications, it can take either of two keywords:

ALL Prints a file trailer page after each file in the job
ONE Prints a file trailer page after the last file in the job

If you want the /TRAILER qualifier to apply to individual files in a multifile job, place the qualifier directly after each file that you want to have a trailer page.

/USER=username

Requires the CMKRNL privilege and R (READ) and W (WRITE) access to the user authorization file (UAF). Allows you to print a job on behalf of another user. The print job runs exactly as if that user had submitted it.

example

```
$ PRINT ALPHA.TXT + BETA/FLAG + GAMMA/FLAG + *.LIS/FLAG  
Job ALPHA (queue SYS$PRINT, entry 237) pending
```

The PRINT command in this example submits the files ALPHA.TXT, BETA.TXT, GAMMA.TXT, and the highest versions of all files with the file type LIS as a single print job. Flag pages separate the individual files. Notice that the file type for BETA and GAMMA is TXT, the file type of the first file in the list.

PURGE

Deletes all but the highest-numbered versions of the specified files.

format

PURGE [*file-spec*[,...]]

parameter

***file-spec*[,...]**

Specifies one or more files to be purged. Wildcard characters are allowed in the directory, file name, and file type fields; however, no version number can be specified. As a default, the PURGE command purges all files in the current directory. There are no file name or file type defaults with the PURGE command.

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

***/BEFORE*[=*time*]**

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

***/BY_OWNER*[=*uic*]**

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

***/NOCONFIRM* (default)**

Controls whether a request is issued before each PURGE operation to confirm that the operation should be performed on that file. The following responses are valid:

DCL-230 DCL Commands
PURGE

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/ERASE

/NOERASE (default)

Erases the specified files from the disk so that the purged data no longer exists physically on the deallocated disk blocks.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the PURGE operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/KEEP=number-of-versions

Specifies the maximum number of versions of the specified files to be retained in the directory. If you do not include the /KEEP qualifier, all but the highest-numbered version of the specified files are deleted from the directory.

/LOG

/NOLOG (default)

Controls whether file specifications are displayed as the files are deleted.

/MODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

example

```
$ PURGE [MAL.TESTFILES]/LOG
%PURGE-I-FILPURG, DISK1:[MAL.TESTFILES]AVE.OBJ;1 deleted (3 blocks)
%PURGE-I-FILPURG, DISK1:[MAL.TESTFILES]BACK.OBJ;2 deleted (5 blocks)
%PURGE-I-TOTAL, 2 files deleted (8 blocks)
```

The PURGE command in this example purges all files cataloged in the subdirectory named [MAL.TESTFILES]. The /LOG qualifier requests the PURGE command to display the specification of each file it has deleted as well as the total number of files that have been deleted.

READ

Reads a single record from a specified input file and assigns the record's contents to a specified symbol name.

format

READ *logical-name[:]* *symbol-name*

parameters

logical-name[:]

Specifies the logical name of the input file from which a record is to be read. Use the logical name assigned by the OPEN command when the file was opened. In addition, you can specify the process-permanent files identified by the logical names SYS\$INPUT, SYS\$OUTPUT, SYS\$ERROR, and SYS\$COMMAND.

READ***symbol-name***

Specifies the name of a symbol to be equated to the contents of the record. The name must be 1 through 255 alphanumeric characters and must start with an alphabetic letter, underscore, or dollar sign. When you specify a symbol name for the READ command, the command interpreter places the symbol name in the local symbol table for the current command level. If the symbol has already been defined, the READ command redefines it to the new value being read.

qualifiers***/DELETE***

Deletes a record from an ISAM file after it has been read. An ISAM file must be opened with the /READ and /WRITE qualifiers in order to use READ/DELETE.

/END_OF_FILE=label

Transfers control to the location specified by the label keyword (in the current command procedure) when the end of the file is reached. When the last record in the file is read, the VMS Record Management Services (VMS RMS) return an error condition indicating the end-of-file. If the /END_OF_FILE qualifier is specified, the command interpreter transfers control to the command line at the specified label.

If /END_OF_FILE is not specified, control is given to the error label specified with the /ERROR qualifier when the end of the file is reached. If neither /ERROR nor /END_OF_FILE is specified, then the current ON condition action is taken.

/ERROR=label

Transfers control to the location specified by the label keyword (in the current command procedure) when a read error occurs. If no error routine is specified and an error occurs during the reading of the file, the current ON condition action is taken. Overrides any ON condition action specified. If an error occurs and the target label is successfully given control, the reserved global symbol \$STATUS retains the error code.

/INDEX=n

Specifies the index (n) to be used to look up keys when reading an ISAM file. The default value is 0, the primary index.

/KEY=string

Reads a record with the key that matches the specified character string. Binary and integer keys are not allowed. This qualifier, when used together with /INDEX, allows you random access to ISAM files. Key matches are made by comparing the characters in the /KEY string to characters in the record key.

/MATCH=option

Specifies the ISAM key match algorithm to be used when searching for matching keys. Specify one of the following options:

- EQ Select keys equal to the match value (default)
- GE Select keys greater than or equal to the match value
- GT Select keys greater than the specified key

If you are reading ISAM files and you do not use the */MATCH* qualifier, the default is */MATCH=EQ*.

/NOLOCK

Specifies that the record to be read not be locked and enables a record to be read that has been locked by other accessors. By default, records are locked as they are read and unlocked on the next I/O operation on the file.

/PROMPT=string

Specifies an alternate prompt string to be displayed when reading from the terminal. The default prompt string is *DATA:*.

/TIME_OUT=n

/NOTIME_OUT (default)

Specifies the number of seconds after which the *READ* command is terminated if no input is received. If you enter the */TIME_OUT* qualifier, you must specify a value from 0 through 255. If you enter both the */ERROR=label* and */TIME_OUT* qualifiers, and the time limit expires, the error branch is taken.

example

```
$ READ/ERROR=READERR/END_OF_FILE=OKAY MSGFILE CODE
```

```
.  
. .  
. .
```

```
$ READERR:  
$ CLOSE MSGFILE
```

```
.  
. .  
. .
```

DCL-234 DCL Commands

READ

```
$ OKAY:  
$ CLOSE MSGFILE  
$ EXIT
```

The READ command reads records from the file MSGFILE and places the contents into the symbol CODE. The READ command also uses the /ERROR and /END_OF_FILE qualifiers to specify labels to receive control at the end-of-file and on error conditions. At the end-of-file, control is transferred to the label OKAY. On other read errors, control is transferred to the READERR label.

RECALL

Displays previously entered commands on the screen for subsequent execution.

format

RECALL [*command-specifier*]

parameter

command-specifier

Specifies the number or the first several characters of the command you want to recall. The specified characters should be unique. If they are not unique, the RECALL command displays the most recently entered command line that matches those characters. The number of the command can be from 1 to 20 (where 1 is the last command entered). The RECALL command itself is never assigned a number. If no command specifier is entered, the RECALL command recalls the most recently entered command.

qualifiers

/ALL

Displays all the commands (and their numbers) available for recall.

/ERASE

Erases the contents of the recall buffer.

example

```
$ RECALL T
```

The RECALL command in this example recalls the last command entered that begins with the letter *T*.

RENAME

Changes all or part of a file specification of an existing disk file or disk directory.

format

RENAME *input-file-spec[,...] output-file-spec*

parameters

input-file-spec[,...]

Specifies the name of one or more files whose specifications are to be changed. Wildcard characters are allowed.

output-file-spec

Provides the new file specification to be applied to the input file. The RENAME command uses the device, directory, file name, and file type of the input file as defaults for fields in the output file that are either not specified, or indicated by a wildcard character. The RENAME command supplies output file version numbers in the following ways:

1. If the output file specification contains an explicit version number, that version number is used.
2. If the output file specification contains a wildcard as the version number, the version number of the input file is used.
3. If the input file specification contains a wildcard as the version number, the version number of each input file names a corresponding output file.
4. If no file exists with the same file name and type as the output file, version 1 is used.
5. If a file already exists with the same file name and type as the output file, the next higher version number is used (unless the /NONEWVERSION qualifier is specified).

qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /BACKUP selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /CREATED, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

/NOCONFIRM (default)

Controls whether a request is issued before each RENAME operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	<input type="checkbox"/>	RET

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the RENAME operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */EXPIRED* selects files according to their expiration dates. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

/LOG

/NOLOG (default)

Displays the file specification of each file as it is renamed.

/MODIFIED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/NEW_VERSION (default)

/NONEW_VERSION

Assigns a new version number if an output file specification is the same as that of an existing file. The */NONEW_VERSION* qualifier displays an error message if an output file specification is the same as that of an existing file. A wildcard appearing in the version field of an input or output file overrides these qualifiers.

/SINCE[=time]

Selects the *RENAME* operation only for those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: *TODAY* (default), *TOMORROW*, or *YESTERDAY*. Specify one of the following qualifiers with */BEFORE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

example

```
$ RENAME/NONEW_VERSION SCANLINE.OBJ;2 BACKUP.OBJ
```

The *RENAME* command in this example renames the file *SCANLINE.OBJ;2* to *BACKUP.OBJ;2*. The */NONEW_VERSION* qualifier ensures that, if *BACKUP.OBJ;2* already exists, the *RENAME* command does not rename the file, but instead reports the error.

REPLY

Broadcasts a message to a terminal or terminals.

See the qualifier descriptions for restrictions.

format

REPLY [*message-text*]

parameter

message-text

Specifies the text of the message. The text must be 1 to 128 characters. Enclose the text in quotation marks (") if it contains spaces, special characters, or lowercase characters.

qualifiers

/ABORT=identification-number

Sends a message to the user or magnetic tape file system corresponding to the unique identification number and cancels the request.

/ALL

Requires OPER privilege. Broadcasts a message to all terminals that are attached to the system or VAXcluster. These terminal must be turned on and have broadcast-message reception enabled. Incompatible with **/USERNAME** and **/TERMINAL**.

/BELL

Rings a bell at the terminal receiving a message when entered with the **/ALL**, **/TERMINAL**, or **/USERNAME** qualifiers; two bells when entered with **/URGENT**; and three bells when entered with **/SHUTDOWN**.

/BLANK_TAPE=identification-number

Requires VOLPRO privilege. Sends a message to the magnetic tape file system indicated by the identification number to override the checking of volume label information. The volume label must be specified in the message text parameter. The current terminal must be enabled as an operator terminal for TAPES.

/DISABLE[=(keyword[,...])]

Requires OPER privilege. Requires OPER and SECURITY privileges for security messages. If the Operator Communication Facility (OPCOM) is running, restores to normal (that is, nonoperator) status the terminal at which the command is entered. The **/DISABLE** qualifier cannot be entered from a batch job. To restrict the types of messages displayed on an operator's terminal, specify one of the following keywords:

CARDS	Inhibits messages sent to the card readers
CENTRAL	Inhibits messages sent to the central system operator
CLUSTER	Inhibits messages from the connection manager pertaining to cluster state changes
DEVICES	Inhibits messages pertaining to mounting disks
DISKS	Inhibits messages pertaining to mounting and dismounting disk volumes
NETWORK	Inhibits messages pertaining to networks; the keyword CENTRAL must also be specified to inhibit network messages
OPER1 through OPER12	Inhibits messages sent to operators identified as OPER1 through OPER12
PRINTER	Inhibits messages pertaining to print requests
SECURITY	Inhibits messages pertaining to security events; requires SECURITY privilege.
TAPES	Inhibits messages pertaining to mounting and dismounting tape volumes

/ENABLE[=(keyword[,...])]

Requires OPER privilege. Requires OPER and SECURITY privileges for security messages. If the Operator Communication Facility (OPCOM) is running, designates as an operator's terminal the terminal at which the REPLY command is entered. Cannot be entered from a batch job. To enable the following types of messages displayed on an operator's terminal, specify one of the following keywords:

CARDS	Displays messages sent to the card readers
CENTRAL	Displays messages sent to the central system operator
CLUSTER	Displays messages from the connection manager pertaining to cluster state changes
DEVICES	Displays messages pertaining to mounting disks
DISKS	Displays messages pertaining to mounting and dismounting disk volumes
NETWORK	Displays messages pertaining to networks; the keyword CENTRAL must also be specified to inhibit network messages
OPER1 through OPER12	Displays messages sent to operators identified as OPER1 through OPER12
PRINTER	Displays messages pertaining to print requests

DCL-240 DCL Commands
REPLY

SECURITY Allows messages pertaining to security events; requires SECURITY privilege

TAPES Allows messages pertaining to mounting and dismounting tape volumes

/INITIALIZE_TAPE=identification-number

Sends a message to the magnetic tape file system indicated by the identification number to initialize a magnetic tape volume. This qualifier can be used whenever the file system requests the mounting of a new volume. The system performs normal protection and expiration checks before initializing the volume. The current terminal must be enabled as an operator terminal for TAPES.

If the tape drive cannot read the volume, the mount fails and an error message is returned. Use the /BLANK_TAPE qualifier to override the checking of information on a volume label.

/LOG

/NOLOG

Requires OPER privilege. If the Operator Communication Facility (OPCOM) is running, closes the current operator's log file and opens a new one. (The /NOLOG qualifier closes the current log file, but does not open a new log file.) The current terminal must be enabled as an operator terminal.

/NODE[=(node-name[,...])]

Sends a message to the local VAXcluster node only. The optional parameter list allows you to specify which nodes will receive the message. Default sends messages to all cluster nodes.

/NOTIFY (default)

/NONOTIFY

Sends a message describing success back to the originating terminal.

/PENDING=identification-number

Requires OPER privilege. Sends a message to the user specified by the identification number and prevents the user from entering other commands until the operator fulfills or aborts the request. The current terminal must be enabled as an operator terminal.

/SHUTDOWN

Sends a message beginning "SHUTDOWN..."; if used with /BELL, rings three bells at terminals receiving the message.

/STATUS

Requires OPER privilege. Reports the current operator status and all outstanding user requests for the terminal from which this command was entered. The current terminal must be enabled as an operator terminal.

/TEMPORARY

Designates the terminal at which the command is entered to be an operator's terminal for the current interactive session only. This qualifier is meaningful only when used with the */ENABLE* qualifier.

/TERMINAL=(terminal-name[,...])

Requires OPER privilege. Broadcasts the message to specified terminals, where the terminal-name keyword is the device name of the terminal. Incompatible with */ALL* and */USERNAME*.

/TO=identification-number

Requires OPER privilege. Sends a message to the user or file system specified by the identification number and completes the request. The current terminal must be enabled as an operator terminal.

Note that you can also use a variation of *REPLY/TO* in response to a *MOUNT/ASSIST* command where you redirect the mount operation to another device. Whenever you must substitute a device, load the user's volume on the alternate device and ready the device before entering the *REPLY* command. Use the following syntax:

REPLY/TO=identification-number "SUBSTITUTE device-name"

You can abbreviate the word *SUBSTITUTE* to *S* and use upper or lowercase characters. After a space, use the remainder of the message-text space to name the substituted device.

/URGENT

Sends a message beginning "URGENT..."; if used with the */BELL* qualifier, rings two bells at terminals receiving the message.

/USERNAME[=(username[,...])]

Requires OPER privilege. Broadcasts a message to all terminals at which users are logged in to the system (or VAXcluster), or only to the terminals of the specified users. Incompatible with */ALL* and */TERMINAL*.

/WAIT

Sends a message synchronously and then waits. The default is to send a message to *OPCOM*, which does the actual I/O. On a VAXcluster, the message is sent to the local node.

DCL-242 DCL Commands

REPLY

example

```
%OPCOM, 19-APR-1990 10:19:33.21, request 5, from user SYSTEM
OPA0, Please mount OPGUIDE on DBA3:
$ REPLY/PENDING=5 "YOU'LL HAVE TO WAIT-THERE ARE SEVERAL REQUESTS BEFORE YOURS"
.
.
.
REPLY/TO=5
19-APR-1990 10:20:25.50, request 5 completed by operator OPA0
```

In this example the OPCOM message indicates that a user wants the operator to place the disk volume labeled OPGUIDE on the disk drive DBA3 and ready the device. The REPLY/PENDING command indicates that the operator can perform the task but not immediately; the /PENDING qualifier prevents the user from entering other commands until the operator fulfills or aborts the request. After mounting the disk on the drive the operator sends a message indicating that the request has been fulfilled. When no message is specified, OPCOM sends a standard message indicating that the task has been performed.

REQUEST

Displays a message at a system operator's terminal and optionally requests a reply. All messages are logged at the operator's console and in the operator's log file, if that file is initialized.

To use this command, you must start the OPCOM process at boot time by specifying the DCL command @SYS\$SYSTEM:STARTUP OPCOM in the site-specific startup command file, SYS\$MANAGER:SYSTARTUP.COM.

format

REQUEST *"message-text"*

parameter

"message-text"

Specifies the text of the message to be displayed. The string can be up to 128 characters. If the string contains spaces, special characters, or lowercase characters, enclose it in quotation marks ("").

qualifiers

/REPLY

Requests a reply to the message and issues a unique identification number to which the operator sends the response. The system displays a message that the operator has been notified; you cannot enter any commands until the operator responds. If you press CTRL/C before the operator responds, you can then enter another message to the operator, or press CTRL/Z to cancel the request.

/TO=(operator[,...])

Specifies one or more operators to whom you want to send the message. Possible keywords are as follows:

CARDS	Sends the message to operators designated to respond to card reader requests
CENTRAL	Sends the message to the central system operator
CLUSTER	Sends the message to operators designated to respond to cluster-related requests
DEVICES	Sends the message to operators who mount and dismount disks
DISKS	Sends the message to operators who mount and dismount disk volumes
NETWORK	Sends the message to the network operator
OPER1 through OPER12	Sends the message to operators identified as OPER1 through OPER12
PRINTER	Sends the message to operators designated to handle print requests
SECURITY	Sends the message to operators designated to respond to security-related requests
TAPES	Sends the message to operators designated to mount and dismount tape volumes

example

```
$ REQUEST/REPLY "Are you there?"
%OPCOM-S-OPRNOTIF, operator notified, waiting...14:54:30.33
CTRL/C
REQUEST-Enter message or cancel request with ^Z
REQUEST-Message?CTRL/Z
%OPCOM-S-OPRNOTIF, operator notified, waiting... 14:59:01.38
%OPCOM-F-RQSTCAN, request was cancelled
```

In this example the REQUEST command issues a message and requests a response. When no operator replies to the question, CTRL/C is used to interrupt the request; then CTRL/Z is used to cancel it.

RETURN

Terminates a GOSUB subroutine procedure and returns control to the command following the calling GOSUB command.

format

RETURN *[status-code]*

parameter

status-code

Defines a longword (integer) value or expression equivalent to an integer value that gives the exit status of the subroutine by defining a numeric value for the reserved global symbol `$$STATUS`. The value can be tested by the next outer command level. The low-order three bits of the longword integer value change the value of the reserved global symbol `$$SEVERITY`.

If you do not specify a status-code, the current value of `$$STATUS` is saved. When control returns to the outer command level, `$$STATUS` contains the status of the most recently executed command or program.

The low-order three bits of the status value contained in `$$STATUS` represent the severity of the condition. The reserved global symbol `$$SEVERITY` contains this portion of the condition code. Severity values range from zero through four, as shown in the following table:

Value	Severity
0	Warning
1	Success
2	Error
3	Information
4	Severe (fatal) error

example

```
$ SHOW TIME
19-APR-1990 14:25:42
$ GOSUB SYMBOL
$ EXIT
$ SYMBOL:
$ SHOW SYMBOL RED
RED = "SET DEFAULT [JONES.DCL]"
$ RETURN 1
```

The `GOSUB` command transfers control to the subroutine labeled `SYMBOL`. After the subroutine is executed, the `RETURN` command transfers control back to the command following the calling `GOSUB` statement, giving `$$STATUS` and `$$SEVERITY` a value of 1. The procedure then exits.

RUN (Image)

Executes an image within the context of your process. You can abbreviate the RUN command to a single letter, **R**.

If you specify an image name in the command line with an explicit version number (or a semicolon), the image runs with current process privileges. If you do not specify an explicit version number (or semicolon), the image runs with any privileges with which it was installed. If you have DECnet software installed and want to execute an image over the network, you must have READ access to the file.

format

RUN *file-spec*

parameter

file-spec

Specifies an executable image to be executed. The file type defaults to EXE. Wildcard characters are not allowed.

qualifier

/DEBUG

/NODEBUG

Executes the image under control of the debugger. The default is /DEBUG if the image is linked with /DEBUG and /NODEBUG if the image is linked without /DEBUG. The /DEBUG qualifier is invalid if the image is linked with /NOTRACEBACK. The /NODEBUG qualifier overrides the effect of LINK/DEBUG. If the image was linked with /TRACEBACK, traceback reporting is performed when an error occurs.

example

```
$ RUN LIBRA
```

The image LIBRA.EXE starts executing in the process. If the image LIBRA has been installed with amplified privileges, it runs with those privileges because you have not explicitly specified a version number or a semicolon. Alternatively, the image LIBRA.EXE still runs with its amplified privileges, if you enter the RUN command as follows:

```
$ RUN LIBRA.EXE
```

RUN (Process)

Creates a subprocess or a detached process to run an image and deletes the process when the image completes execution. A subprocess is created if any of the qualifiers except `/UIC` or `/DETACHED` is specified. A detached process is created if the `/UIC` qualifier is specified and you have the `DETACH` user privilege.

format

RUN *file-spec*

parameter

file-spec

Specifies the file name of an executable image to be executed in a separate process. The default file type is `EXE`. Wildcard characters are not allowed in the file specification.

qualifiers

/ACCOUNTING (default)

/NOACCOUNTING

Requires ACNT privilege to disable accounting. Logs accounting records in the system accounting file for the created process.

/AST_LIMIT=quota

Specifies the maximum number of asynchronous system traps (ASTs) that the created process can have outstanding. If you do specify an AST limit quota, the default quota established at system generation time is used. The minimum required for any process to execute is 2. The AST limit quota is nondeductible.

/AUTHORIZE

/NOAUTHORIZE (default)

Requires DETACH privilege. When the image to be executed is the system login image (`LOGINOUT.EXE`), this qualifier searches the user authorization file to validate a detached process. The `/NOAUTHORIZE` qualifier creates a detached process that runs under the control of the command interpreter.

/BUFFER_LIMIT=quota

Specifies the maximum amount of memory, in bytes, that the process can use for buffered I/O operations or for temporary mailbox creation. If you do not specify a buffered I/O quota, the default value established at system generation time is used. The minimum amount required for any process to execute is 1024 bytes.

/DELAY=delta-time

Places the created process in hibernation and awakens it after a specified time interval. If you specify both */DELAY* and */INTERVAL*, the first wakeup request occurs at the time specified by */DELAY*. All subsequent wakeups occur at the interval specified by */INTERVAL*.

/DETACHED
/NODETACHED

Creates a detached process with the same user identification code (UIC) as the current process. (To create a detached process with a different UIC, use the */UIC* qualifier.) By default, the detached process has the same resource quotas as the current process; the *DETACH* privilege allows you to specify any quotas you need for the detached process. Unless you have the *DETACH* privilege, the maximum number of detached processes that you can create is limited to the quota defined by *MAX_DETACH* in your user authorization file.

/DUMP
/NODUMP (default)

When an image terminates because of an unhandled error, */DUMP* causes the contents of the address space to be written to the file named *SYS\$LOGIN:IMAGEDUMP.DMP*.

/ENQUEUE_LIMIT=quota

Specifies the maximum number of locks that a process can have outstanding at any one time. The default quota is that established at system generation time. The minimum required for any process to operate is 2.

/ERROR=file-spec

Defines an equivalence name string of 1 to 63 alphanumeric characters for the logical device name *SYS\$ERROR*. The logical name and equivalence name are placed in the process logical name table for the created process. (The */ERROR* qualifier is ignored if you are running *SYS\$SYSTEM:LOGINOUT*.)

/EXTENT=quota

Specifies the maximum size to which the image being executed in the process can increase its physical memory size. The default quota is that established at system generation time. The minimum value required for any process to execute is 10 pages. The extent quota is nondeductible.

/FILE_LIMIT=quota

Specifies the maximum number of files that a process can have open at any one time. The default quota is the quota established at system generation time. The minimum amount required for any process to execute is 2. The file limit quota is pooled.

DCL-248 DCL Commands
RUN (Process)

/INPUT=file-spec

Defines an equivalence name string of 1 to 63 characters for SYS\$INPUT. The logical name and equivalence name are placed in the process logical name table for the created process.

/INTERVAL=delta-time

Requests that the created process be placed in hibernation and awakened at regularly scheduled intervals. If you specify the /DELAY or /SCHEDULE qualifier with the /INTERVAL qualifier, the first wakeup occurs at the time specified by /DELAY or /SCHEDULE; all subsequent wakeups occur at intervals specified by /INTERVAL. If you specify neither /DELAY nor /SCHEDULE with /INTERVAL, the first wakeup occurs immediately by default.

/IO_BUFFERED=quota

Specifies the maximum number of system-buffered I/O operations that the created process can have outstanding at any one time. The default quota is the quota established at system generation time. The minimum required for any process to execute is 2. The buffered I/O quota is nondeductible.

/IO_DIRECT=quota

Specifies the maximum number of direct I/O operations that the created process can have outstanding at any one time. The default quota is the quota established at system generation time. The minimum required for any process to execute is 2. The direct I/O quota is nondeductible.

/JOB_TABLE_QUOTA=quota

Allows you to specify a quota for a detached process's jobwide logical name table. Note that the /JOB_TABLE_QUOTA qualifier is relevant only for detached processes. If the /JOB_TABLE_QUOTA qualifier is specified in a RUN command which results in the creation of a subprocess, it is ignored.

/MAILBOX=unit

Specifies the unit number of a mailbox to receive a termination message when the created process is deleted. If no mailbox is specified, the creating process receives no notification when the subprocess or detached process has been deleted.

/MAXIMUM_WORKING_SET=quota

Specifies the maximum size to which the image being executed in the process can increase its working set size. The default quota is the quota established at system generation time. The minimum value required for any process to execute is 10 pages. The maximum working set quota is nondeductible.

/OUTPUT=file-spec

Defines an equivalence name string of 1 to 63 characters for the logical device name SYS\$OUTPUT. Both the equivalence name and the logical name are placed in the process logical name table for the created process.

/PAGE_FILE=quota

Specifies the maximum number of pages that can be allocated in the paging file for the process. The default quota is the quota established at system generation time. The minimum value required for a process to execute is 256 pages. The paging file quota is pooled.

/PRIORITY=n

Requires ALTPRI privilege to set the priority higher than your current process. Specifies the base priority at which the created process executes. The value of *n* is a decimal number from 0 through 31. The default priority is that of the current process.

/PRIVILEGES=(privilege[,...])

Requires SETPRV privilege to specify privileges that you do not have. Defines user privileges for the created process. By default, the created process has the same privileges as its creator. If you specify only one privilege, you can omit the parentheses.

You can also use the keyword NOSAME as the privilege parameter. If you specify */PRIVILEGES=NOSAME*, the created process has no privileges.

If you specify a version number (or semicolon) in the file-spec parameter, the current process privileges are used, overriding any privileges specified with the */PRIVILEGES* qualifier.

/PROCESS_NAME=process-name

Specifies a name of 1 to 15 characters for the created process. The process name is implicitly qualified by the group number of the process's user identification code (UIC). By default, the name is null.

/QUEUE_LIMIT=quota

Specifies the maximum number of timer queue entries that the created process can have outstanding at any one time. The default quota is the quota established at system generation time. A process does not require any timer queue quota in order to execute. The timer queue entry quota is pooled.

/RESOURCE_WAIT (default)

/NORESOURCE_WAIT

Places the created process in a wait state when a resource required for a particular function is not available. If you specify */NORESOURCE_WAIT*, the process receives an error status code when a resource is unavailable.

/SCHEDULE=absolute-time

Places the created process in hibernation and awakens it at the specified time.

/SERVICE_FAILURE

/NOSERVICE_FAILURE (default)

Enables or disables an exception condition notification if an error occurs during a system service request. By default, an error status code is returned to the process.

/SUBPROCESS_LIMIT=quota

Specifies the maximum number of subprocesses that the created process is allowed to create. The default quota is the quota established at system generation time. A process does not require any subprocess quota in order to execute. The subprocess limit quota is pooled.

/SWAPPING (default)

/NOSWAPPING

Requires PSWAPM privilege to inhibit process swapping. Permits the process to be swapped. By default, a process may be swapped out of the balance set whenever it is in a wait state.

/TIME_LIMIT=limit

Specifies the maximum amount of CPU time (in delta time) a created process can use. CPU time is allocated to the created process in units of 10 milliseconds. When it has exhausted its CPU time limit quota, the created process is deleted.

If this quota is not specified and the created process is a detached process, the detached process receives a default value of 0, that is, unlimited CPU time.

If this quota is not specified and the created process is a subprocess, the subprocess receives half the CPU time limit quota of the creating process.

If this quota is specified as 0, the created process has unlimited CPU time providing that the creating process also has unlimited CPU time. If, however, the creating process does not have unlimited CPU time, the created process receives half the CPU time limit quota of the creating process.

The CPU time limit quota is a consumable quota; that is, the amount of CPU time used by the created process is not returned to the creating process when the created process is deleted.

/UIC=uic

Specifies that the created process be a detached process and assigns it a user identification code (UIC).

/WORKING_SET=default

Specifies the number of pages in the working set of the created process. The default working set size is the size established at system generation time. The minimum number of pages required for a process to execute is 10 pages. The value specified cannot be greater than the quota specified

with `/MAXIMUM_WORKING_SET`. The maximum working set quota is nondeductible.

example

```
$ RUN/INTERVAL=1:40/PROCESS_NAME=STAT  STATCHK  
%RUN-S-PROC_ID, identification of created process is 00050023
```

```
.  
. .  
. . .
```

```
$ CANCEL STAT
```

In this example, the `RUN` command creates a subprocess named `STAT` to execute the image `STATCHK.EXE`. The process is scheduled to execute the image at intervals of 1 hour and 40 minutes. The process hibernates; however, because neither the `/DELAY` nor `/SCHEDULE` qualifier is specified, the first wakeup occurs immediately.

The `CANCEL` command subsequently cancels the wakeup requests posted by the `/INTERVAL` qualifier. If the process is currently executing the image, it completes the execution and hibernates.

RUNOFF

Invokes the DIGITAL Standard Runoff (DSR) text formatter to format one or more ASCII files. Creates formatted files from source DSR (RNO) files, unformatted table of contents (RNT) files, and unformatted index (RNX) files. Optionally creates intermediate (BRN) files for input to `RUNOFF/CONTENTS` and `RUNOFF/INDEX` commands.

For more information about using the commands available within DSR, see the Reference Section. For information about the DCL commands `RUNOFF/CONTENTS` and `RUNOFF/INDEX`, see the DCL command descriptions.

format

```
RUNOFF file-spec[,...]
```

parameter

```
file-spec[,...]
```

Specifies one or more ASCII files (containing text and DSR commands) to be formatted by the `RUNOFF` command. The input file type defaults to `RNO`; you must specify the file type for `RNT` and `RNX` files. Separate multiple files with commas. Wildcard characters are not allowed in the file specification.

DSR produces an output file having the same file name as the input file. The output file type depends on the input file type. The default output file type is MEM. Specify SYS\$INPUT to type the input from your terminal or a command procedure; terminate input from the terminal by pressing CTRL/Z.

qualifiers

/BACKSPACE

Positional qualifier. Controls whether DSR uses the ASCII backspace character to perform character-by-character overprinting. By default, DSR performs line-by-line overprinting.

/BOLD[=n]

/NOBOLD

Positional qualifier. Specifies the number of times characters are overstruck in a bolding operation. You can specify the number of times DSR overprints flagged text by stating a value for n. N must be 0 or a positive integer and defaults to 1. A specification of /BOLD=0 or /NOBOLD disables all bolding, even if the appropriate flags are recognized and enabled.

/CHANGE_BARS[="character"]

/NOCHANGE_BARS

Positional qualifier. Controls whether DSR generates change bars in the formatted file. The default change-bar character is the vertical bar (|). The change bars appear 3 spaces to the left of the lines of text that you have marked for change bars. You can replace the default change-bar character by supplying a substitute character for the /CHANGE_BARS[="character"] qualifier. You must specify the replacement character as either a character enclosed in quotation marks or as an octal, decimal, or hexadecimal value for the desired character.

/DEBUG[=(option[,...])]

/NODEBUG (default)

Positional qualifier. Traces certain operations by placing the DSR commands in the output file. The options are as follows:

ALL	Specifies all five options (CONDITIONALS, CONTENTS, FILES, INDEX, and SAVE_RESTORE).
CONDITIONALS	Causes DSR to ignore all conditional processing commands (.IF, .IFNOT, .ELSE, .ENDIF) in the input file. DSR includes both "true" and "false" conditional information in the output file along with formatted text.
CONTENTS	Causes DSR to output all .SEND TOC commands along with the text being sent to the table of contents.

FILES	Causes DSR to output all .REQUIRE commands as well as the text of the require files.
INDEX	Causes DSR to output the indexing commands, .INDEX and .ENTRY, in addition to the text to which they refer.
SAVE_RESTORE	Causes DSR to output all .SAVE and .RESTORE commands.

If you specify more than one option, separate them with commas and enclose the list in parentheses. If you specify /DEBUG without specifying any options, ALL is assumed.

/DEVICE=(option[,...])

Positional qualifier. Controls whether DSR generates an output file (LNI) that is suitable for printing on an LN01, LN01E, or an LN03 laser printer. You can choose options from the following list to indicate output device, page orientation, and type of emphasis for flagged characters in your LNI file:

LN01	Produces an output file suitable for printing on an LN01 laser printer; the default paper size is 8 1/2 by 11 inches; the default mode is PORTRAIT. The output file name is the same as the input file name; the default file type is LNI.
LN01E	Produces an output file suitable for printing on an LN01E laser printer using the standard European paper size (A4). The output file name is the same as the input file name. The default file type is LNI; the default mode is PORTRAIT. Incompatible with LN01.
LN03	Produces an output file suitable for printing on an LN03 laser printer; the default paper size is 8 1/2 x 11 inches. The output file name is the same as the input file name. The default file type is LNI; the default mode is PORTRAIT.
LANDSCAPE	Causes the appropriate fonts for landscape mode to be loaded into the LN01; pages are printed with the long dimension at top using a smaller type size. (The page is 11 inches wide and 8 1/2 inches long.) Allowable page dimensions are 0 to 73 lines per page and 0 to 132 characters per line. Incompatible with PORTRAIT.
PORTRAIT (default)	Causes the appropriate fonts for portrait mode to be loaded into the LN01; pages are printed with the short dimension at top using a larger type size. (The page is 8 1/2 inches wide and 11 inches long.) Allowable page dimensions are 0 to 66 lines per page and 0 to 80 characters per line. Incompatible with LANDSCAPE.

RUNOFF

ITALIC (default)	Causes the italic and bold-italic fonts to be loaded into the LN01 printer. Italicizes characters flagged for underlining. Italicized characters can also be bolded depending on the type of emphasis you specify in your input file.
UNDERLINE	Causes the text and bold fonts to be loaded into the LN01. Underlines characters flagged for underlining. The LN01 allows only 63 consecutive characters (counting a space as a character) to be underlined per line. If you want to underline individual words and not the spaces between them, you will be able to underline only 63 words per line. Incompatible with ITALIC.

/DOWN[=n]

/NODOWN (default)

Positional qualifier. Controls whether DSR inserts a specified number of blank lines at the top of each page. These blank lines precede any header information. The number of blank lines you specify (n) does not affect the number of text lines on a page.

If you specify the /DOWN qualifier without a value, five blank lines are inserted. If you specify /DOWN=0 or omit the qualifier, no blank lines are inserted, except those associated with the print device or header layout.

/FORM_SIZE=n

Specifies the maximum number of lines per page including running heads and running feet. Defaults to /FORM_SIZE=66, which is standard for 11-inch paper. For laser printers, set the number of lines as follows:

Paper Size	Lines	Mode
8.05	69	Landscape
8.28	71	Landscape (LN01E default)
8.51	73	Landscape (LN01, LN03 default)
11.00	66	Portrait (LN01, LN03 default)
11.66	70	Portrait (LN01E default)
12.33	74	Portrait
13.00	78	Portrait
14.00	84	Portrait

/INTERMEDIATE[=file-spec]

/NOINTERMEDIATE (default)

Positional qualifier. Controls whether DSR generates an intermediate output file that can be used as input to the DSR table of contents utility and the DSR indexing utility. See the descriptions of RUNOFF/CONTENTS and RUNOFF/INDEX in the Reference Section for more information on producing tables of contents and indexes.

/LOG

/NOLOG (default)

Controls whether a termination message is displayed at the terminal after successful completion of the DSR operation. The message states the DSR version number, the number of diagnostic messages (if any), the number of output pages, and the output file specification.

/MESSAGES=(option[,...])

Positional qualifier. Specifies the destination of all DSR error messages. To indicate a specific destination, use one or both of the following options:

- | | |
|--------|---|
| OUTPUT | Messages are sent to the output MEM file |
| USER | Messages are displayed on the terminal (SYS\$ERROR) |

If you specify both options, separate them with commas and enclose the list in parentheses. The default, */MESSAGES=(OUTPUT,USER)*, sends messages to the output MEM file and displays them on the terminal.

/OUTPUT[=file-spec]

/NOOUTPUT

Positional qualifier. Specifies that an output file is to be produced and optionally names it. If you specify */OUTPUT* without a file specification, or if you omit the qualifier, the directory and file name default to that of the DSR file. If you specify */NOOUTPUT*, no output file is produced. The output file type depends on the input file type. The default input file type is RNO and the default output file type is MEM.

The file type defaults to one of the following:

- | | |
|-----|---|
| BLB | For an RNB input file |
| CCO | For an RNC input file |
| DOC | For an RND input file |
| ERR | For an RNE input file |
| HLP | For an RNH input file |
| LNI | For an RNO input file with <i>/DEVICE</i> set to LN01, LN01E, or LN03 |
| MAN | For an RNM input file |
| MEC | For an RNT input file |
| MEM | For an RNO input file with no <i>/DEVICE</i> specification |
| MEX | For an RNX input file |
| OPR | For an RNP input file |
| PLM | For an RNL input file |
| STD | For an RNS input file |

/PAGES=string

Positional qualifier. Specifies that only the pages within the specified range be generated as output. By default, DSR generates output for all pages. Specify the range as follows:

start-page-no:end-page-no,...

You can specify up to five ranges (*/PAGES="2-9:2-12, 4-1:4-10, 5-9:5-9, A-1:A-3, Index-1:Index-5"*). You can omit the colon and the end page number on the last range. You can omit the quotation marks if you specify only one range. Page numbers must be specified in their default form, not the form specified in a *.DISPLAY* command. You can specify just the appendix letter or name to produce an entire appendix. You can specify just the word *INDEX* to produce an entire index.

/PAUSE

/NOPAUSE (default)

Controls whether DSR pauses after printing each page of output. You can use the */PAUSE* qualifier to insert single sheets of paper or reproduction masters into hardcopy output devices. When output is halted, the terminal bell rings to remind you to insert a new form. Press the space bar to resume processing. Do not use this qualifier in a batch job.

/REVERSE_EMPHASIS

Positional qualifier. Directs DSR to change the order in which flagged text is underlined on an output device. If you use this qualifier, the printer first prints the characters to be underlined, then issues a carriage return without a line feed, and prints the underscores to underline the flagged text. If you view your file on the terminal, the flagged text is overwritten by the underline character.

/RIGHT[=n]

/NORIGHT (default except for LN01)

Positional qualifier. Causes the text on each page (including header information) to be shifted to the right the number of columns specified by *n*. This qualifier does not affect the page width. If you specify */RIGHT* without specifying a number, text is shifted to the right five spaces. If you specify a value of zero or omit the qualifier, no shift occurs.

The defaults (if */RIGHT* is not specified) for LN01 files are as follows:

Mode	LN01	LN01E	LN03
Landscape	9	13	9
Portrait	2	2	2

/SEPARATE_UNDERLINE[="character"]

Positional qualifier. Prints underlines as separate characters on the next line instead of overstriking with underscores on the same

line. The value specifies the character to be used for the underline character and defaults to a hyphen (-). You can specify the underline character as a single printable character or a number preceded by a radix indicator (%D, %O, or %X) to represent the ASCII value of a printable or nonprintable character. Incompatible with the /[NO]UNDERLINE_CHARACTER qualifiers.

/SEQUENCE

/NOSEQUENCE (default)

Positional qualifier. Controls whether DSR precedes the lines in the output file with the line numbers of the corresponding lines in the DSR file. For editors that generate line numbers in the input file, the /SEQUENCE qualifier causes similar numbering to appear in the output file. The line numbers appear in the left margin at the beginning of each line of output. If the text editor does not generate sequential numbers in the input file, sequential numbers are still generated in the output file, but without leading zeros.

/SIMULATE

/NOSIMULATE (default)

Controls whether DSR uses line feeds or form feeds to advance to the top of each page. For devices that do not have a form-feed capability, use /SIMULATE to generate enough blank lines to cause a skip to the top of each new page. The /SIMULATE qualifier also causes a pause before the first page of output. To continue after the pause, press the space bar.

/UNDERLINE_CHARACTER[="character"]

/NOUNDERLINE_CHARACTER

Positional qualifier. Specifies the character to be used for the underline character. Defaults to an underscore (_). You can specify the underline character as a single printable character (enclosed in quotation marks) or as a number preceded by a radix indicator (%D, %O, or %X) to represent the ASCII value of a printable or nonprintable character. A specification of /NOUNDERLINE_CHARACTER overrides any .ENABLE UNDERLINING command in the DSR file. Incompatible with /SEPARATE_UNDERLINE.

/VARIANT=string

Positional qualifier. Controls the processing of the conditional commands (.IF, .IFNOT, .ELSE, and .ENDIF) by specifying the names of the segments to be processed. You must name conditional structures introduced by .IF to process them. You must name conditional structures introduced by .IFNOT to exclude them. You must not name conditional structures introduced by .ELSE to process them. If you specify multiple names in a string, separate them by commas and enclose the string in quotation marks.

example

```
$ RUNOFF CHAPT1/RIGHT=10,CHAPT2
```

The **RUNOFF** command in this example produces a **CHAPT1.MEM** file with margins ten spaces to the right of the margins specified in the input file **CHAPT1.RNO**. It also generates a **CHAPT2.MEM** file whose margins are not affected by the **/RIGHT=10** qualifier.

RUNOFF/CONTENTS

Invokes the DIGITAL Standard Runoff (DSR) table of contents utility to create an RNT file that can be processed by DSR to make a table of contents. The input file for this command is an intermediate binary file (BRN) that is produced with the **RUNOFF** command and the **/INTERMEDIATE** qualifier (see the **RUNOFF** command).

For more information about using the DSR table of contents utility and the commands available within DSR, see the Reference Section.

format

RUNOFF/CONTENTS *file-spec[,...] or file-spec[+...]*

parameter

file-spec[,...] or file-spec[+...]

Specifies one or more intermediate binary files (BRN) that contain information (chapter titles, header levels, sections, and so on) for making a table of contents. To create a BRN file, use the **RUNOFF** command with the **/INTERMEDIATE** qualifier. If you omit the input file type, the DSR table of contents utility uses a default file type of BRN. **RUNOFF/CONTENTS** will also process BTC files that the previous version of DSR produced. For single input files, the table of contents utility produces an output file with the same file name as the input file. The output file type is RNT.

If you separate multiple input files with commas, separate RNT files for each input file are created. If you separate multiple input files with plus signs (+), a single RNT file that contains table of contents information for all of the input files is created. The default output file name is the same as the first input file name; the default file type is RNT. Wildcard characters are not allowed in the file specification.

qualifiers

/BOLD

/NOBOLD (default)

Controls whether the bolding specified in chapter and header titles in the input file appears in the table of contents.

/DEEPEST_HEADER=n

Controls how many levels of header levels are output in the table of contents. You can specify any number of header levels (up to 6) to be displayed by changing the value of n. The default is */DEEPEST_HEADER=6*.

/IDENTIFICATION

/NOIDENTIFICATION (default)

Controls whether the current version number of the DSR table of contents utility is reported.

/INDENT

/NOINDENT (default)

Controls how many spaces the header levels after level 1 are indented in the table of contents. If you omit this qualifier, or if you specify */NOINDENT*, all header levels after header level 1 are indented 2 spaces. If you specify */INDENT*, each header level after header level 1 is indented 2 spaces beyond the preceding header level.

/LOG

/NOLOG (default)

Controls whether the DSR table of contents utility displays the name of each input file as it is processed and after it is processed. The name of each output file created may also be displayed. If there are any errors in processing, the DSR table of contents utility sends messages to the terminal even if */NOLOG* is in effect.

/OUTPUT[=file-spec]

/NOOUTPUT

Specifies that an output file is to be produced and optionally names it. If you specify the */OUTPUT* qualifier without a file specification, or if you omit the qualifier entirely, the output file name matches the input file name. The default file type is RNT. The */NOOUTPUT* qualifier suppresses the creation of an output file. You can use */NOOUTPUT* to check an input file for errors without using system resources to generate an output file.

/PAGE_NUMBERS=(option[,...])

Controls whether the page number references in the table of contents are running page numbers or chapter-oriented page numbers; also controls how many levels of headers have page references listed in the table of contents. To specify these options, select from the following list:

DCL-260 DCL Commands
RUNOFF/CONTENTS

Option	Purpose
LEVEL= <i>n</i>	Specifies that header levels up to and including header level <i>n</i> have page numbers listed in the table of contents. The default is to display page numbers for 6 levels of headers.
NORUNNING	Specifies chapter-oriented page numbers (such as 1-3, 10-42). You can specify chapter-oriented numbers for the table of contents even if the document does not have chapter-oriented numbers. NORUNNING is the default.
RUNNING	Specifies running page numbers (such as 3, 42). You can specify running page numbers for the table of contents even if the document does not have running page numbers.

If you supply more than one option, separate them with commas and enclose the list in parentheses.

/REQUIRE=file-spec
/NOREQUIRE (default)

Allows you to change or delete the heading on the first page of a table of contents. The default heading is the word CONTENTS centered on the page and followed by one blank line. You can either substitute another word as a heading, or have no heading.

To change the heading, do one of the following:

- If you do not want any heading, specify a null file as the file specification for */REQUIRE*.

```
$ RUNOFF/CONTENTS/REQUIRE=nl:
```
- If you want to use a different heading, create or edit a file that specifies the heading that you want. Use the file that you create as the file specification for the */REQUIRE* qualifier.

/SECTION_NUMBERS (default)
/NOSECTION_NUMBERS

Controls whether the DSR table of contents utility displays section numbers in the table of contents. The */SECTION_NUMBERS* qualifier displays section numbers for all header levels in the table of contents. */NOSECTION_NUMBERS* suppresses the display of section numbers for all header levels.

/UNDERLINE
/NOUNDERLINE (default)

Controls whether the underlining specified in chapter and header titles in the input file appears in the table of contents.

example

```
$ RUNOFF/INTERMEDIATE CHPT1,CHPT2,CHPT3
```

Before using RUNOFF/CONTENTS, you must use RUNOFF/INTERMEDIATE to create a BRN file as input for the DSR table of contents utility. The command line in this example creates three separate files: CHPT1.BRN, CHPT2.BRN, and CHPT3.BRN.

RUNOFF/INDEX

Invokes the DIGITAL Standard Runoff (DSR) indexing utility to create an RNX file that can be processed by DSR to create an index. The input file for this command is an intermediate binary file (BRN) that is produced with the RUNOFF command and the /INTERMEDIATE qualifier (see the RUNOFF command). For more information about using the DSR indexing utility and the commands available within DSR, see the Reference Section.

format

RUNOFF/INDEX *file-spec[,...] or file-spec[+...]*

parameter

file-spec[,...] or file-spec[+...]

Specifies one or more intermediate binary files (BRN) that contain information (index entries, page number references, and so on) for making an index. To create a BRN file, use the RUNOFF command with the /INTERMEDIATE qualifier. See the RUNOFF command for more information on the /INTERMEDIATE qualifier.

If you omit the input file type, the DSR indexing utility uses a default file type of BRN. RUNOFF/INDEX also processes BIX files that the previous version of DSR produced. For single input files, the indexing utility produces an output file with the same file name as the input file. The output file type is RNX. If you separate multiple input files with commas, separate RNX files for each input file are created. If you separate multiple input files with plus signs (+), a single RNX file that contains indexing information for all of the input files is created. The default output file name is the same as the first input file name; the default file type is RNX. Wildcard characters are not allowed in the file specification.

qualifiers

/IDENTIFICATION

/NOIDENTIFICATION (*default*)

Reports the current version number of the DSR indexing utility.

/LINES_PER_PAGE=n

The value *n* specifies the number of lines of index entries on each page of the finished index. This number does not include the number of lines required for headings and footings. The default is 55 lines. This value is designed to work properly in the default formatting environment of DSR. You must calculate the value *n* if you change the default environment in any of the following ways:

- If you use subtitles in the document that requires the RNX file
- If you make the page length for the document anything other than 58 lines per page
- If you use any .LAYOUT other than zero (0)

To calculate the correct value for */LINES_PER_PAGE* use the following formula:

```
/LINES_PER_PAGE=n  
n = .PAGE SIZE ( the first parameter is length value)  
    minus 4 if subtitles are used, minus 3 if no subtitles  
    minus the number of lines reserved for .LAYOUT 1,  
        .LAYOUT 2, or .LAYOUT 3.
```

/LOG

/NOLOG (default)

Controls whether the DSR index utility displays the name of each input file as it is processed and after it is processed, as well as the name of each output file created. If there are any errors in processing, INDEX sends messages to the terminal even if */NOLOG* is in effect.

/OUTPUT[=file-spec]

/NOOUTPUT

Specifies that an output file is to be produced and optionally names it. If you specify the */OUTPUT* qualifier without a file specification, or if you omit the qualifier entirely, the output file name matches the input file name. The default file type is RNX. You can change the name of the output file by supplying a file specification for the value *file-spec*. The */NOOUTPUT* qualifier suppresses the creation of an output file. You can use */NOOUTPUT* to check an input file for errors without using system resources to generate an output file.

/PAGE_NUMBERS=option

Controls whether the page number references in the index are running page numbers or chapter-oriented page numbers. To specify the type of page numbers you want, select from the following options:

Option	Purpose
NORUNNING	Specifies chapter-oriented page numbers (such as 1-3, 10-42). You can specify chapter-oriented numbers for an index even if they do not appear in the document. NORUNNING is the default.
RUNNING	Specifies running page numbers (such as 1, 50, 230). You can specify running page numbers for an index even if the document does not display running page numbers.

/REQUIRE=file-spec***/NOREQUIRE (default)***

Allows you to change the heading on the first page of an index. The default heading is the word INDEX centered on the page and followed by three blank lines. The substitute heading is contained in the file you specify, which can contain DSR commands and text.

To change the heading:

1. Create or edit a file that specifies the format and the text that you want as the heading on the first index page.
2. Use the file you create as the file-spec for /REQUIRE.

See the /RESERVE qualifier for more information.

/RESERVE=n***/NORESERVE (default)***

Allows you to reserve space at the top of the first page of the index for text or white space that you want to include with the /REQUIRE=file-spec qualifier. Determine how many lines of text or white space you are adding to the top of the first page of the index. Use this number as the value n for the /RESERVE qualifier.

example

```
$ RUNOFF/INDEX/LINE_PER_PAGE=52 CHPT2
```

In this example, the RUNOFF/INDEX command takes the file CHPT2.BRN as input and creates CHPT2.RNX. The RNX file produces an index with 52 lines of index entries per page. The lines per page had to be adjusted because the writer used a page layout with the page numbers centered at the bottom of the page (.LAYOUT 1, .LAYOUT 2, .LAYOUT 3). This page layout takes up three more spaces than .LAYOUT 0, which is the default for DSR. To produce the final index, you must use the RNX file as input to DSR.

SEARCH

Searches one or more files for the specified string(s) and displays those lines containing that string or strings.

format

SEARCH *file-spec[,...]* *search-string[,...]*

parameters

file-spec[,...]

Specifies one or more files to be searched. You must specify at least one file name. If you specify two or more file names, separate them with commas. Wildcard characters are allowed in the file specification.

search-string[,...]

Specifies the character string to be located in the specified files. Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks.

qualifiers

/BACKUP

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */BACKUP* selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */CREATED*, */EXPIRED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: *TODAY* (default), *TOMORROW*, or *YESTERDAY*. Specify one of the following qualifiers with */BEFORE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

***/NOCONFIRM* (default)**

Controls whether a request is issued before each **SEARCH** operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXACT

/NOEXACT (default)

Controls whether the SEARCH command matches the search string exactly or treats uppercase and lowercase letters as equivalents. By default, SEARCH ignores case differences in letters.

/EXCLUDE=(file-spec,...)

Excludes the specified files from the SEARCH operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/FORMAT=option

Formats output in one of five ways:

DUMP	Displays all control characters (including <HT>, <CR>, and <LF>) and nonprintable characters as ANSI mnemonics.
NONULLS	Same as DUMP, but removes all null characters from the input file before reformatting.
NOFF	Replaces control characters in text with ANSI mnemonics (for example, CTRL/C is replaced with <ETX>). The terminal formatting characters <HT>, <CR>, <LF>, <VT> are passed without change. Form feed characters are replaced with <FF>.
PASSALL	Moves control and nonprintable characters to the output device without translating them. The terminal driver cannot send 8-bit characters to the terminal unless either SET TERMINAL/PASSALL or SET TERMINAL/EIGHT_BIT is already in effect.
TEXT	Replaces control characters in text with ANSI mnemonics (for example, CTRL/C is replaced with <ETX>). The terminal formatting characters <HT>, <CR>, <LF>, <VT>, and <FF> are passed without change. TEXT is the default format.

/HEADING (default)

/NOHEADING

Includes file names in the output file and displays a line of 30 asterisks as a window separator between groups of lines that belong to different files. With the default heading format, file names are printed only when more than one file is specified or when wildcard characters are used.

/HIGHLIGHT

/HIGHLIGHT(=option)

/HIGHLIGHT=BOLD (default on ANSI video terminals with advanced video)

/HIGHLIGHT=REVERSE (default on ANSI video terminals without advanced video)

/NOHIGHLIGHT (default for all other output)

Controls whether the actual strings that are matched are emphasized in the output. The emphasis, or highlighting, can be one of several options:

BLINK	The matched strings are highlighted using the ANSI blink character attribute (advanced video only).
BOLD	The matched strings are highlighted using the ANSI bold character attribute (advanced video only). If /HIGHLIGHT is used without an option, BOLD is assumed.
REVERSE	The matched strings are highlighted with the ANSI underline video attribute (possible without advanced video).

UNDERLINE

The matched strings are highlighted with the ANSI underline video attribute (possible without advanced video). Without the advanced video option, either REVERSE or UNDERLINE will appear depending on whether the cursor is selected as block or underline. The two options REVERSE and UNDERLINE have the same effect.

HARDCOPY(=option)

This specifies that the strings should be highlighted in a manner suitable for most hardcopy printers. Hardcopy highlighting has two options: OVERSTRIKE and UNDERLINE. With overstrike highlighting, matched strings are double-printed, so that they appear darker. The matched strings are underlined with the underscore character.

Hardcopy printing is accomplished by adding a carriage return and spacing back over the line to overprint the string or underlines. Note that this can as much as double the length of the line, and perhaps lead to truncation if the device buffer size is too small.

Digital recommends that you use /HIGHLIGHT=UNDERLINE with the Digital LN01 printer instead of /HIGHLIGHT=HARDCOPY=UNDERLINE. The LN01 ignores OVERSTRIKE highlighting.

Digital recommends that you use either /HIGHLIGHT=BOLD or /HIGHLIGHT=UNDERLINE with the DIGITAL LN03 printer instead of /HIGHLIGHT=HARDCOPY=UNDERLINE. The LN03 ignores OVERSTRIKE highlighting.

/LOG

/NOLOG (default)

Outputs a message to the current SYS\$OUTPUT device for each file searched. The message includes the file name, the number of records, and the number of matches for each file searched.

/MATCH=option

Interprets and matches multiple search strings in one of the following ways:

- AND A match occurs only if the record contains all the strings.
- NOR A match occurs only if the record contains none of the strings.
- NAND A match occurs only if the record does not contain all of the strings.
- OR A match occurs if the record contains any of the strings.

/MODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /EXPIRED. If you specify none of these four time modifiers, the default is /CREATED.

/NUMBERS

/NONUMBERS (default)

Controls whether the source line number is displayed at the left margin of each line in the output.

/OUTPUT[=file-spec]

/NOOUTPUT

Controls whether the results of the search are output to a specified file. The output is sent to the current default output device (SYS\$OUTPUT) if you omit the /OUTPUT qualifier or omit the file specification with the qualifier.

/REMAINING

/NOREMAINING (default)

Includes in the output all records from the first matched record to the end of the file. This qualifier overrides the value n in /WINDOW, but allows /WINDOW=n1.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with /BEFORE to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/STATISTICS

/NOSTATISTICS (default)

Controls whether the following statistics about the search are displayed:

- Number of files searched
- Number of records searched
- Number of characters searched
- Number of records matched
- Number of lines printed
- Buffered I/O count
- Direct I/O count
- Number of page faults
- Elapsed CPU time
- Elapsed time

/WINDOW[=(n1,n2)]
/NOWINDOW (default)

Specifies the number of lines to be displayed with the search string. If you specify the */WINDOW* qualifier without the value *n1* and *n2*, two lines above the search string, the search string, and the two lines below the search string are included in the output. If you specify */WINDOW* with a single number (*n1*), *n1* specifies the number of lines to display including the search string. Half the lines precede the matched search string and half follow it. (If *n* is even, 1 line is added to the lines following the matched search string.) If you specify *n1* and *n2*, the */WINDOW* qualifier displays *n1* lines above the search string, the search string, and *n2* lines below the search string. Either of these numbers can be zero. If you specify */WINDOW=0*, the file name of each file containing a match (but no records) is included in the output. If you omit the */WINDOW* qualifier, only the line containing a match is displayed.

example

```
$ SEARCH/OUTPUT=RESULTS.DAT/WINDOW=9 DISLIST.MEM NAME
```

The **SEARCH** command searches the file **DISLIST.MEM** for occurrences of the character string **NAME** and sends the output to the file **RESULTS.DAT**. The four lines preceding and following each occurrence of **NAME** are included in the output.

SET ACCOUNTING

Enables or disables the logging of various activities in the accounting log file **SYS\$MANAGER:ACCOUNTNG.DAT**. You can also use **SET ACCOUNTING** to close the current accounting log file and open a new one with a version number incremented by 1.

Requires OPER privilege.

format

SET ACCOUNTING

parameters

None.

qualifiers

/DISABLE[=(keyword[,...])]

Disables the logging of all activities in the accounting log file. To disable specific activities, you include one or more keywords with */DISABLE*. You can specify the following keywords:

DCL-270 DCL Commands

SET ACCOUNTING

Keyword	Function
BATCH	Inhibits/allows the recording of batch job termination
DETACHED	Inhibits/allows the recording of detached process termination
IMAGE	Inhibits/allows the recording of image activation
INTERACTIVE	Inhibits/allows the recording of interactive job termination
LOGIN_FAILURE	Inhibits/allows the recording of login failures
MESSAGE	Inhibits/allows the recording of user messages
NETWORK	Inhibits/allows the recording of network job termination
PRINT	Inhibits/allows the recording of all print jobs
PROCESS	Inhibits/allows the recording of all process termination
SUBPROCESS	Inhibits/allows the recording of all subprocess termination

/ENABLE[(keyword[,...])]

Enables the logging of all activities in the accounting file. To enable specific activities, you include one or more keywords with **/ENABLE**. Use the same keywords with **/ENABLE** that you use with **/DISABLE**.

/NEW_FILE

Closes the current accounting file and opens a new version of that file.

example

```
$ SET ACCOUNTING/ENABLE=(BATCH, INTERACTIVE)
```

The command in this example requests that all batch and interactive jobs be recorded in the accounting file at job termination.

SET ACL

Allows you to create or modify the access control list (ACL) of an object. Alternatively, you may use the VMS Access Control List (ACL) Editor to manipulate ACLs.

format

```
SET ACL object-name
```

parameter

object-name

Specifies the object whose access control list (ACL) is being modified. Wildcard characters are allowed in object names only if the object type is **FILE**. Each file must be a disk file on a Files-11 Structure Level 2 formatted volume. Logical name tables must be system logical name tables.

qualifiers

/ACL[=(ace[,...])]

Specifies one or more access control entries (ACEs) to be modified. When no ACE is specified, the entire access control list is affected. Separate multiple ACEs with commas. The specified ACEs are inserted at the top of the ACL unless the ***/AFTER*** qualifier is given. (Note that security alarm ACEs are always placed at the beginning of the ACL.)

/AFTER=ace

Indicates that all access control entries (ACEs) specified with the ***/ACL*** qualifier will be added after the ACE specified with the ***/AFTER*** qualifier. By default, any ACEs added to the ACL are always placed at the top of the list. (Note that security alarm ACEs are always placed at the beginning of the ACL.)

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

/NOCONFIRM (default)

Issues a request for confirmation before each modification. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL
	RET	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and the RETURN key. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

DCL-272 DCL Commands

SET ACL

/CREATED

Modifies the ACLs of files selected according to their creation date. Relevant only with the */BEFORE* and */SINCE* qualifiers.

/DEFAULT

Creates an ACL for the specified files as if the files were newly created. For a directory file, the */DEFAULT* qualifier propagates the entire ACL (except ACEs with the *NOPROPAGATE* option) so that a particular access protection can be propagated throughout a directory tree. For all other files, the */DEFAULT* qualifier propagates the *DEFAULT* option ACEs in the ACL of the parent directory to the ACL of the specified files.

/DELETE

Indicates that the access control entries (ACEs) specified with the */ACL* qualifier are to be deleted. If no ACEs are specified with */ACL*, the entire ACL is deleted (except those with the *PROTECTED* option).

/EDIT

Invokes the ACL Editor and allows you to use the */JOURNAL*, */MODE*, or */RECOVER* qualifiers. Any other qualifiers specified with */EDIT* are ignored.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the SET ACL operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/JOURNAL[=file-spec]

/NOJOURNAL

Controls whether a journal file is created from the editing session. By default, a journal file is created if the editing session ends abnormally. You must specify */EDIT* in order to use this qualifier.

/LIKE=(OBJECT_TYPE=type,OBJECT_NAME=name)

Deletes the ACL of the specified object and replaces it with the ACL of the object specified with */LIKE*.

You can specify the following keywords for *OBJECT_TYPE*: *DEVICE*, *FILE*, *SYSTEM_GLOBAL_SECTION*, *GROUP_GLOBAL_SECTION*, *QUEUE*, or *LOGICAL_NAME_TABLE*. This qualifier cannot be used with the */EDIT* qualifier.

/LOG

/NOLOG (default)

Controls whether the SET ACL command displays the object name of the object that has been affected by the command. This qualifier cannot be used with the */EDIT* qualifier.

/MODE=[NO]PROMPT

Determines whether the ACL editor prompts for field values. By default, the ACL editor selects prompt mode. You must specify the /EDIT qualifier to use this qualifier.

/NEW

Indicates that any existing ACE in the ACL of the object specified with SET ACL (except those with the PROTECTED option) is to be deleted. To use the /NEW qualifier, you must specify a new ACL or ACE with the /ACL, /LIKE, or /REPLACE qualifier. This qualifier cannot be used with the /EDIT qualifier.

/OBJECT_TYPE=type

Specifies the type of the object whose ACL is being edited. By default, the ACL editor assumes that the object whose ACL is being edited is a file. If the object is not a file, the /OBJECT qualifier is required. Possible keywords are as follows: FILE (includes directory files), DEVICE, SYSTEM_GLOBAL_SECTION, GROUP_GLOBAL_SECTION, QUEUE, or LOGICAL_NAME_TABLE.

/RECOVER[=file-spec]

/NORECOVER (default)

Specifies the name of the journal file to be used in a recovery operation. If the file specification is omitted with /RECOVER, the journal is assumed to have the same name as the input file and a file type of JOU. No wildcard characters are allowed with the /RECOVER file-spec parameter. You must specify /EDIT in order to use this qualifier.

/REPLACE=(ace[,...])

Deletes the access control entries (ACEs) specified with the /ACL qualifier and replaces them with those specified with /REPLACE. Any ACEs specified with the /ACL qualifier must exist and must be specified in the order in which they appear in the ACL. This qualifier cannot be used with the /EDIT qualifier.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

example

```
$ SET ACL/LIKE=(OBJECT_TYPE=FILE,OBJECT_NAME=USER.LIS) ACCOUNTS.LIS
```

This example replaces the ACL of the file ACCOUNTS.LIS with the ACL for the file USER.LIS.

SET AUDIT

Enables or disables security auditing on a VMS system. The SET AUDIT command is also used to modify the characteristics of the audit server process, set up long-term journaling (*archiving*) of audit events, and monitor resource consumption on the system. (Note that you must specify the /ALARM qualifier when enabling or disabling security auditing and when using the /FAILURE_MODE qualifier.)

Requires the SECURITY privilege.

format

SET AUDIT

event definition qualifiers

/ALARM

Causes alarm messages to be sent to all terminals enabled as security operators. See the description of the DCL command REPLY/ENABLE for details on how to enable terminals as security operators. The /ALARM qualifier is required when enabling or disabling security auditing with the /ENABLE or /DISABLE qualifiers, or when specifying a failure mode with the /FAILURE_MODE qualifier.

/DISABLE=(keyword[,...])

Disables security auditing for the specified events. To disable alarms for all events, specify the keyword ALL. You can also specify the appropriate keywords to selectively disable alarms for from one to all events that are currently enabled. You must specify at least one keyword. See the /ENABLE qualifier description for a list of the keywords to use with the /DISABLE qualifier.

/ENABLE=(keyword[,...])

Enables security auditing for the specified events. To enable alarms for all events, specify the keyword ALL. You can also specify the appropriate keywords to selectively enable alarms for from one to all events that are currently enabled. You must specify at least one keyword.

The possible events that may be specified in the keyword list of either the /ENABLE or /DISABLE qualifier are as follows:

ACL	An event requested by an access control list (ACL) item, including ACLs on files and global sections.
ALL	All possible events.

AUTHORIZATION

The modification of any portion of the system user authorization file (SYSUAF) or network proxy authorization file (NETPROXY), including any password changes; the modification of any portion of the rights database (RIGHTSLIST).

BREAKIN=(keyword[,...])

The occurrence of one or more of the following classes of break-in attempts, as specified by one or more of the keywords:

- ALL** All possible sources of break-ins, as defined by the remaining keywords
- DETACHED** Detached process break-in attempt
- DIALUP** Dialup break-in attempt
- LOCAL** Local break-in attempt
- NETWORK** Network server break-in attempt
- REMOTE** Remote break-in attempt

FILE_ACCESS=(keyword[,...])

The occurrence of file and global section access events (regardless of the value specified in the object's access control list, if any). You can specify one or more of the following keywords to describe the object access event to be noted.

- ALL** All types of object access events, as defined by the remaining keywords.
- BYPASS** Successful object access due to the use of the BYPASS privilege
[:access
[,access...]]
- FAILURE** Unsuccessful object access
[:access
[,access...]]
- GRPPRV** Successful object access due to the use of the GRPPRV privilege
[:access
[,access...]]
- READALL** Successful object access due to the use of the READALL privilege
[:access
[,access...]]
- SUCCESS** Successful object access
[:access
[,access...]]
- SYSPRV** Successful object access due to the use of the SYSPRV privilege
[:access
[,access...]]

**DCL-276 DCL Commands
SET AUDIT**

Most of the keywords permit you to define the type of object access that was obtained with the following keywords:

ALL	All types of object access events, as defined by the remaining keywords. If no access types are specified, ALL is assumed by the system.
READ	READ access
WRITE	WRITE access
EXECUTE	EXECUTE access
DELETE	DELETE access
CONTROL	Owner access

INSTALL

The occurrence of any INSTALL operations.

LOGFAILURE=(keyword[,...])

The occurrence of one or more of the following classes of login failure, as specified by one or more of the keywords:

ALL	All possible types of login failures, as defined by the remaining keywords
BATCH	Batch process login failure
DETACHED	Detached process login failure
DIALUP	Dialup interactive login failure
LOCAL	Local interactive login failure
NETWORK	Network server task login failure
REMOTE	Interactive login failure from another network node, for example, with a SET HOST command

SUBPROCESS Subprocess login failure

LOGIN=(keyword[,...])

The occurrence of one or more of the following classes of login attempts, as specified by one or more of the keywords:

ALL	All possible sources of logins, as defined by the remaining keywords
BATCH	Batch process login
DETACHED	Detached process login

SET AUDIT

	DIALUP	Dialup interactive login
	LOCAL	Local interactive login
	NETWORK	Network server task login
	REMOTE	Interactive login from another network node, for example, with a SET HOST command
	SUBPROCESS	Subprocess login
LOGOUT=(keyword[,...])		The occurrence of one or more of the following classes of logouts, as specified by one or more of the keywords:
	ALL	All possible sources of logouts, as defined by the remaining keywords
	BATCH	Batch process logout
	DETACHED	Detached process logout
	DIALUP	Dialup interactive process logout
	LOCAL	Local interactive process logout
	NETWORK	Logout by a network server task
	SUBPROCESS	Subprocess or detached process logout
	REMOTE	Logout of a process that logged in interactively from another network node
MOUNT		The issuance of a MOUNT or DISMOUNT request.

/FAILURE_MODE[=keyword]

Specifies how the VMS operating system proceeds following a failed attempt to write a security alarm to OPCOM's mailbox. Specify one of the following keywords with the /FAILURE_MODE qualifier:

Option	Description
WAIT	Indicates that processes are placed in the MWAIT state to wait until the resource is available. This is the default.
IGNORE	Indicates that failing security alarms are to be ignored. The first failed alarm causes an error message to be written to the operator console and log file. The system maintains a count of the lost alarms, which can be displayed with SHOW AUDIT.
CRASH	Forces a system failure if security alarms cannot be written.

The /ALARM qualifier is required when specifying an audit failure mode.

/VERIFY (default)***/NOVERIFY***

Specifies that control is not returned to the user (at the DCL command level) until the audit server has completed the request.

audit journal qualifiers

/DESTINATION=file-spec

Specifies the name and location of the security audit log file in the audit server database. The device, if part of the file specification, must be a disk volume. Because the system security log file is automatically created when the system is first installed and restored each time the system boots, this qualifier is only required when you want to move the log file.

Once you have updated the audit server database, execute the command SET AUDIT/SERVER=NEW_LOG to make the new location of the log file known to all audit server processes in the cluster. The previous audit log file is closed, and all subsequent audit event messages generated on the cluster are redirected to the new audit log file.

The /JOURNAL=SECURITY qualifier is required when redirecting the system security audit log file with the /DESTINATION qualifier.

/JOURNAL[=journal-name]

Specifies the name of the audit journal. The default, /JOURNAL=SECURITY, represents the system security audit log file, and is currently the only supported audit journal type. The /JOURNAL qualifier is required when changing the location of the audit log file with the /DESTINATION qualifier.

audit server qualifiers

/INTERVAL=(option-keyword[,...])

Specifies the delta times to be used for regular audit server operations.

In most cases, the defaults noted should be sufficient.

Option Keyword	Description
ARCHIVE_FLUSH=time	Specifies the period of time the audit server waits before flushing information to be archived. The default is 1 minute.
JOURNAL_FLUSH=time	Specifies the period of time the audit server waits before flushing information in the various audit journal buffers. The default is 5 minutes.

Option Keyword	Description
RESOURCE_SCAN=time	Specifies the period of time the audit server waits before monitoring the volume containing the audit journal for resource exhaustion. Resource exhaustion occurs when the volume has no free disk space. The default is 5 minutes.
RESUME_SCAN=time	Specifies the period of time the audit server waits before reviewing an existing resource exhaustion condition. The default is 15 minutes.

/LISTENER=device
/NOLISTENER

Specifies the name of a mailbox device which receives a copy of all security audit events. The user-defined mailbox can be used for processing of system security events as they occur, rather than logging events to the system security audit log file for inspection at a later time.

Specify the SET AUDIT/NOLISTENER command to remove a listener device from the system.

/SERVER=option-keyword[,...]

Specifies the audit server characteristics to be modified.

In most cases, the defaults noted should be sufficient.

Option Keyword	Description
CREATE_SYSTEM_LOG	Causes the audit server to create a new local system security audit log file. Other audit servers in the cluster are not affected. This keyword may be used by sites operating a multiple-environment cluster where it may be necessary to create a new log file on a specific node in the cluster. CREATE_SYSTEM_LOG is synonymous with NEW_LOG for nonclustered systems.
EXIT	Initiates an audit server shutdown. This is the only method for removing the audit server process from the system; the audit server cannot be deleted or suspended.

DCL-280 DCL Commands
SET AUDIT

Option Keyword	Description
FINAL_ACTION=action	Specifies the action taken by the audit server when resource exhaustion conditions have been met. Resource exhaustion occurs when the audit server attempts to buffer audit messages and runs out of virtual memory. (See the <i>Guide to VMS System Security</i> for more information about resource monitoring.) Specify one of the following values: CRASH Crash the system if the system runs out of virtual memory. This is the default. IGNORE_NEW Ignore new event messages until resources are available. Events messages leading up to the resource condition are saved; new messages are lost. PURGE_OLD Removes old event messages until resources are available in order to save the most current messages.
FLUSH	Copies all buffered audit and archive records to the audit log file and security archive file, respectively.
NEW_LOG	Creates a new clusterwide audit log file. The audit log file is created by the audit server process running on the local system and is opened by all audit servers in the cluster. (Typically, this is used daily to generate a new version of the audit log file.)
REDIRECT_SYSTEM_LOG	Causes the audit server on the local node to redirect security event messages to a new audit log file, whose location was previously defined by the /DESTINATION qualifier. Audit server processes (and log files) on other nodes in the cluster are unaffected.
RESUME	Requests the audit server process to resume normal activity on the system, if adequate disk space is available. Normally, once a resource monitoring <i>action</i> threshold has been reached, the audit server process suspends most system activity and waits 15 minutes before attempting to resume normal system activity.
START	Starts the audit server process on the system.

/VERIFY (default)

/NOVERIFY

Specifies that control is not returned to the user (at the DCL command level) until the audit server has completed the request.

archiving qualifiers

/ARCHIVE=option-keyword[,...]

Specifies the classes of audit messages events to be written to the security archive file. Specify one or more of the following keywords:

Option Keyword	Description
NONE	Disables archiving on the system. By default, archiving is disabled on the system.
[NO]ALL	Enables or disables archiving of all system security events.
SYSTEM_ALARM	Enables archiving of all system-generated alarm events.
SYSTEM_AUDIT	Enables archiving of all system-generated audit events. Reserved for future use.
USER_ALARM	Enables archiving of all user-generated alarm events. Reserved for future use.
USER_AUDIT	Enables archiving of all user-generated audit events. Reserved for future use.

/DESTINATION=file-spec

Specifies the name of the archive log file. Events may be archived to a local or remote file on any file-structured disk device. See the *Guide to VMS System Security* for information about creating a security archive file.

/VERIFY (default)

/NOVERIFY

Specifies that control is not returned to the user (at the DCL command level) until the audit server has completed the request.

resource monitoring qualifiers

/[NO]EXCLUDE=process-id

Adds a process ID (PID) to the audit server process exclusion list. The process exclusion list contains those processes which will not be suspended by the audit server process if a resource exhaustion reaches the *action* threshold. By default, the following processes are always contained in the process exclusion list and are never candidates for process suspension:

CACHE_SERVER
 CLUSTER_SERVER
 CONFIGURE
 JOB_CONTROL
 OPCOM

DCL-282 DCL Commands

SET AUDIT

SWAPPER
VWS\$DISPLAYMGR
VWS\$EMULATORS

Use the SET AUDIT/NOEXCLUDE=*process-id* command to remove a process from the process exclusion list. (PIDs are not automatically removed from the process exclusion list when processes log out from the system.)

/JOURNAL[=journal-name]

Specifies the name of the audit journal. The default, /JOURNAL=SECURITY, represents the system security audit log file, and is currently the only supported audit journal type. The /JOURNAL qualifier is required when specifying resource monitoring characteristics with the /RESOURCE or /THRESHOLD qualifiers.

/RESOURCE=option-keyword[,...]

Controls whether resource monitoring is in effect on the system, specifies the method used to monitor available resources, and determines the action the audit server will take if the available resources are depleted. The /JOURNAL qualifier is required when specifying resource monitoring with the /RESOURCE qualifier.

Option Keyword	Description								
DISABLE	Disables resource monitoring on the security audit log file.								
ENABLE	Enables resource monitoring on the security audit log file. By default, resource monitoring is enabled.								
MONITOR_MODE=mode	Specifies the method the audit server uses to monitor available resources. Specify one of the following keywords: <table><tbody><tr><td>COUNT</td><td>Controls whether resource monitoring is based on the amount of free disk space required to store a fixed number of event messages.</td></tr><tr><td>PERCENTAGE</td><td>Controls whether resource monitoring is based on the percentage of the disk volume or volume set available.</td></tr><tr><td>SPACE</td><td>Controls whether resource monitoring is based on the number of free blocks on the disk. This is the default method used for resource monitoring.</td></tr><tr><td>TIME</td><td>Controls whether resource monitoring is based on the amount of free disk space needed to store events which occur over a fixed period of time (in seconds).</td></tr></tbody></table>	COUNT	Controls whether resource monitoring is based on the amount of free disk space required to store a fixed number of event messages.	PERCENTAGE	Controls whether resource monitoring is based on the percentage of the disk volume or volume set available.	SPACE	Controls whether resource monitoring is based on the number of free blocks on the disk. This is the default method used for resource monitoring.	TIME	Controls whether resource monitoring is based on the amount of free disk space needed to store events which occur over a fixed period of time (in seconds).
COUNT	Controls whether resource monitoring is based on the amount of free disk space required to store a fixed number of event messages.								
PERCENTAGE	Controls whether resource monitoring is based on the percentage of the disk volume or volume set available.								
SPACE	Controls whether resource monitoring is based on the number of free blocks on the disk. This is the default method used for resource monitoring.								
TIME	Controls whether resource monitoring is based on the amount of free disk space needed to store events which occur over a fixed period of time (in seconds).								

SET AUDIT

/THRESHOLD=type

Specifies the thresholds the audit server uses for resource monitoring. The values which may be specified for each of the thresholds described depends on the mode of resource monitoring enabled on the system (see `/RESOURCE=MONITOR_MODE`). The `/JOURNAL` qualifier is required when modifying audit server thresholds with the `/THRESHOLD` qualifier.

Threshold Type	Meaning
WARNING=value	Specifies the threshold at which the audit server notifies all security operator terminals that resource exhaustion has occurred.
ACTION=value	Specifies the threshold at which the audit server suspends normal system activity.
RESUME=value	Specifies the threshold at which the audit server resumes normal system activity.

The following table lists the default warning, action, and resume thresholds for each resource monitor mode. Normally, the defaults listed should be sufficient.

Monitor Mode	Resource Monitoring Thresholds		
	WARNING	ACTION	RESUME
SPACE (blocks)	1000	250	750
PERCENTAGE (of volume)	1	0	1
COUNT (number of messages)	5000	1250	3750
TIME (seconds)	1000	250	750

/VERIFY (default)***/NOVERIFY***

Specifies that control is not returned to the user (at the DCL command level) until the audit server has completed the request.

example

```
$ SET AUDIT/ALARM/DISABLE=ALL
```

The SET AUDIT command in this example disables all security alarms and audit journal messages.

```
$ SET AUDIT/JOURNAL=SECURITY -
_$ /DESTINATION=AUDIT$: [AUDIT]SECURITY_AUDIT.LOG
$ SET AUDIT/SERVER=NEW_LOG
```

The first SET AUDIT command in this example updates the audit server database with the new name and location of the system security audit log file. The second command in the example causes all audit server processes in the cluster to open the new log file.

DCL-284 DCL Commands

SET AUDIT

```
$ SET AUDIT/ALARM/ENABLE=ALL/DISABLE=FILE:ALL
```

The SET AUDIT command in this example enables all classes of security events except file access alarms.

SET BROADCAST

Enables you to selectively screen out various kinds of messages from being broadcast to your terminal.

format

```
SET BROADCAST=(class-name[,...])
```

parameter

class-name

Specifies the class of message that you want to enable or disable for broadcast to your terminal. If you specify only one class, you can omit the parentheses. The class names are as follows:

ALL	All message classes enabled
[NO]DCL	CTRL/T and SPAWN/NOTIFY messages
[NO]GENERAL	All normal REPLY messages or messages from \$BRDCST
[NO]MAIL	Notification of mail
NONE	All message classes disabled
[NO]OPCOM	Messages issued by OPCOM
[NO]PHONE	Messages from the Phone Utility
[NO]QUEUE	Messages referring to print or batch jobs issued by the queue manager
[NO]SHUTDOWN	Messages issued from REPLY/SHUTDOWN
[NO]URGENT	Messages issued from REPLY/URGENT
[NO]USER1 - [NO]USER16	Messages from the specified user groups

example

```
$ SET BROADCAST=NONE
```

```
.  
. .  
. . .
```

```
$ SET BROADCAST=(SHUTDOWN, URGENT, DCL, OPCOM)
```


In this example, the first SET BROADCAST command screens out all messages. Later the second SET BROADCAST command restores shutdown, urgent, DCL, and OPCOM messages. General, phone, mail, queue, and user messages are still screened.

SET CARD_READER

Defines the default translation mode for cards read from a card reader. All subsequent input read from the specified card reader are converted using the specified mode.

format

SET CARD_READER *device-name[:]*

parameter

device-name[:]

Specifies the name of the card reader for which the translation mode is to be set. The device must not be currently allocated to any other user.

qualifiers

/026

Sets the card reader for cards punched on an 026 punch.

/029

Sets the card reader for cards punched on an 029 punch.

/LOG

/NOLOG (default)

Controls whether log information is displayed at the terminal to confirm that the card reader is set.

example

```
$ ALLOCATE CR:
  _CRA0: ALLOCATED
$ SET CARD_READER CRA0:/029
$ COPY CRA0: [MALCOLM.DATAFILES]CARDS.DAT
```

The ALLOCATE command requests the allocation of a card reader by specifying the generic device name. When the ALLOCATE command displays the name of the device, the SET CARD_READER command sets the translation mode at 029. Then the COPY command copies all the cards read by the card reader CRA0 into the file CARDS.DAT in the directory [MALCOLM.DATAFILES].

SET CLUSTER/EXPECTED_VOTES

Sets the total expected votes in the cluster to a value that you specify or, if no value is specified, sets the total votes to a value determined by the system.

Requires OPER privilege and the /EXPECTED_VOTES qualifier.

format

SET CLUSTER/EXPECTED_VOTES [=value]

example

```
$ SET CLUSTER/EXPECTED_VOTES=9
```

The SET CLUSTER command in this example sets the total expected votes to 9, which is the value specified in the command string.

SET COMMAND

Invokes the Command Definition Utility to add commands to your process command table or to a specified command table file.

format

SET COMMAND [file-spec[,...]]

SET CONTROL

Enables or disables CTRL/Y or CTRL/T. CTRL/Y interrupts a command and returns you to the DCL command level. CTRL/T momentarily interrupts a command to print a line of statistics.

SET CONTROL=T requires that SET TERMINAL/BROADCAST be set for the information to be displayed at your terminal.

format

SET [NO]CONTROL[=(T,Y)]

parameter

(T,Y)

Specifies that T (CTRL/T) or Y (CTRL/Y) be enabled or disabled. If you specify both characters, separate them with a comma and enclose the list in parentheses. If you do not specify either T or Y, Y is the default.

example

```
$ CTRL/T  
NODE22::SMITH 16:21:04 (DCL) CPU=00:03:29.39 PF=14802 IO=18652 MEM=68  
$ SET NOCONTROL=T  
$ CTRL/T
```

As shown in this example, when you press CTRL/T, the system displays the appropriate information. The SET NOCONTROL=T command disables the CTRL/T function. Now when you press CTRL/T, no information is displayed.

SET DAY

Sets the default day type specified in the user authorization file (UAF) for the current day.

Requires OPER privilege.

format

SET DAY

qualifiers

/DEFAULT

Overrides any previous SET DAY specification and specifies that the normal UAF defaults are to be used to determine today's day type.

/LOG

/NOLOG (default)

Controls whether log information is displayed at the terminal to confirm that the new SET DAY information has been set.

/PRIMARY

Sets today until midnight to a primary day.

/SECONDARY

Sets today until midnight to a secondary day.

example

```
$ SET DAY/PRIMARY
```

The SET DAY command in this example overrides the current default day type and sets the today until midnight to a primary day.

SET DEFAULT

Sets your default device and directory specifications. The new default is applied to all subsequent file specifications that do not explicitly include a device or directory name.

format

SET DEFAULT [*device-name[:]*][*directory-spec*]

parameters

device-name[:]

The name of the device you want to go to.

directory-spec

The name of the directory you want to go to. A directory name must be enclosed in brackets. Use the minus sign to specify the next higher directory from the current default.

You must specify either the *device-name* parameter or the *directory-spec* parameter. If you specify only the device name, the current directory is the default for the *directory-spec* parameter. If you specify only the directory name, the current device is the default for the *device-name* parameter.

example

```
$ SET DEFAULT $FLOPPY1: [WATER.MEMOS]
```

The **SET DEFAULT** command in this example sets your default to the WATER.MEMOS subdirectory on \$FLOPPY1.

SET DEVICE

Establishes a print device or terminal as a spooled device or establishes the operational status for a device.

Requires OPER privilege.

format

SET DEVICE *device-name[:]*

parameter

device-name[:]

Specifies the name of the device whose spooling or operational status is to change. The device must be a print device or a terminal if you want to change the spooling status; the device must be a disk or magnetic tape if you want to change the operational status.

qualifiers

/AVAILABLE
/NOAVAILABLE

Controls whether the specified disk or magnetic tape is to be considered available. You must dismount the specified disk or magnetic tape before entering the SET DEVICE/[NO]AVAILABLE command. If you specify /NOAVAILABLE, any attempt to allocate or mount the specified disk or magnetic tape is prevented.

/DUAL_PORT
/NODUAL_PORT

Controls whether the port seize logic in the device driver of the specified disk is to be enabled. This qualifier should be used only on disks that contain a dual port kit and have been dismounted.

/ERROR_LOGGING
/NOERROR_LOGGING

Controls whether device errors are logged in the error log file. When you specify the /ERROR_LOGGING qualifier, all error messages reported by the device on which error-logging is enabled are recorded in the error log file. Use the SHOW DEVICE/FULL command to find out the current status.

/LOG
/NOLOG (default)

Controls whether log information is displayed at the terminal.

/SPOOLED[=(queue-name[:],intermediate-disk-name[:])]
/NOSPOOLED

Controls whether files are spooled to an intermediate disk. The queue name indicates the printer queue to which a file is queued. If a queue name is not supplied, the default is the name of either the printer or terminal. The intermediate disk name identifies the disk to which the spooled files are written. If the intermediate disk name is not supplied, the default is SYS\$DISK (the current default disk). The intermediate disk must be mounted before files can be written to it.

example

```
$ SET DEVICE/SPOOLED=(LPA0) LPA0:
```

In this example, the /SPOOLED qualifier requests that the printer queue LPA0 be spooled to an intermediate disk before files directed to the disk are printed. Because no intermediate disk was specified, the intermediate disk defaults to SYS\$DISK.

SET DEVICE/SERVED

Allows you to make a disk on a local node available to all the nodes in a cluster. The /SERVED qualifier is required.

Applies only to VAXcluster environments.

format

SET DEVICE/SERVED *node-name*\$DDcu:

parameter

node-name\$DDcu:

Specifies the device name of the device that you want to make available to the cluster.

description

The SET DEVICE/SERVED command is used in conjunction with the Mass Storage Control Protocol (MSCP) server to make a disk on a local node available to all nodes on the cluster. The local node must be a member of a VAXcluster, and the local MSCP server must have been invoked by the SYSGEN Utility.

NOTE: Unless the disk device that you intend to make available to the cluster is a system disk, it must not already be mounted when you enter the SET DEVICE/SERVED command.

The SET DEVICE/SERVED command string can be included as part of the local startup command file, and entered before the Mount Utility mounts the disk to be served (made available to the entire cluster).

example

```
$ SET DEVICE/SERVED DRA4:
```

The SET DEVICE/SERVED command in this example instructs the MSCP server to make the disk device DRA4 on your local node available to all other processors on your cluster.

SET DIRECTORY

Modifies the characteristics of one or more directories.

See the qualifier descriptions for restrictions.

format

SET DIRECTORY [*device-name[:]*]*directory-spec*[,...]

parameters

device-name[:]

Specifies the device on which the directory that you want to modify is located. The device name parameter is optional.

***directory-spec*[,...]**

Specifies one or more directories to be modified. If you specify two or more directories, separate them with commas. Wildcard characters are allowed.

qualifiers

/BACKUP

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */BACKUP* selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */CREATED*, */EXPIRED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

***/BEFORE*[=*time*]**

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: *TODAY* (default), *TOMORROW*, or *YESTERDAY*. Specify one of the following qualifiers with */BEFORE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

***/BY_OWNER*[=*uic*]**

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

***/NOCONFIRM* (default)**

Controls whether a request is issued before each *SET DIRECTORY* operation to confirm that the operation should be performed on that file. The following responses are valid:

DCL-292 DCL Commands

SET DIRECTORY

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL

RET

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and RETURN. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(file-spec[,...])

Excludes the specified files from the SET DIRECTORY operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/LOG

/NOLOG (default)

Controls whether the system displays the directory specification of each directory that is modified as the command executes.

/MODIFIED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /MODIFIED selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP,

/CREATED, and **/EXPIRED**. If you specify none of these four time modifiers, the default is **/CREATED**.

/OWNER_UIC[=*uic*]

Requires **SYSPRV** privilege to specify a UIC other than your own. Specifies an owner UIC for the directory. The default UIC is that of the current process.

/SINCE[=*time*]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: **TODAY** (default), **TOMORROW**, or **YESTERDAY**. Specify one of the following qualifiers with **/BEFORE** to indicate the time attribute to be used as the basis for selection: **/BACKUP**, **/CREATED** (default), **/EXPIRED**, or **/MODIFIED**.

/VERSION_LIMIT[=*n*]

Specifies the total number of versions that a file in the specified directory can have. If you do not specify a version limit, a value of 0 is used, indicating that the number of versions of a file is limited only to the Files-11 architectural limit—32,767. If you change the version limit for the directory, the new value applies only to files created after the change has been made.

example

```
$ SET DIRECTORY/VERSION_LIMIT=5/CONFIRM [SMITH...]
```

The **SET DIRECTORY** command in this example sets a version limit of five for all files in the **SMITH** directory and all subdirectories of **[SMITH]**. The **/CONFIRM** qualifier requests that you confirm whether or not the specified directory should actually be modified. Note that it only affects the files created after the command is entered.

SET DISPLAY

Directs the output of a DECwindows application. Output can be directed from any VAX processor running a DECwindows application, including workstations, to any DECwindows workstation.

Both source and destination nodes must be part of the same network.

format

```
SET DISPLAY [display-device]
```

parameters

display-device

Specifies a logical name for the workstation display you are creating or modifying. If you are directing application output to multiple workstation displays, you can use different logical names to point to each display. If you do not specify a display-device string, the logical name DECW\$DISPLAY is used. This means that by default, application output will be displayed on the workstation display device referred to by DECW\$DISPLAY.

By entering the command SHOW DISPLAY, you can see the workstation node where applications will be displayed by default. If you specified your own logical name in the SET DISPLAY/CREATE command, include that logical name in the SHOW DISPLAY command.

qualifiers

/CREATE

Creates the workstation display device (WSAn:) on which a DECwindows application is displayed. You must specify the /CREATE qualifier the first time you use the SET DISPLAY command, but you need not respecify it if you continue to redirect output from applications to other workstations with subsequent SET DISPLAY commands.

When /CREATE is specified without /NODE, the workstation device defaults to the current node.

/[NO]PERMANENT

Cancels the redirected display by deassigning the logical name DECW\$DISPLAY. If you specified a logical name as the display-device parameter with the SET DISPLAY/CREATE command, entering the SET DISPLAY/NOPERMANENT display-device command cancels the redirected display by deassigning the logical name you specified.

The DECwindows Session Manager defines DECW\$DISPLAY in your job logical name table when you open a terminal (DECterm) window. When you redirect application output to another workstation with the SET DISPLAY/CREATE command, an additional DECW\$DISPLAY logical name is defined in your process logical name table. This definition supersedes the definition in the job logical name table. Output from applications run from the process in which you executed the SET DISPLAY/CREATE command will be displayed on the workstation referred to by the definition of DECW\$DISPLAY in the process logical name table. Enter the SHOW DISPLAY command to see where this application will be displayed. To see whether multiple definitions for DECW\$DISPLAY exist, enter the command SHOW LOGICAL DECW\$DISPLAY.

SET DISPLAY

If DECW\$DISPLAY is still defined (for example, in the job logical name table) after you specify the /NOPERMANENT qualifier, any DECwindows applications run from this process will be displayed on the workstation device to which output is now directed. Enter the SHOW DISPLAY command if you are unsure of the node to which DECW\$DISPLAY refers.

Use caution when entering the SET DISPLAY/NOPERMANENT command. If you modify or delete the definition of DECW\$DISPLAY from the job logical name table, you will be unable to start another session. Be careful not to specify the /NOPERMANENT qualifier without having first redirected the display with the SET DISPLAY/CREATE command.

You cannot specify /NOPERMANENT and /CREATE on the same command line.

/NODE=workstation_display

Defines the workstation on which you want to display DECwindows applications. The node name you provide cannot be a cluster alias (a name that represents multiple nodes configured in a VAXcluster), but must instead identify an actual node.

You must create a workstation display device with the /CREATE qualifier before you can redirect the output from applications to other workstations. Do not enter the SET DISPLAY/NODE=workstation_display command without having previously specified the /CREATE qualifier.

Make sure that you are authorized to display applications on the workstation you specify. See the *VMS DECwindows User's Guide* for more information about using the DECwindows Session Manager to authorize yourself to display applications from other nodes.

Each node, both source and destination, must be defined in each other's network node database. For example, to display applications on node HUBBUB from ZEPHYR, HUBBUB must be entered in ZEPHYR's network node database. ZEPHYR must be defined in HUBBUB's network node database. In addition, users on ZEPHYR must be authorized in the DECwindows Session Manager to display applications on HUBBUB. See the *VMS Networking Manual* and the *VMS Network Control Program Manual* for information about entering nodes in a network node database.

/TRANSPORT=transport-name

Defines the mechanism, for example, DECNET or LOCAL, that passes information between the application and the workstation. The transport mechanism is used to send input from the user to the application and output from the application to the display. If you specify the /CREATE qualifier, the default transport is DECNET.

Use the /TRANSPORT=LOCAL qualifier to optimize the performance of applications running and displaying on the same node.

DCL-296 DCL Commands

SET DISPLAY

example

```
$ SHOW DISPLAY
Device:      WSA1:
Node:        0
Transport:   LOCAL
Server:      0
Screen:      0

$ SET DISPLAY/CREATE/NODE=ZEPHYR
$ SHOW DISPLAY
Device:      WSA2:
Node:        ZEPHYR
Transport:   DECNET
Server:      0
Screen:      0

$ SPAWN/NOWAIT/INPUT=NL: RUN SYS$SYSTEM:DECW$CLOCK

$ SET DISPLAY/NOPERMANENT

$ SHOW DISPLAY
Device:      WSA1:
Node:        0
Transport:   LOCAL
Server:      0
Screen:      0
```

In this example, you are logged in to your workstation, here referred to as node 0. (0 is the standard shorthand notation for representing your node.) You want to run the DECwindows Clock on your workstation and display it on another workstation, ZEPHYR.

Assuming you are authorized to display applications on ZEPHYR, you redirect the application's output to ZEPHYR with the SET DISPLAY command and enter the SHOW DISPLAY command to verify the location of the redirected display. You then run Clock. When you finish running Clock, you disable the redirected display by entering the SET DISPLAY/NOPERMANENT command. Finally, you enter the SHOW DISPLAY command to verify that any applications subsequently run on your node will also be displayed there.

Note that a new workstation display device, WSA2, is created when you enter the SET DISPLAY/CREATE command. When you cancel the redirected display with the SET DISPLAY/NOPERMANENT command, application output is once again displayed on the workstation display device referred to by WSA1.

```
$ SET DISPLAY/CREATE/NODE=FLOPSY RABBIT
$ SHOW DISPLAY RABBIT

Device:      WSA2:
Node:        FLOPSY
Transport:   DECNET
Server:      0
Screen:      0
```

```
$ RUN/DETACHED/OUTPUT=WSA2: SYSS$SYSTEM:DECW$CLOCK
$ SET DISPLAY/CREATE/NODE=ZEPHYR ZNODE
$ SHOW DISPLAY ZNODE

Device:     WSA3:
Node:       ZEPHYR
Transport:  DECNET
Server:     0
Screen:     0

$ RUN/DETACHED/OUTPUT=WSA3: SYSS$SYSTEM:DECW$CALENDAR
$ RUN SYSS$SYSTEM:DECW$BOOKREADER
$ SHOW DISPLAY

Device:     WSA1:
Node:       0
Transport:  LOCAL
Server:     0
Screen:     0
```

In this example, you are logged in to your node, and want to direct the output from applications to several workstation displays in the same session. By specifying different logical names in the SET DISPLAY command, you can redirect the output without changing the logical name definition for DECW\$DISPLAY. This allows you to display the output from most applications on your default display but occasionally display output on another workstation. You can also continue to run and display applications on your node. In this example, Clock is displayed on node FLOPSY, Calendar is displayed on node ZEPHYR, and Bookreader is displayed on your workstation.

Note that to run your applications with the DCL command RUN/DETACHED, you must use the device name that equates to the logical display device name you specified in the SET DISPLAY command. Use the SHOW DISPLAY command to obtain this device name.

SET ENTRY

Changes the current status or attributes of a job that is not currently executing in a queue.

Requires OPER privilege, EXECUTE (E) access to the queue, or DELETE (D) access to the specified jobs.

format

SET ENTRY *entry-number* [...]

parameter

entry-number[,...]

Specifies the **entry number** (or a list of entry numbers) of the jobs you want to change.

The system assigns a unique entry number to each queued print or batch job in the system. By default, the PRINT and SUBMIT commands display the entry number when they successfully queue a job for processing. These commands also create or update the local symbol \$ENTRY to reflect the entry number of the most recently queued job. To find a job's entry number, enter the SHOW ENTRY or SHOW QUEUE command.

qualifiers

/AFTER=time

/NOAFTER

Requests that the specified job be held until after a specific time. If the specified time has already passed, the job is queued for immediate processing. You can specify either an absolute time or a combination of absolute and delta times.

/BURST

/NOBURST

Controls whether two file flag pages with a burst bar between them are printed preceding each file in a job.

/CHARACTERISTICS=(characteristic[,...])

/NOCHARACTERISTICS

Specifies the name or number of one or more characteristics to be associated with the job. Characteristics can refer to such things as color of ink. If you specify only one characteristic, you can omit the parentheses.

/CLI=filename

Specifies the name of a command language interpreter (CLI) to use in processing the batch job. The file name specifies that the CLI be SYS\$SYSTEM:filename.EXE. If you do not specify the /CLI qualifier, the job is run by the CLI specified in the user authorization file (UAF), or whatever CLI was specified when the job was originally submitted to the queue.

/COPIES=n

Specifies the number of copies to print. The value of *n* can be any number from 1 to 255. When you use the /COPIES qualifier with the SET ENTRY command, the number of copies can apply only to the entire print job. You cannot use this qualifier to specify different numbers of copies for individual files within a multifile job.

/CPUTIME=time

Specifies a CPU time limit for the batch job. You can specify time as delta time, 0, INFINITE, or NONE.

/FEED
/NOFEED

Controls whether form feeds are inserted into the print job when the printer reaches the bottom margin of the form in use. You can suppress this automatic form feed (without affecting any of the other carriage control functions that are in place) by using the */NOFEED* qualifier. When you use the */FEED* qualifier with the SET ENTRY command, the qualifier applies to all files in the print job. You cannot use this qualifier to specify form feeds for individual files within a multifile job.

/FLAG
/NOFLAG

Controls whether a flag page is printed preceding each file in a print job. The flag page contains the name of the user submitting the job, the job entry number, and other information about the file being printed.

/FORM=form

Specifies the name or number of the form to be associated with the print job. If you omit the */FORM* qualifier, the default form for the execution queue is associated with the job. To see which forms have been defined for your system, use the SHOW QUEUE/FORM command.

/HEADER
/NOHEADER

Controls whether a heading line is printed at the top of each output page in a print job.

/HOLD
/NOHOLD

Controls whether the job is to be made available for immediate processing or held for processing later. If you specify */HOLD*, the job is not released for processing until you enter SET ENTRY/NOHOLD or SET ENTRY/RELEASE. You can use the SET ENTRY command to release a job that was previously submitted with a */HOLD* qualifier, or you can place a job on hold so that it will run later.

/JOB_COUNT=n

Requests that an entire print job be printed *n* times, where *n* is a decimal integer from 1 to 255. This qualifier overrides the */JOB_COUNT* qualifier with the PRINT command.

/KEEP
/NOKEEP

Controls whether the batch job log file is deleted after it is printed.

DCL-300 DCL Commands
SET ENTRY

/LOG_FILE[=file-spec]
/NOLOG_FILE

Creates a log file with the specified file specification. You can specify a different device name, as long as the process executing the batch job has access to the device on which the log file will reside. Logical names in the file specification are translated in the context of the process that executes the SET ENTRY command. If you omit the /LOG_FILE qualifier and specify the /NAME qualifier, the log file is written to a file having the same file name as that specified by the /NAME qualifier; the file type is LOG. When you omit the /LOG_FILE qualifier, the job-name value used with /NAME must be a valid file name.

/LOWERCASE
/NOLOWERCASE

Indicates whether the print job must be printed on a printer that can print both uppercase and lowercase letters. The /NOLOWERCASE qualifier means that files can be printed on printers that print only uppercase letters. If all available printers can print both uppercase and lowercase letters, you do not need to specify /LOWERCASE.

/NAME=job-name

Names the job. The job name must be 1 to 39 alphanumeric characters. The SHOW ENTRY and SHOW QUEUE commands display the job name. For batch jobs, the job name is also used for the batch job log file. For print jobs, the job name is also used on the flag page of the printed output. The default job name is the name of the first file in the job.

/NOCHECKPOINT

For a batch job, erases the value established by the most recently executed SET RESTART_VALUE command. For a print job, clears the stored checkpoint so that the job will restart from the beginning.

/NODELETE

Cancels file deletion for a job that was submitted with the /DELETE qualifier. If you did not specify the /DELETE qualifier when the job was originally submitted to the queue, you cannot use the SET ENTRY command to establish file deletion at a later time. You cannot use the /NODELETE qualifier to cancel deletion of individual files in a multifile job.

/NOTE=string

Specifies a message of up to 255 characters to appear on the flag page of the print job. Enclose messages containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (" ").

/NOTIFY
/NONOTIFY

Controls whether a message notifies you when your job has been completed or aborted. Notification is sent to any terminal session on the same cluster at which you are logged in.

/OPERATOR=string

Specifies a message string of up to 255 characters to be sent to the operator just before the print job begins to print. Enclose the message in quotation marks (" ") if it contains spaces, special characters, or lowercase characters.

/PAGES=(*[lowlim,]uplim*)

Specifies the number of pages to print for the specified job. You can use the /PAGES qualifier to print portions of long files. By default, all pages of the file are printed. When you use the /PAGES qualifier with the SET ENTRY command, the qualifier can apply only to an entire job. You cannot use this qualifier to specify different numbers of pages to be printed for individual files within a multifile job. The lowlim specifier refers to the first page of the file that you want to print. If you omit the lowlim specifier, the printing starts on the first page of the file.

The uplim specifier refers to the last page of the file that you want to print.

/PARAMETERS=(*parameter[,...]*)

Specifies from one to eight optional parameters to be passed to the job. Each parameter can have as many as 255 characters. If you specify only one parameter, you can omit the parentheses. The commas delimit individual parameters. To specify a parameter that contains any special characters or delimiters, enclose the parameter in quotation marks. For batch jobs, the parameters define values to be equated to the symbols named P1 through P8 in each command procedure in the job. The symbols are local to the specified command procedures.

/PASSALL
/NOPASSALL

Specifies whether the symbiont bypasses all formatting of the print job and sends the output QIO to the driver with format suppressed. All qualifiers affecting formatting, as well as the /HEADER, /PAGES, and /PAGE_SETUP qualifiers, are ignored. When you use the /PASSALL qualifier with the SET ENTRY command, the qualifier applies to the entire job. You cannot use this qualifier to specify PASSALL mode for individual files within a multifile job.

/PRINTER[=*queue-name*]
/NOPRINTER

Queues the batch job log file for printing when the job is completed. The default output queue for the log file is SYS\$PRINT. The /PRINTER

DCL-302 DCL Commands
SET ENTRY

qualifier allows you to specify an output queue. The /NOPRINTER qualifier assumes the /KEEP qualifier.

/PRIORITY=*n*

Requires OPER or ALTPRI privilege to raise the priority above the value of the SYSGEN parameter MAXQUEPRI. Specifies the job-scheduling priority of the job. The value of *n* is an integer in the range of 0 through 255, where 0 is the lowest priority and 255 is the highest.

The default value for /PRIORITY is the value of the SYSGEN parameter DEFQUEPRI. No privilege is needed to set the priority lower than the MAXQUEPRI value.

/RELEASE

Releases for processing jobs submitted with the /HOLD qualifier or /AFTER qualifier, jobs held in a queue with the /RETAIN qualifier, and jobs refused by a user-written symbiont.

/REQUEUE=*queue-name*[:]

Requests that the job be moved from the original queue to the specified queue.

/RESTART

/NORESTART

Specifies whether a batch or print job is restarted after a system failure or a STOP/QUEUE/REQUEUE command.

/SETUP=*module*[,...]

Extracts the specified modules from the device control library (containing escape sequence modules for programmable printers) and copies the modules to the printer before each file in a print job is printed. When you use the /SETUP qualifier with the SET ENTRY command, the qualifier applies to the entire print job. You cannot use this qualifier to specify different setup modules for individual files within a multifile job.

/SPACE

/NOSPACE

Controls whether the output of a print job is double-spaced. Specifying /NOSPACE causes the output to be single-spaced. When you use the /SPACE qualifier with the SET ENTRY command, the qualifier applies to the entire job. You cannot use this qualifier to specify different spacing for individual files within a multifile job.

/TRAILER

/NOTRAILER

Controls whether a trailer page is printed at the end of each file in a print job. The trailer page displays the entry number, as well as information about the user submitting the job and the files being printed. When you use the /TRAILER qualifier with the SET ENTRY command, trailer pages are placed at the end of each file in a multifile job.

/WSDEFAULT=n

Defines for a batch job a working set default, the default number of physical pages that the job can use. If the queue on which the job executes has a nonzero default working set, the smaller of the specified job and queue values is used. If the queue on which the job executes has a working set default of 0, the smaller of the specified job value and the value established in the user authorization file (UAF) is used. If you specify 0 or NONE, the specified queue or UAF value is used. Working set default values must range between the numbers specified by the SYSGEN parameters PQL_MWSDEFAULT and WSMAX.

/WSEXTENT=n

Defines for the batch job a working set extent, the maximum amount of physical memory that the job can use. The job uses the maximum amount of physical memory only when the system has excess free pages. If the queue on which the job executes has a nonzero working set extent, the smaller of the specified job and queue values is used. If the queue on which the job executes has a working set extent of 0, the smaller of the specified job value and the value established in the user authorization file (UAF) is used. If you specify 0 or NONE, the specified queue or UAF value is used. Working set extent values must range between the numbers specified by the SYSGEN parameters PQL_MWSEXTENT and WSMAX.

/WSQUOTA=n

Defines for the batch job a working set quota, the amount of physical memory that the job is guaranteed. If the queue on which the job executes has a nonzero working set quota, the smaller of the specified job and queue values is used. If the queue on which the job executes has a working set quota of 0, the smaller of the specified job value or the value established in the user authorization file (UAF) is used. If you specify 0 or NONE, the specified queue or UAF value is used. Working set quota values must range between the numbers specified by the SYSGEN parameters PQL_MWSQUOTA and WSMAX.

example

```
$ PRINT/HOLD MYFILE.DAT
  Job MYFILE (queue SYS$PRINT, entry 112) holding
$ SET ENTRY 112/RELEASE/JOB_COUNT=3
```

The PRINT command in this example requests that the file MYFILE.DAT be queued to the system printer, but placed in a hold status. The SET ENTRY command releases the job for printing and requests that three copies of the job be printed.

SET FILE

Modifies the characteristics of one or more files.

See the **qualifier descriptions for restrictions**.

format

SET FILE *file-spec[,...]*

parameter

file-spec[,...]

Specifies one or more files to be modified. If you specify two or more files, separate them with commas. Wildcard characters are allowed.

qualifiers

/ACL

Modifies an access control list (ACL) associated with one or more files. For more information, see the description of the SET FILE/ACL command.

/BACKUP

/NOBACKUP

Specifies that BACKUP records the contents of the file. The /NOBACKUP qualifier causes BACKUP to record the attributes of the file but not its contents. Valid only for Files-11 Structure Level 2 files.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

/NOCONFIRM (default)

Controls whether a request is issued before each SET FILE operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL

RET

You can use any combination of upper- and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CREATED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation.

/DATA_CHECK=[([NO]READ,[NO]WRITE)]

Specifies whether a READ data check (rereading each record), a WRITE data check (reading each record after it is written), or a combination of the two is performed on the file during transfers. By default, a WRITE data check is performed.

/END_OF_FILE

Resets the end-of-file mark to the highest block allocated.

/ENTER=new-file-spec

Use with caution. Assigns an additional name to a single file so that the file has a second name, or alias. However, both the original name and the alias reference the same file. For this reason, take care when deleting files that have aliases. To keep the file but remove one of its names, use the /REMOVE qualifier with SET FILE. No wildcards are allowed in the file specification.

/ERASE_ON_DELETE

Specifies that the specified files are erased from the disk (not just merely written over) when the DELETE or PURGE command is issued for the files. See DELETE/ERASE for more information.

/EXCLUDE=(file-spec[,...])

Excludes the specified file from the SET FILE operation. You can include a directory name but not a device name in the file specifications. Wildcard characters are supported for file specifications. However, you cannot use relative version numbers to exclude a specific version. If you specify only one file, you can omit the parentheses.

/EXPIRATION_DATE=date

/NOEXPIRATION_DATE

Requires ownership of the file or access control. Controls whether an expiration date is assigned to the specified files. Absolute date keywords are allowed. If you specify 0 as the date, today's date is used.

/EXTENSION[=*n*]

Sets the extend quantity default for the file. The value of *n* can range from 0 through 65,535. If you omit the value specification or specify a value of 0, VMS RMS calculates its own */EXTENSION* value.

/GLOBAL_BUFFER=*n*

Sets the VAX RMS global buffer count (the number of buffers that can be shared by processes accessing the file) for the specified files. The value *n* must be an integer in the range 0 through 32,767. A value of 0 disables buffer sharing.

/LOG

/NOLOG (default)

Displays the file specification of each file modified as the command executes.

/MODIFIED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with */CREATED*, which also allows you to select files according to time attributes. If you do not specify */MODIFIED*, the default is */CREATED*.

/NODIRECTORY

Use with extreme caution. This qualifier removes the directory attributes of a file and allows you to delete the corrupted directory file even if other files are contained in the directory. When you delete a corrupted directory file, the files contained within it are lost. Use *ANALYZE/DISK_STRUCTURE/REPAIR* to place the lost files in [SYSLOST]. You can then copy the lost files to a new directory. This qualifier is valid only for the Files-11 Structure Level 2 files.

/OWNER_UIC[=*uic*]

Requires GRPPRV to set the owner to another member of the same group. Requires SYSPRV to set the owner to any UIC outside your group. Specifies an owner user identification code (UIC) for the file. The default is the UIC of your process.

/PROTECTION[=(*code*)]

Cannot be used to change the protection on a file via DECnet.

Enables you to change or reset the protection for one or more of your files. The ownership categories are SYSTEM, OWNER, GROUP, AND WORLD. The access categories are R (read), W (write), E (execute), and D (delete). If you specify */PROTECTION* without the ownership and access code, the file protection is set according to the current default protection.

/REMOVE

Use with caution. This qualifier enables you to remove one of the names of a file that has more than one name, without deleting the file. If you have created an additional name for a file with the */ENTER* qualifier of SET FILE, you can use the */REMOVE* qualifier to remove either the original name or the alias.

/SEMANTICS=semantics-tag

/NOSEMANTICS

Use */SEMANTICS* to create or change a semantics tag.

Use */NOSEMANTICS* to remove a semantics tag from a file.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: TODAY (default), TOMORROW, or YESTERDAY.

/STATISTICS

/NOSTATISTICS (default)

Enables the gathering of RMS statistics on the specified file. These statistics can subsequently be viewed using the Monitor Utility, which is invoked with the DCL command MONITOR.

/TRUNCATE

Truncates the file at the end of the block containing the end-of-file (EOF) marker, that is, releases allocated but unused blocks of the file.

/UNLOCK

Makes one or more improperly closed files accessible.

/VERSION_LIMIT[=n]

Specifies the maximum number of versions for the specified file. If you do not specify a version limit, a value of 0 is used, indicating that the number of versions of a file is limited only to the Files-11 architectural limit of 32,767. When you exceed that limit, the earliest version of the file is deleted from the directory without notification to the user.

example

```
$ SET FILE/EXPIRATION_DATE=19-APR-1990:11:00 BATCH.COM;3
```

The SET FILE command requests that the expiration date of the file BATCH.COM;3 be set to 11:00 a.m., April 19, 1990.

SET HOST

Connects your terminal (through the current host processor) to another processor, called the remote processor. Both processors must be running DECnet.¹

- You can use the SET HOST command only if your system is connected by DECnet to another system.
- You must have an account on the remote system to log in after the SET HOST command has made the connection.
- The SET HOST command requires the network mailbox privilege NETMBX.

format

SET HOST *node-name*

parameter

node-name

Specifies the node name of the remote processor to which you will connect.

qualifiers

/BUFFER_SIZE=*n*

Changes the packet size of the protocol message sent between the terminal and the remote processor if a connection to the remote processor is already established. The default buffer size is 1010 bytes; however, *n* can range from 140 bytes to 1024 bytes. The value of *n* is reset to 140 bytes if a value below 140 is specified; a value for *n* above 1024 bytes is reset to 1024.

/LOG[=*file-spec*]

/NOLOG (default)

Controls whether a log file of the entire session is kept. If you use /LOG without the file specification, the log information is stored in the file SETHOST.LOG.

/RESTORE

/NORESTORE

Saves current terminal characteristics before a remote terminal session is begun and restores them when the remote session is terminated.

¹ Available under separate license.

example

```
$ SET HOST ITALIC
Username: BROWN
Password:
Welcome to VAX/VMS Version 5.0 on node ITALIC
```

.
.
.

```
$ LOGOUT
BROWN logged out at 19-APR-1990 15:04:25.27
%REM-S-END, Control returned to node _CASLON::
```

In this example, the name of the local node is CASLON. This SET HOST command connects the user terminal to the processor at the network node named ITALIC. The remote processor then prompts for user name and password. Use the normal login procedure to log in to the remote processor.

Once you are logged in at a remote node, you can use the SET HOST command to establish communication with another node. After logging into node ITALIC, you could type SET HOST BODONI. You would again be prompted for a user name and password. If you then supply a valid user name and password, you will be logged in at node BODONI. Note that when you log out at node BODONI, control is returned to node ITALIC. You must log out from node ITALIC to return to your local node, CASLON.

SET HOST/DTE

Connects your system to a remote system through an out-going terminal line. Exit from the remote system by typing CTRL/\; that is, type a backslash (\) while holding down the CTRL key.

You must have an account on the remote system in order to log in to that system after the connection is made. Also requires the ability to assign a channel to the terminal port specified. By default, SYSPRV privilege is required but can be changed by setting the device protection for the terminal port.

DCL-310 DCL Commands

SET HOST/DTE

format

SET HOST/DTE *terminal-name*

parameter

terminal-name

Specifies the name of an out-going terminal line, which connects your system directly to another system or to a modem.

qualifiers

/DIAL=(NUMBER:number[,MODEM_TYPE:modem-type])

Allows a modem attached to the out-going terminal line to be autodialed using the autodial protocol of that modem. The NUMBER keyword is the telephone number to be autodialed and is a required parameter. The MODEM_TYPE: keyword is optional and can be used to specify a modem-type of DF03, DF112 or DMCL. By default, a modem-type of DF03 is assumed. DMCL is any modem that uses the DEC Modem Command Language.

/LOG[=file-spec]

/NOLOG

Controls whether a log file of the entire session is kept. If you do not specify a file, the log information is stored in the file `SETHOST.LOG`.

example

```
$ SET HOST/DTE/DIAL=(NUMBER:5551234#,MODEM_TYPE:DF112) TTA2:  
Username: SMITH  
Password:
```

The SET HOST/DTE command in this example accomplishes the same thing as in the first example, except that it uses the DF112 modem. Note that the number sign (#) is required to activate the autodialer in the DF112.

SET HOST/DUP

Connects your terminal to a storage controller through the appropriate bus for that controller.

For use only with storage controllers. Requires the DIAGNOSE privilege.

format

SET HOST/DUP/SERVER=server-name
/TASK=task-name node-name

parameter

node-name
Specifies the node name of the storage controller.

qualifiers

/LOG[=file-spec]
/NOLOG (default)
Controls whether a log file of the entire session is kept. If you use **/LOG** without the file specification, the log information is stored in the file **HSCPAD.LOG**.

/SERVER=server-name
Specifies the server name for the target storage controller.

This qualifier is required.

/TASK=task-name
Specifies the utility or diagnostic name to be executed on the target storage controller under direction of the server.

This qualifier is required.

example

```
$ SET HOST/DUP/SERVER=DUP$/TASK=DIRECT BLKHOL
%HSCPAD-I-LOCPROGEXE, Local program executing - type ^\ to exit utility
The SET HOST/DUP command in this example connects the user
terminal to the utility program called DIRECT executing on a storage
controller named BLKHOL under direction of the DUP$ server.
```

SET HOST/HSC

Connects your terminal to a remote HSC50 disk and tape controller through the Computer Interconnect bus.

Used only with remote HSC50s. Requires the DIAGNOSE privilege.

DCL-312 DCL Commands

SET HOST/HSC

format

SET HOST/HSC *node-name*

parameter

node-name

Specifies the node name of the remote HSC50.

qualifier

/LOG[=file-spec]

/NOLOG (default)

Controls whether a log file of the entire session is kept. If you use **/LOG** without the file specification, the log information is stored in the file HSCPAD.LOG.

example

```
$ SET HOST/HSC HSC001
%HSCPAD-I-LOCPROGEXE, Local program executing - type ^\ to exit, ^Y for prompt
HSC50>
```

This SET HOST/HSC command connects the user terminal to the HSC named HSC001.

SET KEY

Sets and locks the key definition state for keys defined with the DEFINE/KEY command.

format

SET KEY

qualifiers

/LOG (default)

/NOLOG

Controls whether the system displays a message indicating that the key state has been set.

/STATE=state-name

/NOSTATE

Specifies the name of the state. The state name can be any alphanumeric string. If you omit the **/STATE** qualifier or use **/NOSTATE**, the current state is left unchanged. The default state is **DEFAULT**.

example

```
$ SET KEY /STATE=EDITING
```

The **SET KEY** command in this example sets the key state to the **EDITING** state. You can now use the key definitions that were defined for the **EDITING** state.

SET LOGINS

Sets the interactive limit (number of interactive users allowed on the system), or displays the interactive limit and the current number of interactive users.

Requires OPER privilege.

format

SET LOGINS

parameters

None.

qualifier

/INTERACTIVE[=*n*]

Establishes the number of interactive users allowed to gain access to the system. If *n* is specified, the interactive limit is set to *n*. If *n* is not specified, the **SET LOGINS** command displays the current interactive limit and the number of interactive users.

example

```
$ SET LOGINS/INTERACTIVE=5
```

```
%SET-T-INTSET, login interactive limit=5, current interactive value=3
```

In this example, the **SET LOGINS** command specifies that only five interactive users can be logged in to the system.

SET MAGTAPE

Defines the default characteristics associated with a specific magnetic tape device for subsequent file operations.

The SET MAGTAPE command is valid for magnetic tape devices mounted with foreign volumes.

format

SET MAGTAPE *device-name[:]*

parameter

device-name[:]

Specifies the name of the magnetic tape device for which the characteristics are to be set. The device must not be currently allocated to any other user.

qualifiers

/DENSITY=density

Specifies the default density, in bits per inch (bpi), for all write operations on the magnetic tape device when the volume is mounted as a foreign tape or as an unlabeled tape. The density can be specified as 800, 1600, or 6250, if supported by the magnetic tape drive.

/END_OF_FILE

Writes a tape mark at the current position on the magnetic tape volume.

/LOG

/NOLOG

Displays information about the operations performed on the magnetic tape volume.

/LOGSOFT (default)

/NOLOGSOFT

Controls whether soft errors on the specified device are to be logged in the error log file. Soft errors are errors corrected by the hardware without software intervention. This qualifier only affects devices that support hardware error correction, such as the TU78 magnetic tape drive. When used with other devices, this qualifier has no effect.

/REWIND

Requests that the volume on the specified device be rewound to the beginning of the magnetic tape.

/SKIP=option

Requests that the magnetic tape volume be positioned according to any of the following options:

BLOCK:n	Directs the SET MAGTAPE command to skip the specified number of blocks
END_OF_TAPE	Directs the SET MAGTAPE command to position the volume at the end-of-tape mark

FILES:n Directs the SET MAGTAPE command to skip the specified number of files

RECORD:n Directs the SET MAGTAPE command to skip the specified number of records

/UNLOAD

Requests that the volume on the specified device be rewound and unloaded.

example

```
$ MOUNT MTB1:/FOREIGN  
$ SET MAGTAPE MTB1: /DENSITY=800
```

The MOUNT command in this example mounts a foreign tape on the device MTB1. The SET MAGTAPE command defines the density for writing the magnetic tape at 800 bpi.

SET MESSAGE

Sets the format for system messages or specifies a process level message file. Lets you override or supplement the system messages.

format

SET MESSAGE [*file-spec*]

parameter

file-spec

Specifies the name of the process level message file. Messages in this file supersede messages for the same conditions in the system message file or in an existing process message file. The file type defaults to EXE. No wildcard characters are allowed. If you do not specify this parameter, the qualifiers apply to the system message file.

qualifiers

/DELETE

Removes any process permanent message files currently in effect. Do not specify the file-spec parameter with the /DELETE qualifier.

/FACILITY (default)

/NOFACILITY

Formats messages so that the facility name prefix appears.

/IDENTIFICATION (default)

/NOIDENTIFICATION

Formats messages so that the message identification prefix appears.

DCL-316 DCL Commands

SET MESSAGE

/SEVERITY (default)

/NOSEVERITY

Formats messages so that the severity level appears.

/TEXT (default)

/NOTEXT

Formats messages so that the message text appears.

example

```
§ SET MESSAGE/TEXT/NOFACILITY/NOIDENTIFICATION/NOSEVERITY
§ SHOW DEVICES/MUONTED
unrecognized qualifier - check validity, spelling, and placement
\MUONTED\
```

The SET MESSAGE command in this example formats the error message so that only the text appears.

SET ON

Enables error checking by the command interpreter after the execution of each command in a command procedure. Specify SET NOON to disable error checking.

format

SET [NO]ON

parameters

None.

description

Use the SET NOON command to override default error checking. When SET NOON is in effect, the command interpreter continues to place the status code value in \$STATUS and the severity level in \$SEVERITY, but does not perform any action based on the values. As a result, the command procedure continues to execute no matter how many errors are returned. The SET ON or SET NOON command applies only at the current command level.

example

```
§ SET NOON
§ DELETE *.SAV;*
§ SET ON
§ COPY *.OBJ *.SAV
```

This command procedure routinely copies all object modules into new files with the file type SAV. The DELETE command first deletes all existing files with the SAV file type, if any. The SET NOON command ensures that the procedure continues executing even if there are no files with the SAV file type in the current directory. Following the DELETE command,

the SET ON command restores error checking. Then the COPY command makes copies of all existing files with OBJ file type.

SET OUTPUT_RATE

Sets the rate at which output is written to a batch job log file.

For use only within command procedures that are submitted as batch jobs.

format

SET OUTPUT_RATE[=*delta-time*]

parameter

delta-time

The time interval at which output is written from the output buffer to the batch job log file. If no delta time is specified, the information is written in the output buffer to the log file, but the output rate is not changed from the default of once per minute. Specify *delta-time* as [ddd-][hh:mm:ss.cc].

example

```
$ SET OUTPUT_RATE=:0:30
```

```
.  
. .  
. . .
```

This command, when executed within a batch job, changes the default output rate from once a minute to once every 30 seconds.

SET PASSWORD

Establishes, changes, or removes a password. SET PASSWORD can be used by users to change their own passwords and by system managers to change the system password.

See the qualifier descriptions for restrictions.

DCL-318 DCL Commands
SET PASSWORD

format

SET PASSWORD

parameters

None.

description

All user accounts on a system have passwords. A password is required for logging in to the system. A password contains up to 31 alphanumeric characters. The dollar sign (\$) and underscore (_) are also permitted. Uppercase and lowercase characters are equivalent. All lowercase characters are converted to uppercase before the password is encrypted. (For example, *EAGLE* is the same as *eagle*.)

Use the following procedure to change your password:

1. Enter the SET PASSWORD command.
2. The system prompts you for your current password. Enter your current password.
3. The system prompts you for a new password. Enter a new password, or press the RETURN key to disable your current password.
4. The system prompts you to verify the password. Enter the new password to verify. (If the two entries of the new password do not match, the password does not change.)

The following guidelines are recommended to minimize the chances of passwords being discovered by trial-and-error or by exhaustive search:

- Make passwords at least six characters long.
- Avoid names or words that are readily associated with you.
- Change your passwords at least once every month.

To ensure that the above guidelines are met, use the /GENERATE[=*value*] qualifier. This qualifier generates random passwords of up to 12 characters in length. The system manager can require individual users to use the /GENERATE qualifier.

qualifiers

/GENERATE[=*value*]

Generates a list of 5 random passwords. Press RETURN to repeat the procedure until a suitable password appears. If no value is specified, SET PASSWORD uses a default value of 6, and generates passwords from 6 to 8 characters long. Values greater than 10 are not accepted and produce errors.

/SECONDARY

Creates or allows you to replace a secondary password. The procedure is the same as setting your primary password. To remove your secondary password, press the RETURN key when SET PASSWORD/SECONDARY prompts you for a new password and verification. If you remove the secondary password, your system manager must restore it. The /SECONDARY and /SYSTEM qualifiers are incompatible.

/SYSTEM

Requires both SECURITY and CMKRNL privileges. Changes the system password, rather than a user password. The /SYSTEM and /SECONDARY qualifiers are incompatible. Refer to the *Guide to VMS System Security* for more information about the use of system passwords.

example

```
$ SET PASSWORD
Old password: HONCHO
New password: BIG_ENCHILADA
Verification: BIG_ENCHILADA
```

In response to the SET PASSWORD command, the system first prompts for the old password and then for the new password. The system then prompts again for the new password to verify it. The password changes if the user is authorized to change this account's password, if the old password is given correctly, and if the new password is given identically twice. Otherwise, an error message appears and the password remains unchanged.

In a real session, neither the old password nor the new password and its verification appear on the screen or paper.

SET PRINTER

Establishes the characteristics of a specific line printer. The default values listed for qualifiers to the SET PRINTER command are the defaults for an initially bootstrapped system.

Requires OPER privilege. If the printer is a spooled device, the logical I/O privilege (LOG_IO) is required to modify its characteristics.

format

```
SET PRINTER printer-name[:]
```

DCL-320 DCL Commands
SET PRINTER

parameter

printer-name[:]

Specifies the name of a line printer to set or modify its characteristics. If the printer has been set to /SPOOLED, the logical I/O privilege (LOG_IO) is required to modify its characteristics.

qualifiers

/CR

/NOCR (default)

Controls whether the printer driver outputs a carriage return character. Use this qualifier for printers on which line feeds do not imply carriage returns.

/FALLBACK

/NOFALLBACK (default)

Determines whether or not the printer attempts to translate characters belonging to the DEC Multinational Character Set into 7-bit equivalent representations. If a character cannot be translated, an underscore character is substituted.

/FF (default)

/NOFF

Indicates whether the printer performs a mechanical form feed.

/LA11

Specifies the printer as an LA11.

/LA180

Specifies the printer as an LA180.

/LOG

/NOLOG (default)

Determines whether information confirming the printer setting is displayed at the terminal from which the SET PRINTER command was entered.

/LOWERCASE

/NOLOWERCASE

Indicates whether the printer prints both uppercase and lowercase letters or only uppercase. When the operator specifies the /NOLOWERCASE qualifier, all letters are translated to uppercase.

/LP11 (default)

Specifies the printer as an LP11.

/PAGE=lines-per-page

Establishes the number of lines per page on the currently installed form; the number of lines can range from 1 to 255 and defaults to 64. The printer driver uses this value to determine the number of line feeds that must be entered to simulate a form feed.

/PASSALL

/NOPASSALL (default)

Controls whether the system interprets special characters or passes them as 8-bit binary data.

/PRINTALL

/NOPRINTALL (default)

Controls whether the line printer driver outputs printable 8-bit multinational characters.

/TAB

/NOTAB (default)

Controls how the printer handles TAB characters. The */NOTAB* qualifier expands all tab characters to spaces and assumes tab stops at eight character intervals.

/TRUNCATE (default)

/NOTRUNCATE

Controls whether the printer truncates data exceeding the value specified by the */WIDTH* qualifier. Note that the */TRUNCATE* and */WRAP* qualifiers are incompatible.

/UNKNOWN

Specifies the printer as nonstandard.

/UPPERCASE

/NOUPPERCASE

Indicates whether the printer prints both uppercase and lowercase letters or only uppercase ones. When you specify */UPPERCASE*, all letters are translated to uppercase.

/WIDTH=n

Establishes the number of characters per output line on currently installed forms. The width, *n*, can range from 0 through 65535 for LP11 controllers, and from 0 through 255 for DMF32 controllers. The default value is 132 characters per line.

/WRAP

/NOWRAP (default)

Controls whether the printer generates a carriage return/line feed when it reaches the end of a line.

example

```
$ SET PRINTER/PAGE=60/WIDTH=80 LPA0:
```

The SET PRINTER command in this example establishes the size of an output page as 60 lines and the width of a line as 80 characters for printer LPA0.

SET PROCESS

Changes the execution characteristics associated with the specified process for the current terminal session or job. If no process is specified, changes are made to the current process.

Requires GROUP privilege to change other processes in the same group. Requires WORLD privilege to change processes outside your group.

format

```
SET PROCESS [process-name]
```

parameter

process-name

Requires that you own the process or that you have GROUP privilege and that the process is in your group. Specifies the name of the process for which the characteristics are to be changed. Process names can be up to 23 alphanumeric characters long in the following format:

```
[node-name::]process-name
```

- The node name can have as many as 6 alphanumeric characters.
- The colons count for 2 characters.
- The process name can have as many as 15 characters.

A local process name can look like a remote process name. Therefore, if you specify ATHENS::SMITH, the system checks for a process named ATHENS::SMITH on the local node before checking node ATHENS for a process named SMITH.

The default process is the current process. The process name is compatible only with the /PRIORITY, /RESUME, and /SUSPEND qualifiers.

qualifiers

/DUMP

/NODUMP (default)

Causes the **contents** of the address space to be written to the file named SYS\$LOGIN:IMAGEDUMP.DMP when an image terminates due to an unhandled error.

/IDENTIFICATION=pid

Requires GROUP or WORLD privilege for processes other than your own. Specifies the process identification value (PID) of the process for which characteristics are to be changed. Overrides the process-name parameter. Compatible only with the **/PRIORITY**, **/RESUME**, and **/SUSPEND** qualifiers.

/NAME=string

Changes the name of the current process to a string of 1 through 15 characters.

/PRIORITY=n

Requires ALTPRI privilege to set the priority higher than the base priority of the specified process. Changes the priority for the specified process. If you do not have the ALTPRI privilege, the value you specify is compared to your current base priority, and the lower value is always used.

/PRIVILEGES=(privilege[,...])

Requires SETPRV privilege to enable a privilege you do not have. Enables privileges for the process. Use the **SHOW PROCESS/PRIVILEGES** command to determine what privileges are currently enabled.

/RESOURCE_WAIT

/NORESOURCE_WAIT

Enables resource wait mode so that the process waits for resources to become available. If you specify the **/NORESOURCE_WAIT** qualifier, the process receives an error status code when system dynamic memory is not available or when the process exceeds one of the following resource quotas: direct I/O limit, buffered I/O limit, or buffered I/O byte count (buffer space) quota.

/RESUME

Allows a process suspended by a previous **SET PROCESS/SUSPEND** command to resume operation. The **/RESUME** qualifier is equivalent to the **/NOSUSPEND** qualifier.

DCL-324 DCL Commands

SET PROCESS

/SUSPEND[=SUPERVISOR]
/SUSPEND=KERNEL
/NOSUSPEND

Requires privileges as described in text. Temporarily stops the process's activities. The process remains suspended until another process resumes or deletes it. Use the qualifiers ***/NOSUSPEND*** and ***/RESUME*** to resume a suspended process.

Specify either of the following keywords with ***/SUSPEND*** to produce different results:

Keyword	Result
SUPERVISOR (default)	Specifies that the named process is to be suspended to allow the delivery of Asynchronous System Traps (ASTs) at EXEC or KERNEL mode. Specifying this keyword is optional.
KERNEL	Specifies that the named process is to be suspended such that no asynchronous system traps (ASTs) can be delivered. To specify the KERNEL keyword, you must be in either kernel mode or exec mode, or have either CMKRNL or CMEEXEC privilege enabled. Note that this was the default behavior of SET PROCESS/SUSPEND for versions of VMS prior to Version 5.0.

Depending on the operation, the process from which you specify ***/SUSPEND*** requires privileges. You must have GROUP privilege to suspend another process in the same group, unless that process has the same UIC. You must have WORLD privilege to suspend any other process in the system.

Note that you can specify SET PROCESS/SUSPEND=KERNEL to override a previous SET PROCESS/SUSPEND=SUPERVISOR. SET PROCESS/SUSPEND=SUPERVISOR does not, however, override SET PROCESS/SUSPEND=KERNEL.

/SWAPPING (default)
/NOSWAPPING

Requires the user privilege process swap privilege (PSWAPM) to disable swapping for your process. Permits the process to be swapped.

example

```
$ RUN/PROCESS NAME=TESTER CALC
%RUN-S-PROC_ID, identification of created process is 0005002F
$ SET PROCESS/PRIORITY=10 TESTER
```

The RUN command in this example creates a subprocess and gives it the name TESTER. Subsequently, the SET PROCESS/PRIORITY command assigns the subprocess a priority of 10.

SET PROMPT

Replaces the default DCL prompt (\$) with the specified string.

format

SET PROMPT[=*string*]

parameter

string

Specifies the new prompt string. The following rules apply:

- All valid ASCII characters can be used.
- No more than 32 characters are allowed.
- To include spaces or lowercase letters, enclose the string in quotation marks. Otherwise, letters are automatically converted to uppercase; leading and trailing spaces are removed.

If you do not specify the string parameter with the SET PROMPT command, the default DCL prompt (\$) is restored.

qualifier

/CARRIAGE_CONTROL (*default*)

/NOCARRIAGE_CONTROL

Inserts carriage return and line feed characters before the prompt string. Type the qualifier after the string parameter.

example

```
$ SET PROMPT ="What's next?"  
What's next? SHOW TIME  
19-APR-1990 14:08:58
```

The SET PROMPT command in this example replaces the DCL prompt (\$) with the phrase "What's next?". When you see the prompt on your screen, you can enter any DCL command. This example uses the SHOW TIME command.

SET PROTECTION

Establishes the protection that limits other users' access to a file or a group of files.

You cannot change the protection on a file on a network node other than the one you are currently logged in to.

format

SET PROTECTION[=(*code*)] *file-spec*[,...]

parameters

code

Defines the protection to be applied to the specified files. If you omit the code, the access is set to the current default protection.

file-spec[,...]

Specifies one or more files for which the protection is to be changed. A file name and file type are required. If you omit a version number, the protection is changed only for the highest existing version of the file. Wildcard characters are allowed.

qualifiers

/CONFIRM

/NOCONFIRM (default)

Controls whether the SET PROTECTION command displays the file specification of each file before applying the new protection, and requests you to confirm that the file's protection should be changed. To change the protection, type Y (YES) or T (TRUE) at the system prompt and press RETURN. If you enter anything else, such as N or NO, the file protection is not changed.

/LOG

/NOLOG (default)

Controls whether the system displays the file specification of each file for which the protection is changed as the command executes.

/PROTECTION=(*code*)

File-spec qualifier. If you follow a file specification with the /PROTECTION qualifier, the code specified with /PROTECTION overrides the command's code parameter. The /PROTECTION qualifier lets you assign different protection codes to several files with a single SET PROTECTION command.

example

```
$ SET PROTECTION A.DAT, B.DAT/PROTECTION=OWNER:RWED, C.DAT
```

The SET PROTECTION command in this example specifies that the file A.DAT receive the default protection established for your files. The existing protection for the file B.DAT is overridden, only for the owner category, to provide read, write, execute, and delete access. Note that no protection is specified for the file C.DAT at either the command or file level. Like A.DAT, C.DAT receives the default protection.

Since no version numbers are specified, the protection settings affect only the highest versions of the three files.

```
$ DIR/PROTECTION INCOME.DAT
Directory DBAO:[SMITH]
INCOME.DAT;2          (RWED, RWED, RWED, RWED)
INCOME.DAT;1          (RWED, RWED, RWED, RWED)
Total of 2 files.
$ SET PROTECTION=(OWNER:RWE) INCOME.DAT;1
$ PURGE
```

In this example, the file INCOME.DAT;1 has been protected against deletion by the owner. However, because the owner is also a member of the group and world categories, the file is still vulnerable to deletion. The subsequent PURGE command deletes INCOME.DAT;1.

To protect the file against deletion by you (the owner), you also need to protect the file against deletion by all outer access categories. The following command shows the proper way to do this.

```
$ SET PROTECTION=(OWNER:RWE, GROUP:RWE, WORLD:RWE) INCOME.DAT;1
```

SET PROTECTION/DEFAULT

Establishes the default protection to be applied to all files subsequently created.

format

```
SET PROTECTION[=(code)]/DEFAULT
```

parameter

code

Defines the default protection to be applied to all files. To override this default protection use either the SET PROTECTION or CREATE commands. If you do not specify a protection code, the current default protection remains unchanged.

example

```
$ SET PROTECTION=(GROUP:RWED,WORLD:R)/DEFAULT
```

The **SET PROTECTION/DEFAULT** command in this example sets the default protection to grant unlimited access to other users in the same group and read access to all users. The default protections for system and owner are not changed.

SET PROTECTION/DEVICE

Establishes the protection to be applied to a specific non-file-structured device. The protection for a device limits the type of access available to users. The **/DEVICE** qualifier is required.

Requires OPER privilege.

format

```
SET PROTECTION=(ownership[:access],...)/DEVICE  
device-name[:]
```

parameters

ownership

An ownership category—SYSTEM, OWNER, GROUP, or WORLD. Each category can be abbreviated to its first character. Any protection code category that the operator does not specify will remain unchanged.

access

An access category—R (READ), W (WRITE), L (LOGICAL I/O), and P (PHYSICAL I/O)—to be assigned to a specified type of owner. A null access specification means no access.

device-name[:]

Specifies the name of the non file-structured device whose protection is to be set or modified.

qualifier

/OWNER_UIC=*uic*

Requests that the specified user identification code (UIC) be assigned ownership of the device for the purpose of access checks. The default owner is the UIC of the process entering the **SET PROTECTION** command.

example

```
$ SET PROTECTION=(S:RWLP,O:RWLP,G,W)/DEVICE LAA0:
```

The command in this example requests that the protection for device LAA0 be set to allow all types of access to system processes and processes with the UIC of the current process. This command also denies access to anyone else.

SET QUEUE

Changes the attributes of the specified queue. The /QUEUE qualifier is required.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

```
SET QUEUE queue-name[:]
```

parameter

queue-name[:]

Specifies the name of an execution queue or a generic queue.

qualifiers

/BASE_PRIORITY=n

Specifies the base process priority at which jobs are initiated from a batch execution queue. The base priority specifier can be any decimal value from 0 through 15.

You also can specify this qualifier for an output execution queue. In this context the /BASE_PRIORITY qualifier establishes the base priority of the symbiont process when the symbiont process is created.

/BLOCK_LIMIT=(*[lowlim,]uplim*)

/NOBLOCK_LIMIT

Limits the size of print jobs that can be processed on an output execution queue. This qualifier allows you to reserve certain printers for certain size jobs. You must specify at least one of the parameters. The lowlim parameter is a decimal number referring to the minimum number of blocks that are accepted by the queue for a print job. If a print job is submitted that contains fewer blocks than the lowlim value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is decreased sufficiently, the job is processed.

DCL-330 DCL Commands

SET QUEUE

The `uplim` parameter is a decimal number referring to the maximum number of blocks that are accepted by the queue for a print job. If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is increased sufficiently, the job is processed.

/CHARACTERISTICS=(characteristic[,...])
/NOCHARACTERISTICS

Specifies one or more characteristics for processing jobs on an execution queue. If a queue does not have all the characteristics that have been specified for a job, the job remains pending. If you specify only one characteristic, you can omit the parentheses. Each time you specify `/CHARACTERISTICS`, all previously set characteristics are cancelled. Only the characteristics specified with the qualifier are established for the queue. Queue characteristics are installation-specific. The characteristic parameter can be either a value from 0 through 127 or a characteristic name that has been defined by the `DEFINE/CHARACTERISTIC` command.

/CLOSE

Prevents jobs from being entered in the queue through `PRINT` or `SUBMIT` commands or as a result of requeue operations. To allow jobs to be entered, use the `/OPEN` qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled). When a queue is marked closed, jobs executing continue to execute and jobs pending in the queue continue to be candidates for execution.

/CPUDEFAULT=time

Defines the default CPU time limit for jobs in a batch execution queue. You can specify time as delta time, 0, `INFINITE`, or `NONE`. You can specify up to 497 days of delta time.

If the queue does not have a defined `CPUMAXIMUM` time limit and the value established in the user authorization file (UAF) has a specified CPU time limit of `NONE`, either the value 0 or the keyword `INFINITE` allows unlimited CPU time. If you specify `NONE`, the CPU time value defaults to the value specified either in the UAF or by the `SUBMIT` command (if included). CPU time values must be greater than or equal to the number specified by the `SYSGEN` parameter `PQL_MCPULM`. The time cannot exceed the CPU time limit set by the `/CPUMAXIMUM` qualifier.

/CPUMAXIMUM=time

Defines the maximum CPU time limit for all jobs in a batch execution queue. You can specify time as delta time, 0, `INFINITE`, or `NONE`. You can specify up to 497 days of delta time.

The **/CPUMAXIMUM** qualifier overrides the time limit specified in the user authorization file (UAF) for any user submitting a job to the queue. Either the value 0 or the keyword **INFINITE** allows unlimited CPU time. If you specify **NONE**, the CPU time value defaults to the value specified either in the UAF or by the **SUBMIT** command (if included). CPU time values must be greater than or equal to the number specified by the **SYSGEN** parameter **PQL_MCPULM**.

/DEFAULT=(option[,...])
/NODEFAULT

Establishes defaults for certain options of the **PRINT** command. Defaults are specified by the list of options. If you specify only one option, you can omit the parentheses. After you set an option for the queue with the **/DEFAULT** qualifier, you do not have to specify that option in your **PRINT** commands. If you do specify these options in your **PRINT** command, the values specified with the **PRINT** command override the values established for the queue with the **/DEFAULT** qualifier. Possible options are as follows:

- | | |
|------------------------------|---|
| [NO]BURST[=keyword] | Controls whether two file flag pages with a burst bar between them are printed preceding output. If you specify the value ALL (default), these flag pages are printed before each file in the job. If you specify the value ONE , these flag pages are printed once before the first file in the job. |
| [NO]FEED | Specifies whether a form feed is inserted automatically at the end of a page. |
| [NO]FLAG[=keyword] | Controls whether a file flag page is printed preceding output. If you specify the value ALL (default), a file flag page is printed before each file in the job. If you specify the value ONE , a file flag page is printed once before the first file in the job. |
| FORM=type | Specifies the default form for an output execution queue. If a job is submitted without an explicit form definition, this form is used to process the job. See also /FORM_MOUNTED . |
| [NO]TRAILER[=keyword] | Controls whether a file trailer page is printed following output. If you specify the value ALL (default), a trailer page is printed with each file in the job. If you specify the value ONE , a trailer page is printed once with the last file in the job. |

When you specify the **BURST** option for a file, the **[NO]FLAG** option does not add or subtract a flag page from the two flag pages that are printed preceding the file.

/DESCRIPTION=string
/NODESCRIPTION

A string of up to 255 characters used to provide operator-supplied information about the queue.

DCL-332 DCL Commands
SET QUEUE

Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (").

The /NODESCRIPTION qualifier removes any descriptive text that may have been associated with the queue.

/DISABLE_SWAPPING
/NODISABLE_SWAPPING

Controls whether batch jobs executed from a queue can be swapped in and out of memory.

/ENABLE_GENERIC
/NOENABLE_GENERIC

Specifies whether files queued to a generic queue that does not specify explicit queue names can be placed in this execution queue for processing.

/FORM_MOUNTED=type

Specifies the mounted form for an output execution queue. If the stock of the mounted form is not identical to the stock of the default form, as indicated by the /DEFAULT=FORM qualifier, all jobs submitted to this queue without an explicit form definition enter a pending state. If a job is submitted with an explicit form and the stock of the explicit form is not identical to the stock of the mounted form, the job enters a pending state. In both cases, jobs remain pending until the stock of the mounted form of the queue is identical to the stock of the form associated with the job. To specify the form type, use either a numeric value or a form name that has been defined by the DEFINE/FORM command. Form types are installation-specific.

/JOB_LIMIT=n

Indicates the number of batch jobs that can be executed concurrently from the queue. Specify a number in the range 0 through 255.

/OPEN

Allows jobs to be entered in the queue through PRINT or SUBMIT commands or as the result of requeue operations. To prevent jobs from being entered in the queue, use the /CLOSE qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled).

/OWNER_UIC=uic

Requires OPER privilege or CONTROL and EXECUTE access to the queue. Enables you to change the user identification code (UIC) of the queue.

/PROTECTION=(ownership[:access],...)

Requires OPER privilege or CONTROL and EXECUTE access to the queue. Specifies the protection of the queue. Ownership categories are SYSTEM, OWNER, GROUP, WORLD; each category can be abbreviated to its first character. Access categories are R (READ),

W (WRITE), E (EXECUTE), or D (DELETE); a null access specification means no access. If you include only one protection code, you can omit the parentheses.

/RECORD_BLOCKING
/NORECORD_BLOCKING

Determines whether the symbiont can concatenate (or block together) output records for transmission to the output device. If you specify ***/NORECORD_BLOCKING***, the symbiont sends each formatted record in a separate I/O request to the output device. For the standard VMS print symbiont, record blocking can have a significant performance advantage over single-record mode.

/RETAIN[=option]
/NORETAIN

Holds jobs in the queue in a retained status after they have executed. The ***/NORETAIN*** qualifier enables you to reset the queue to the default. Possible options are as follows:

ALL Holds all jobs in the queue after execution (default)
ERROR Holds in the queue only jobs that complete unsuccessfully

/SCHEDULE=[NO]SIZE

Specifies whether pending jobs in an output queue are scheduled for printing based on the size of the job. When the ***/SCHEDULE=SIZE*** qualifier is in effect, shorter jobs print before longer ones. When ***/SCHEDULE=NOSIZE*** is in effect, jobs are printed in the order they were submitted, regardless of size.

If you enter this command while there are pending jobs in any queue, its effect on future jobs is unpredictable.

/SEPARATE=(option[,...])
/NOSEPARATE

Specifies the mandatory queue attributes or job separation options for an output execution queue. Job separation options cannot be overridden by the PRINT command.

The job separation options are as follows:

[NO]BURST Specifies whether two job flag pages with a burst bar between them are printed at the beginning of each job.
[NO]FLAG Specifies whether a job flag page is printed at the beginning of each job.
[NO]TRAILER Specifies whether a job trailer page is printed at the end of each job.

DCL-334 DCL Commands

SET QUEUE

[NO]RESET=(module[,...])

Specifies one or more device control library modules that contain the job reset sequence for the queue. The specified modules from the queue's device control library (by default SYS\$LIBRARY:SYSDEVCTL) are used to reset the device each time a job reset occurs. The RESET sequence occurs after any file trailer and before any job trailer. Thus, all job separation pages are printed when the device is in its RESET state.

When you specify /SEPARATE=BURST, the [NO]FLAG separation option does not add or subtract a flag page from the two flag pages that are printed preceding the job.

For information on establishing queue attributes that can be overridden, see the description of the /DEFAULT qualifier.

/WSDEFAULT=n

Defines for a batch job a working set default, the default number of physical pages that the job can use. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

If you specify 0 or NONE, the working set default value defaults to the value specified in the UAF or by the SUBMIT command (if included).

/WSEXTENT=n

Defines for the batch job a working set extent, the maximum amount of physical memory that the job can use. The job uses the maximum amount of physical memory only when the system has excess free pages. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

If you specify 0 or NONE, the working set extent value defaults to the value specified in the UAF or by the SUBMIT command (if included).

/WSQUOTA=n

Defines for a batch job the working set quota, the amount of physical memory that is available to the job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue. If you specify 0 or NONE, the working set quota value defaults to the value specified in the UAF or by the SUBMIT command (if included).

example

```
$ INITIALIZE/QUEUE/DEFAULT=BURST/FORM_MOUNTED=LETTER/START SYSSPRINT
```

```
.  
. .  
. . .
```

```
$ STOP/QUEUE/NEXT SYSSPRINT
```

```
$ SET QUEUE /DEFAULT=BURST/FORM_MOUNTED=MEMO SYSSPRINT
```

In this example, the queue is initialized with the INITIALIZE/QUEUE command. The queue has the following attributes: two file flag pages preceding each file in the job and the mounted form LETTER. Later the queue is stopped with the STOP/QUEUE/NEXT command so that the current job finishes processing before the queue stops. The SET QUEUE command changes the mounted form to MEMO.

SET QUEUE/ENTRY

Changes the current status or attributes of a job that is not currently executing in a queue. The /ENTRY qualifier is required.

As of VMS Version 5.0, the SET QUEUE/ENTRY command is superseded by the SET ENTRY command. Note that the SET ENTRY command has the same qualifiers as the SET QUEUE/ENTRY command; only the command parameters are different. Digital recommends usage of the SET ENTRY command.

Requires OPER privilege or EXECUTE (E) access to the specified queue. If you have DELETE (D) access to the specified job, you can alter the attributes for that job. In addition, the queue name parameter is optional.

format

```
SET QUEUE/ENTRY=entry-number queue-name[:]
```

SET RESTART_VALUE

Assigns a value to the global symbol BATCH\$RESTART. This global symbol defines the location at which a batch job is restarted after its execution has been interrupted. Use the SET RESTART_VALUE command in command procedures. This command has no meaning if you enter it interactively.

format

SET RESTART_VALUE=*string*

parameter

string

A string of up to 255 characters specifying the label at which the batch job should begin executing when the batch job is restarted.

example

```
$ IF $RESTART THEN GOTO 'BATCH$RESTART'
```

```
.  
. .  
. .
```

```
$ FIRSTPART:  
$ SET RESTART_VALUE = FIRSTPART  
$ RUN PART1
```

```
.  
. .  
. .
```

```
$ SECONDPART:  
$ SET RESTART_VALUE = SECONDPART  
$ RUN PART2
```

```
.  
. .  
. .
```

In this example, the first command states that, if \$RESTART is true, proceed to the value contained in BATCH\$RESTART. (\$RESTART is true only if the job has been executed before, that is, the job is being rerun after a crash or after having been queued.)

The first SET RESTART_VALUE command assigns the label FIRSTPART to be equal to the symbol BATCH\$RESTART. The next line contains the command to run PART1.EXE.

The second SET RESTART_VALUE command assigns the label SECONDPART to be equal to the symbol BATCH\$RESTART. The last line shown contains the command to run PART2.EXE.

When the job is first submitted using the SUBMIT/RESTART command, the value of \$RESTART is FALSE, so the IF expression is ignored. If the job is stopped during the run of PART1.EXE, the value of BATCH\$RESTART is FIRSTPART. When the job is restarted, the value of \$RESTART is TRUE. Thus, the IF expression is processed and transfers control to the FIRSTPART label in the procedure. PART1.EXE is rerun.

If the job is stopped during the run of PART2.EXE, the value of BATCH\$RESTART is SECONDPART. When the job is restarted, the value of \$RESTART is TRUE. In this instance, the IF—GOTO command transfers control to the SECONDPART label in the procedure so that PART2.EXE can be run. PART1.EXE is not rerun.

SET RIGHTS_LIST

Allows users to modify the process or system rights list. You must specify either /DISABLE or /ENABLE with the SET RIGHTS_LIST command.

format

SET RIGHTS_LIST *id-name[,...]*

parameter

id-name[,...]

Specifies identifiers to be added to or removed from the process or system rights list. The id-name parameter is a string of 1 to 31 alphanumeric characters, underscores, and dollar signs; each name must contain at least one nonnumeric character.

qualifiers

/ATTRIBUTES=(keyword[,...])

Specifies attributes to be associated with the identifiers. Attributes may be added to new or existing identifiers. The following are valid keywords:

[NO]DYNAMIC Indicates whether or not unprivileged holders of the identifiers may add or remove them from the process rights list. The default is NODYNAMIC.

[NO]RESOURCE Indicates whether or not holders of the identifiers may charge resources to them. The default is NORESOURCE.

/DISABLE

Removes the identifiers from the process or system rights list. You cannot use **/DISABLE** with the **/ENABLE** qualifier.

/ENABLE

Adds the identifiers to the process or system rights list. You cannot use **/ENABLE** with the **/DISABLE** qualifier.

/IDENTIFICATION=pid

Specifies the process identification value (PID) of the process whose rights list is to be modified. The PID is assigned by the system when the process is created. When you specify a PID, you can omit the leading zeros.

If you specify the **/IDENTIFICATION** qualifier, you cannot use the **/PROCESS** qualifier. By default, if neither the **/IDENTIFICATION** nor the **/PROCESS** qualifier is specified, the current process is assumed. You cannot use **/IDENTIFICATION** with the **/SYSTEM** qualifier.

/PROCESS[=process-name]

Specifies the name of the process whose rights list is to be modified. The process name can contain from 1 to 15 alphanumeric characters.

If you specify the **/PROCESS** qualifier, you cannot use the **/IDENTIFICATION** qualifier. By default, if neither the **/PROCESS** nor the **/IDENTIFICATION** qualifier is specified, the current process is assumed.

/SYSTEM

Specifies that the desired operation (addition or removal of an identifier) be performed on the system rights list. You cannot use **/SYSTEM** with **/PROCESS** or **/IDENTIFICATION**.

example

```
$ SET RIGHTS_LIST/ENABLE/ATTRIBUTES=RESOURCE MARKETING
```

The **SET RIGHTS_LIST** command in this example adds the **MARKETING** identifier to the process rights list of the current process. Specifying the **RESOURCE** attribute allows holders of the **MARKETING** identifier to charge resources to it.

SET RMS_DEFAULT

Defines default values for the multiblock and multibuffer counts, network transfer sizes, prolog level, and extend quantity used by VMS RMS for file operations. If you set the default for either the multiblock count or the multibuffer count at 0, VMS RMS tries to use the process default value or the system default value, in that order. If these are set at 0, VMS RMS uses a default value of 1.

format

SET RMS_DEFAULT

parameters

None.

qualifiers

/BLOCK_COUNT=count

Specifies a default multiblock count (0 through 127) for record I/O operations *only*, where count is the number of blocks to be allocated for each I/O buffer.

/BUFFER_COUNT=count

Specifies a default multibuffer count (0 through 127) for file operations. If file type is not specified, the default is applied to sequential files.

/DISK

Applies the specified defaults to disk file operations.

/EXTEND_QUANTITY=n

Specifies the number of blocks (n) to extend a sequential file where n can range from 0 to 65535. If you do not specify */EXTEND_QUANTITY*, VMS RMS calculates its own extend value. The */EXTEND_QUANTITY* qualifier value is used when the program does not explicitly specify an extent quantity.

/INDEXED

Applies the multibuffer default to indexed file operations.

/MAGTAPE

Applies the multibuffer default to magnetic tape operations.

/NETWORK_BLOCK_COUNT=count

Specifies a default block count (0 through 127) for network access to remote files, where count represents the number of I/O buffers that VMS RMS allocates for transmitting and receiving data. If you omit the value or specify a value of 0, VMS RMS uses the systemwide block count value. If this value is also 0, VMS RMS uses a size of one block.

DCL-340 DCL Commands

SET RMS_DEFAULT

/PROLOG=n

Specifies a default prolog level for indexed sequential files where acceptable values for n are 0, 2 or 3. If 0 (default) is specified, VMS RMS sets an appropriate prolog level.

/RELATIVE

Applies the multibuffer default to relative file operations.

/SEQUENTIAL (default)

Applies the multibuffer default to sequential file operations.

/SYSTEM

Requires CMKRNL privilege. Applies specified defaults on a systemwide basis to all file operations.

/UNIT_RECORD

Applies the multibuffer default to file operations on unit record devices.

example

```
$ SET RMS_DEFAULT/BUFFER_COUNT=7/NETWORK_BLOCK_COUNT=16/SYSTEM
$ SHOW RMS_DEFAULT
```

	MULTI- BLOCK COUNT	MULTIBUFFER COUNTS					NETWORK BLOCK COUNT
		Indexed	Relative	Disk	Sequential Magtape	Unit Record	
Process	24	0	0	0	8	0	0
System	16	0	0	7	7	7	16

	Prolog	Extend	Quantity
Process	0		0
System	0		0

The SET RMS_DEFAULT command in this example defines the systemwide default multibuffer count at 7 for all sequential file operations on disk, magnetic tape, and unit record devices. The command also sets the network block count at 16.

SET SYMBOL

Controls access to local and global symbols in command procedures.

format

SET SYMBOL

qualifier

/SCOPE=(keyword,...)

Controls access to local and global symbols. Lets you treat symbols as being undefined. Possible keywords are as follows:

NOLOCAL	Causes all local symbols defined in outer procedure levels to be treated as being undefined by the current procedure and all inner procedure levels.
LOCAL	Removes any symbol translation limit set by the current procedure level.
NOGLOBAL	Causes all global symbols to be inaccessible to the current procedure level and all inner procedure levels unless otherwise changed.
GLOBAL	Restores access to all global symbols.

example

```
$ SET SYMBOL/SCOPE=NOLOCAL
```

In this example, all local symbols defined in outer procedure levels are now undefined by the current procedure and all inner procedure levels.

SET TERMINAL

Sets the characteristics of a terminal. Entering a qualifier changes a characteristic; omitting a qualifier leaves the characteristic unchanged.

format

```
SET TERMINAL [device-name[:]]
```

parameter

device-name[:]

Specifies the device name of the terminal. The default is SYS\$COMMAND if that device is a terminal. If the device is not a terminal, an error message is displayed.

qualifiers

/ADVANCED_VIDEO ***/NOADVANCED_VIDEO***

Specifies that the terminal has advanced video attributes and is capable of 132-column video. If the terminal width is set to 132 columns and */ADVANCED_VIDEO* is enabled, the terminal page limit is set to 24 lines. If */NOADVANCED_VIDEO* is enabled, the terminal page limit is set to 12 lines.

/ALTYPEAHD

Causes the terminal driver to create a permanent, alternate type-ahead buffer. The SYSGEN parameter TTY_ALTYPAHD determines the size of the type-ahead buffer. This specification is effective at your next login and stays in effect until you reboot your VAX computer.

You should specify SET TERMINAL/PERMANENT/ALTYPEAHD in SYS\$SYSTEM:STARTUP.COM for those communication lines that require this capability.

DCL-342 DCL Commands
SET TERMINAL

To use this feature interactively, specify **SET TERMINAL/PERMANENT-
/ALTYPEAHD**. This specification is effective at your next login.

/ANSI CRT (default)
/NOANSI CRT

Specifies whether the terminal conforms to ANSI CRT programming standards. Since ANSI standards are a proper subset of the DEC_CRT characteristics, the default for all VT100-family terminals is **/ANSI_CRT**.

/APPLICATION_KEYPAD

Specifies that the keypad is to be set to **APPLICATION_KEYPAD** mode, which allows you to enter DCL commands defined with the **DEFINE/KEY** command. By default, the terminal is set to **NUMERIC_KEYPAD** mode.

/AUTOBAUD
/NOAUTOBAUD

Specifies whether the terminal baud rate is set when you log in and sets the default terminal speed to 9600. You must press the **RETURN** key two or more times at intervals of at least one second for the baud rate to be correctly determined. If you press a key other than **RETURN**, **/AUTOBAUD** might detect the wrong baud rate. If this happens, wait for the login procedure to time out before continuing. The **/AUTOBAUD** qualifier must be used with the **/PERMANENT** qualifier.

The valid baud rates are as follows:

110	1200	4800
150	1800	9600
300	2400	19200
600	3600	

/BLOCK MODE
/NOBLOCK MODE

Performs block mode transmission, local editing, and field protection.

/BRDCSTMBX
/NOBRDCSTMBX

Sends broadcast messages to an associated mailbox if one exists.

/BROADCAST (default)
/NOBROADCAST

Enables reception of broadcast messages (such as those issued by **MAIL** and **REPLY**). Specify the **/NOBROADCAST** qualifier when you are using a terminal as a noninteractive device or when you do not want special output to be interrupted by messages. Use **SET BROADCAST** to exclude certain types of messages from being broadcast, rather than eliminating all messages.

/CRFILL[=*fill-count*]

Generates the specified number of null characters after each carriage return before transmitting the next meaningful character (to ensure that the terminal is ready for reception). The value must be an integer in the range 0 through 9. The default is `/CRFILL=0`.

/DEC_CRT[=(*value1,value2,value3*)]

/NODEC_CRT[=(*value1,value2,value3*)]

Specifies that the terminal conforms to Digital VT100-, VT200-, or VT300-family standards and supports the minimum standards, including the additional Digital escape sequences.

One of the following three optional values may be specified:

- | | |
|-------------|---|
| 1 (default) | Requests that the DEC_CRT terminal characteristic be set. |
| 2 | Requests that the DEC_CRT2 terminal characteristic be set. |
| 3 | Requests that the DEC_CRT3 terminal characteristic be set. A level 3 terminal is described as follows: <ul style="list-style-type: none">• Supports a status line (line 25, at the bottom of the screen)• Supports the IOS Latin-1 character set• Has terminal state interrogation (describes what state your terminal is in) |

/DEVICE_TYPE=*terminal-type*

Informs the system of the terminal type and sets characteristics according to the device type specified. You can specify any of the following terminal types:

UNKNOWN	LA34
FT1 - FT8	LA38
LA12	LA100
LA36	LQP02
LA120	VT125
LN03	LN01K
VT05	VT131
VT52	VT132
VT55	VT173
VT100	VT200
VT101	PRO_SERIES
VT102	LA210
VT105	VT300

DCL-344 DCL Commands

SET TERMINAL

The default characteristics for the VT100-, VT102-, and VT125-series terminals are as follows:

/ADVANCEDVIDEO	/CRFILL=0	/LFFILL=0	/SPEED=9600
/NOALTYPEAHD¹	/ECHO	/LOWERCASE	/TAB
/ANSI_CRT	/NOEIGHT_BIT	/NODMA	/TTSYNC
/NOAUTOBAUD	/NOESCAPE	/PAGE=24	/TYPE_AHEAD
/NOBLOCK_MODE	/NOFORM	/NOPARITY	/WIDTH=80
/NOBRDCSTMBX	/FULLDUP	/NOPASTHRU	/WRAP
/BROADCAST	/NOHOSTSYNC	/NOREADSYN	

¹This is the default characteristic set by the system and is not a valid qualifier for your use.

/DIALUP

/NODIALUP (default)

Specifies that the terminal is a dial-up terminal.

/DISCONNECT

/NODISCONNECT (default)

Specifies that the process connected to this terminal not be disconnected if the line detects a hangup. The **/DISCONNECT** qualifier is valid only when **/PERMANENT** is specified.

/DISMISS

/NODISMISS (default)

Causes the terminal driver to ignore data causing a parity error (instead of terminating the currently outstanding I/O with an error status).

/DMA

/NODMA

Controls the use of direct memory access (DMA) mode on a controller that supports this feature.

/ECHO (default)

/NOECHO

Causes the terminal to display the input it receives. With **/NOECHO**, the terminal displays only system or user application output, or both.

/EDIT_MODE

/NOEDIT_MODE

Specifies that the terminal can perform ANSI-defined advanced editing functions.

/EIGHT_BIT

/NOEIGHT_BIT

Uses 8-bit ASCII protocol rather than 7-bit ASCII protocol.

/ESCAPE
/NOESCAPE (default)
Validates escape sequences.

/FALLBACK
/NOFALLBACK
Displays the 8-bit DEC Multinational Character Set characters on the terminal in their 7-bit representation. The default depends on the **/EIGHTBIT** setting of the terminal.

/FORM
/NOFORM
Transmits a form feed rather than translating it into multiple line feeds.

/FRAME=n
Specifies the number of data bits that the terminal driver expects for every character that is input or output. The value of *n* can be from 5 through 8. The default value depends on the **/PARITY** and **/EIGHTBIT** settings of the terminal.

/FULLDUP (default)
/NOFULLDUP
Operates in full duplex mode. The **/FULLDUP** qualifier is equivalent to **/NOHALFDUP**.

/HALFDUP
/NOHALFDUP (default)
Operates in half duplex mode. The **/HALFDUP** qualifier is equivalent to **/NOFULLDUP**.

/HANGUP
/NOHANGUP (default)
May require **LOG_IO** or **PHY_IO** privilege depending on system generation parameter settings. Controls whether the terminal modem is hung up when you log out.

/HARDCOPY
/NOHARDCOPY
Establishes the device as a hardcopy terminal and outputs a backslash (\) when the DELETE key is pressed. The **/HARDCOPY** qualifier is equivalent to **/NOSCOPE**.

/HOSTSYNC
/NOHOSTSYNC (default)
When you specify the **/HOSTSYNC** qualifier, the system stops transmission to the terminal (by generating a CTRL/S) when the input buffer is full and resumes transmission (by generating a CTRL/Q) when the input buffer is empty.

DCL-346 DCL Commands
SET TERMINAL

/INQUIRE

Sets the device type according to a response elicited from the terminal; the default is UNKNOWN. Works only on Digital terminals, and not on the LA36 or VT05 terminals. Some VT100-family terminals, including the VT101 and VT105, return a VT100-type response. LA38 terminals respond as LA43 terminals.

You can include the SET TERMINAL/INQUIRE command in your LOGIN.COM file to automatically detect the terminal type.

CAUTION: This qualifier clears the type-ahead buffer. If the response sequence is unrecognized, no action message or error message is displayed. The /INQUIRE qualifier should be used only on Digital terminals. However, the LA36 and VT05 terminals do not support this feature.

/INSERT

Sets the terminal to /INSERT mode. This feature allows you to insert characters when editing command lines. The default mode is /OVERSTRIKE, which allows you to type over the current character when editing a command line. Use CTRL/A to switch from one mode to the other.

/LFFILL[=fill-count]

Transmits to the terminal the specified number of null characters after each line feed before transmitting the next meaningful character (to ensure that the terminal is ready for reception). The value must be an integer in the range 0 through 9.

/LINE_EDITING

/NOLINE_EDITING

Enables advanced line-editing features for editing command lines: both RETURN and CTRL/Z are recognized as line terminators, as are escape sequences.

/LOCAL_ECHO

/NOLOCAL_ECHO (default)

Echoes characters locally (rather than the host echoing them) for command level terminal functions. (Do not use /LOCAL_ECHO with utilities that require control over echoing, such as line editing or EDT's screen mode.)

CAUTION: When logging in to terminals with /LOCAL_ECHO set, the VMS operating system has no control over the echoing of passwords.

/LOWERCASE
/NOLOWERCASE

Passes lowercase characters to the terminal. The */NOLOWERCASE* qualifier translates all input to uppercase. */LOWERCASE* is equivalent to */NOUPPERCASE*.

/MANUAL

Indicates manual switching of terminal lines to dynamic asynchronous DDCMP lines when your local terminal emulator does not support automatic switching. The */MANUAL* qualifier should be specified with the */PROTOCOL=DDCMP* and */SWITCH=DECNET* qualifiers.

/MODEM
/NOMODEM

Indicates that the terminal is connected to a modem or a cable that supplies standard EIA modem control signals. If your terminal has the *MODEM* characteristic, typing *SET TERMINAL/NOMODEM* automatically logs you out.

/NUMERIC_KEYPAD (default)

Specifies that the keypad is to be set to */NUMERIC_KEYPAD* mode, which allows you to use the keys on the numeric keypad to type numbers and punctuation marks. In order to use the *DEFINE/KEY* facility, which allows you to enter DCL commands defined with the *DEFINE/KEY* command, set the terminal to */APPLICATION_KEYPAD*. Specifies whether the keys of the numeric keypad are used to type numbers and punctuation marks (*/NUMERIC_KEYPAD*) or to enter DCL commands defined with the *DEFINE/KEY* command (*/APPLICATION_KEYPAD*).

/OVERSTRIKE (default)

Sets the terminal to */OVERSTRIKE* mode. This feature allows you to type over the current character when you are editing a command line. Set your terminal to */INSERT* if you want to insert characters when editing command lines. Use *CTRL/A* to switch from one mode to the other.

/PAGE[=lines-per-page]

For hardcopy terminals, specifies the number of print lines between perforations. (When the terminal reads a form feed, it advances the paper to the next perforation.) The value of *n* can be from 0 through 255 and defaults to 0 (which treats a form feed as a line feed).

/PARITY[=option]
/NOPARITY (default)

Passes data with odd or even parity, where *option* equals *ODD* or *EVEN*. If you specify */PARITY* without an option, the value defaults to *EVEN*.

DCL-348 DCL Commands
SET TERMINAL

/PASTHRU

/NOPASTHRU (default)

Passes all data (including tabs, carriage returns, line feeds, and control characters) to an application program as binary data. The setting of /TTSYNC is allowed.

Make sure that you spell both these qualifiers exactly as they appear in the text.

/PERMANENT

Requires LOG_IO or PHY_IO privilege. Sets characteristics on a permanent basis, that is, over terminal sessions. However, the characteristics revert to their initial values if the system is halted and restarted. Use in a system startup file to establish characteristics for all terminals on the system.

/PRINTER_PORT

/NOPRINTER_PORT

Specifies that the terminal has a printer port (an attribute not set by the SET TERMINAL/INQUIRE command).

/PROTOCOL=DDCMP

/PROTOCOL=NONE (default)

Controls whether the terminal port specified is changed into an asynchronous DDCMP line. The /PROTOCOL=NONE qualifier changes an asynchronous DDCMP line back into a terminal line. Note that /PROTOCOL=DDCMP is a permanent characteristic; therefore, the /PERMANENT qualifier is not required.

/READSYNC

/NOREADSYNC (default)

Uses the CTRL/S and CTRL/Q functions to synchronize data transmitted from the terminal.

The default is /NOREADSYNC; the system does not use CTRL/S and CTRL/Q to control reads to the terminal. The /READSYNC qualifier is useful for certain classes of terminals that demand synchronization or for special-purpose terminal lines where data synchronization is appropriate.

/REGIS

/NOREGIS

Specifies that the terminal understands ReGIS graphic commands.

/SCOPE

/NOSCOPE

Establishes the device as a video terminal. /SCOPE is equivalent to /NOHARDCOPY.

/SECURE_SERVER
/NOSECURE_SERVER (default)

Causes the BREAK key on the terminal to log out the current process (except on a virtual terminal). With */SECURE_SERVER* in effect, pressing the BREAK key when there is no current process initiates the login sequence. With */NOSECURE_SERVER* in effect, the break is ignored.

On terminals set with */AUTOBAUD*, with the */SECURE_SERVER* qualifier in effect, pressing the BREAK key disconnects the current process but is not required to start a new login sequence. However, when */NOAUTOBAUD* is set, the */SECURE_SERVER* characteristic requires a break to initiate a new login sequence.

/SET_SPEED
/NOSET_SPEED

Requires either LOG_IO or PHY_IO privilege. Allows the */SPEED* qualifier to be used to change the terminal speed.

/SIXEL_GRAPHICS
/NOSIXEL_GRAPHICS

Specifies that the terminal is capable of displaying graphics using the sixel graphics protocol.

/SOFT_CHARACTERS
/NOSOFT_CHARACTERS

Specifies that the terminal is capable of loading a user-defined character set.

/SPEED=(input-rate,output-rate)

Sets the baud rate at which the terminal receives and transmits data. If the input and output rates are the same, specify */SPEED=rate*.

Not all terminals support different input and output baud rates. For specific information on baud rates for your terminal, consult the manual for that terminal.

The default transmission rates are installation-dependent.

/SWITCH=DECNET

Causes the terminal lines at each node to be switched to dynamic asynchronous DDCMP lines, when specified with the */PROTOCOL=DDCMP* qualifier. Note that */SWITCH=DECNET* is a permanent characteristic; therefore, the */PERMANENT* qualifier is not required.

/SYSPASSWORD
/NOSYSPASSWORD (default)

Requires LOG_IO privilege. Determines whether the terminal requires that a system password be entered before the *Username:* prompt.

DCL-350 DCL Commands
SET TERMINAL

/TAB
/NOTAB

Does not convert tab characters to multiple blanks. The */NOTAB* qualifier expands all tab characters to blanks and assumes tab stops at 8-character intervals.

/TTSYNC (default)
/NOTTSYNC

Stops transmitting to the terminal when CTRL/S is pressed and resumes transmission when CTRL/Q is pressed.

/TYPE_AHEAD (default)
/NOTYPE_AHEAD

Accepts unsolicited input for the terminal to the limit of the type-ahead buffer.

When you specify */NOTYPE_AHEAD*, the terminal is dedicated, and accepts input only when a program or the system issues a read to the terminal. Logins are disabled on a terminal with */NOTYPE_AHEAD* set. When you specify */TYPE_AHEAD*, the amount of data that can be accepted is governed by the size of the type-ahead buffer. That size is determined by system generation parameters.

/UNKNOWN

Specifies a terminal type that is unknown to the system, which then uses the default terminal characteristics for unknown terminals.

/UPPERCASE
/NOUPPERCASE

Translates lowercase to uppercase characters. The */UPPERCASE* qualifier is equivalent to */NOLOWERCASE*.

/WIDTH=characters-per-line

Specifies the maximum characters per line. This value must be an integer in the range 1 through 511. With */WRAP*, the terminal generates a carriage return and line feed when the width specification is reached.

If the specified width on an ANSI terminal is 132, the screen is set to 132-character mode. If the terminal does not have advanced video option (AVO), the page length limit is set to 12 lines.

/WRAP (default)
/NOWRAP

Generates a carriage return and line feed when the value of */WIDTH* is reached.

example

```
$ SET TERMINAL/WIDTH=132/PAGE=60/NOBROADCAST  
$ TYPE MEMO.DOC  
.  
.  
.  
$ SET TERMINAL/DEVICE=LA36
```

In this example, the first SET TERMINAL command indicates that the width of terminal lines is 132 characters and that the size of each page is 60 lines. The /NOBROADCAST qualifier disables the reception of broadcast messages while the terminal is printing the file MEMO.DOC. The next SET TERMINAL command restores the terminal to its default state.

SET TIME

Resets the system clock, which is used both as a timer to record intervals between various internal events, and as a source clock for displaying the time of day.

Requires both OPER and LOG_IO privileges.

format

SET TIME[=*time*]

parameter

time

Specifies a date in the format day-month-year, or a time in the format hour:minute:second.hundredth, or both. Day must be an integer in the range 1 through 31. Month must be JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC. Year must be an integer in the range 1858 through 9999. Hour must be an integer in the range 0 through 23. Minute must be an integer in the range 0 through 59. Second must be an integer in the range 0 through 59. Hundredth (of a second) must be an integer in the range 0 through 99. The hyphens, colons, and period are required delimiters. Delimit the date and time, when both are specified, with a colon.

example

```
$ SET TIME=19-APR-1990:19:31:0.0
```

The SET TIME command in this example sets the date/time at April 19, 1990, 7:31 p.m.

SET UIC

Changes the user identification code (UIC) of your process.

Requires CMKRNL privilege.

format

SET UIC [*uic*]

parameter

uic

Specifies a valid UIC. Square brackets ([]) are required around the UIC.

example

```
$ SET UIC [370,10]
```

The SET UIC command in this example establishes your UIC as [370,10]. You can now read or modify any files whose access is restricted to this UIC.

SET VERIFY

Controls whether command lines and data lines in command procedures are displayed at the terminal or printed in a batch job log. The information displayed by the SET VERIFY command can help you in debugging command procedures.

format

SET [**NO**]VERIFY[=(**[NO]**PROCEDURE, **[NO]**IMAGE)]

parameter

([NO]**PROCEDURE, **[NO]**IMAGE)**

Specifies one or both types of verification. Procedure verification causes each DCL command line in a command procedure to be written to the output device. Image verification causes data lines (input data that is included as part of the SYS\$INPUT input stream) to be written to the output device. By default, both types of verification are set or cleared with SET VERIFY and SET NOVERIFY. If you specify only one keyword, the other is not affected. If you specify only one keyword, omit the parentheses.

description

By default, the SET VERIFY and SET NOVERIFY commands set or clear both types of verification. The default setting for command procedures executed interactively is SET NOVERIFY. System responses and error messages are, however, always displayed. The default for batch jobs is SET VERIFY.

example

```
$ PROC_VER = F$ENVIRONMENT("VERIFY_PROCEDURE")
$ IMAGE_VER = F$ENVIRONMENT("VERIFY_IMAGE")
$ SET NOVERIFY
```

.
.
.

```
$ TEMP = F$VERIFY(PROC_VER, IMAGE_VER)
```

This command procedure uses the lexical function F\$ENVIRONMENT to save the current procedure and image verification setting. Then the SET NOVERIFY command turns off both procedure and image verification. Subsequently, the F\$VERIFY function is used to restore the original verification settings.

SET VOLUME

Changes the characteristics of one or more mounted Files-11 volumes.

Requires WRITE (W) access to the index file on the volume. If you are not the owner of the volume, requires either a system UIC or SYSPRV privilege.

format

```
SET VOLUME device-name[:][,...]
```

parameter

device-name[:][,...]

Specifies the name of one or more mounted Files-11 volumes.

qualifiers

/ACCESSED[=*n*]

Requires OPER privilege. Specifies the number of directories to be maintained in system space for ready access. You can specify a number (*n*) in the range of 0 through 255. If you specify the qualifier */ACCESSED* and omit the number of directories, a default value of 3 is used.

/DATA_CHECK[=(*option*[,*...*])]

Defines a default for data check operations following all reads and writes to the specified volume. (If you do not specify the */DATA_CHECK* qualifier, no checks are made.) Possible keywords are as follows:

READ Performs checks following all read operations

WRITE Performs checks following all write operations (default)

/ERASE_ON_DELETE

/NOERASE_ON_DELETE (default)

Determines whether the space occupied by a file is overwritten with a system specified pattern when a file on the volume is deleted.

/EXTENSION[=*n*]

Specifies the number of blocks to be used as a default extension size for all files on the volume. You can specify a number (*n*) in the range of 0 through 65,535. If you specify the */EXTENSION* qualifier without specifying a value, a default value of 0 (the VMS RMS default) is used.

/FILE_PROTECTION=(*code*)

Sets the default protection to be applied to all files on the specified disk volume. Specify ownership as **SYSTEM**, **OWNER**, **GROUP**, or **WORLD** and access as **R** (READ), **W** (WRITE), **E** (EXECUTE), or **D** (DELETE). A null access specification means no access.

/HIGHWATER_MARKING

/NOHIGHWATER_MARKING

Determines whether the File Highwater Mark (FHM) volume attribute is set. The FHM attribute guarantees that a user cannot read data that was not written by the user. Applies to Structure Level 2 volumes only.

/LABEL=*volume-label*

Specifies a 1- through 12-character alphanumeric name to be encoded on the volume. Characters are automatically changed to uppercase.

/LOG

/NOLOG (default)

Determines whether the volume specification of each volume is displayed after the modification.

/MOUNT_VERIFICATION
/NOMOUNT_VERIFICATION

Determines whether mount verification is enabled. Mount verification prevents interruption to user input/output operations and notifies the operator of problems with the disk.

/OWNER_UIC[=*uic*]

Sets the owner UIC of the volume to the specified UIC. The default UIC is that of the current process. Brackets are required around the UIC.

/PROTECTION=(*code*)

Specifies the protection to be applied to the volume. The ownership categories are SYSTEM, OWNER, GROUP, and WORLD; the access categories are R (READ), W (WRITE), E (EXECUTE), and D (DELETE). The default protection is all types of access by all categories of user.

/REBUILD

Recovers caching limits for a volume that was improperly dismounted. If a disk volume was dismounted improperly (such as during a system failure), and was then remounted with the MOUNT/NOREBUILD command, you can use SET VOLUME/REBUILD to recover the caching that was in effect at the time of the dismount.

/RETENTION=(*min*[,*max*])

Specifies the minimum and maximum retention times to be used by the file system to determine the expiration date for files on the volume. When a file is created, its expiration date is set to the current time + maximum. Each time the file is accessed, the current time is added to the minimum time. If the sum is greater than the expiration date, a new expiration date is computed.

SET VOLUME/RETENTION=0 is the mechanism by which retention times are disabled on the volume.

/UNLOAD (default)

/NOUNLOAD

Specifies whether the volume is unloaded (spun down) when the DCL command DISMOUNT is entered.

/USER_NAME[=*user-name*]

Specifies a user name of up to 12 alphanumeric characters to be recorded on the volume. The default name is the current process user name.

/WINDOWS[=*n*]

Specifies the number of mapping pointers to be allocated for file windows. The value of *n* can be from 7 through 80; the default value is 7.

example

```
$ SET VOLUME/ACCESSED=25/USER_NAME=MANAGER/LOG DBA0:
```

The SET VOLUME command in this example specifies that 25 directories are to be maintained in system space for ready access for the volume DBA0:. The command also assigns the user name MANAGER to the volume and displays the volume specification after the volume is modified.

SET WORKING_SET

Redefines the default working set size for the process, or sets an upper limit to which the working set size can be changed by an image that the process executes. Working set limits cannot be set to exceed those defined in the user authorization file (UAF).

format

SET WORKING_SET

qualifiers

/ADJUST (default)

/NOADJUST

Enables or disables the system's changing of the process working set.

/EXTENT=n

Specifies the maximum number of pages that can be resident in the working set during image execution.

The extent value must be greater than the minimum working set defined at system generation, and it must be less than or equal to the authorized extent defined in the user authorization file.

If you specify a value greater than the authorized extent, the command sets the working set limit at the maximum authorized value.

/LIMIT=n

Specifies the size to which the working set is to be reduced at image exit.

If you specify a value greater than the current quota, the quota value is also increased.

/LOG

/NOLOG (default)

Determines whether or not confirmation of the SET WORKING_SET command is displayed.

/QUOTA=n

Specifies the maximum number of pages that any image executing in the process context can request. An image can set the working set size for the process by calling the Adjust Working Set Limit (\$ADJWSL) system service.

example

```
$ SHOW WORKING_SET
Working Set      /Limit= 150 /Quota= 700           /Extent= 700
Adjustment enabled  Authorized Quota= 700  Authorized Extent= 700
$ SET WORKING_SET/QUOTA=1000
%SET-I-NEWLIMS, new working set:  Limit = 150  Quota = 700  Extent = 700
```

The SHOW WORKING_SET command in this example displays the current limit, quota, and extent, as well as the authorized quota and authorized extent. The SET WORKING_SET command attempts to set a quota limiting the maximum number of pages any image can request that is greater than the authorized quota. Note from the response that the quota was not increased.

SHOW ACCOUNTING

Displays the activities for which accounting is currently enabled. For more information about the Accounting Utility, see the *VMS System Manager's Manual* in the VMS base documentation set.

format

SHOW ACCOUNTING

parameters

None.

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT

Specifies the file to which the display is written; by default, the display is written to the current SYS\$OUTPUT device.

example

```
$ SHOW ACCOUNTING/OUTPUT=ACCOUNTING.SET
```

The SHOW ACCOUNTING command in this example writes the current setting of SET ACCOUNTING to the file ACCOUNTING.SET.

SHOW ACL

Allows you to display the access control list (ACL) of an object.

format

SHOW ACL *object-name*

parameter

object-name

Specifies the name of the object whose ACL is to be displayed. No wildcard characters are allowed in the object-name specification.

qualifier

/OBJECT_TYPE=type

Defines the object type of the object whose ACL is to be displayed. The following keywords are used to specify the object type:

FILE (default)	The object is a Files-11 disk file.
DEVICE	The object is a device.
SYSTEM_GLOBAL_SECTION	The object is a system global section.
GROUP_GLOBAL_SECTION	The object is a group global section.
QUEUE	The object is a batch or device (terminal, server, or printer) queue.
LOGICAL_NAME_TABLE	The object is a system logical name table.

example

```
$ SHOW ACL/OBJECT_TYPE=DEVICE TTA1
Object type: device, Object name: VTA1
(IDENTIFIER=[SALES,FRANK],ACCESS=READ)
(IDENTIFIER=[123,321]+NETWORK,ACCESS=NONE)
```

.
.
.

The SHOW ACL command in this example displays the ACL of the device TTA1.

SHOW AUDIT

Displays the security auditing characteristics in effect on the system.

Requires the SECURITY privilege.

format

SHOW AUDIT

qualifiers

/ALL

Displays all available auditing information including the following: name and location of the system security audit log file; type of security events enabled on the system; action the system will take if an attempt to write an audit event message fails (failure mode); name and location of the security archive file; information about the audit server process, such as the action taken if the audit server process runs out of virtual memory.

/ALARM

Displays the security events currently enabled on the system.

/ARCHIVE

Displays the name and location of the security archive file (if enabled).

/FAILURE_MODE

Displays the failure mode currently in effect on the system.

/JOURNAL

Displays the name and location of the system security audit log file.

/OUTPUT[=*file-spec*]

/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter **/OUTPUT** without a file specification, the output is sent to the current process default output stream or device, identified by the logical name **SYS\$OUTPUT**.

example

```
$ SHOW AUDIT
Security alarm failure mode is set to:
    WAIT          Processes will wait for resource

Security alarms currently enabled for:
BREAKIN:        (DIALUP, LOCAL, REMOTE, NETWORK, DETACHED)
LOGIN:          (DIALUP)
LOGOUT:         (DIALUP)
```

The **SHOW AUDIT** command in this example reveals that the terminals enabled as security operators will receive an alarm whenever the system detects a possible breakin attempt, or when a dialup user logs in or out.

SHOW BROADCAST

Displays the message classes that are currently affected by the SET BROADCAST command.

format

SHOW BROADCAST

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter */OUTPUT* without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS\$OUTPUT.

example

```
$ SHOW BROADCAST
Broadcasts are currently disabled for:
MAIL
```

The SHOW BROADCAST display in this example indicates that SET BROADCAST=NOMAIL is in effect.

SHOW CLUSTER

Invokes the Show Cluster Utility (SHOW CLUSTER) to monitor and display cluster activity and performance.

format

SHOW CLUSTER

SHOW CPU

Displays the current state of the processors in a VMS multiprocessing system.

Applies only to VMS multiprocessing systems. Requires change mode to kernel (CMKRNL) privilege.

format

SHOW CPU *[cpu-id,...]*

parameter

cpu-id

Decimal value representing the identity of a processor in a multiprocessing system. In a VAX 8300 system, for instance, the CPU ID is the VAXBI node number of the processor; in a VAX 8800, the CPU ID of the left processor is 1 and that of the right processor is 0.

description

The SHOW CPU command displays information about the status, characteristics, and capabilities of the processors active in and available to a VMS multiprocessing system.

You identify the processors to be displayed by using either the /ACTIVE qualifier, the /ALL qualifier, a CPU ID, or list of CPU IDs. If you specify none of these, the SHOW CPU command uses the /ALL qualifier by default.

You identify the type of information to be displayed by using the /BRIEF, /FULL, and /SUMMARY qualifiers. If you specify neither the /SUMMARY, /BRIEF, nor /FULL qualifier, SHOW CPU assumes the /BRIEF qualifier by default. However, if you likewise do not identify a processor or processors as the object of a command, SHOW CPU assumes a default of SHOW/ALL/SUMMARY.

The SHOW CPU/FULL command lists the current process on each configured processor without stopping other activity on the system. The current process might change while the data are displayed. As a result, there might be apparent inconsistencies in the display. For example, a process might be listed as the current process on more than one CPU.

qualifiers

/ACTIVE

Selects as the subject of the display only those processors that are members of the system's active set.

/ALL

Selects all configured processors, active and inactive, as the subject of the display.

/BRIEF

Produces information from the summary display and also lists the current CPU state and current process (if any) for each processor in the configuration.

DCL-362 DCL Commands

SHOW CPU

/FULL

Produces information from the summary display. The /FULL qualifier also lists the current CPU state, current process (if any), revision levels, and capabilities for each configured processor. It indicates which processes can execute only on certain processors in the configuration. In addition, if one or more uniprocessing drivers are present in the system, the /FULL qualifier lists them by name.

/SUMMARY

Produces a display listing the processors in the VMS multiprocessing system, indicating which is the primary, which are configured, and which are active. The /SUMMARY qualifier also indicates the minimum revision levels required for processors in the system, which VMS synchronization image has been loaded into the operating system, and whether multiprocessing is enabled. If the presence of one or more uniprocessing drivers in the system prohibits the enabling of multiprocessing, the SHOW CPU command displays a warning message.

SHOW DEFAULT

Displays the current default device and directory.

format

SHOW DEFAULT

example

```
$ SHOW DEFAULT
DISK1: [ALPHA]
$ SET DEFAULT DISK5: [HIGGINS.SOURCES]
$ SHOW DEFAULT
DISK5: [HIGGINS.SOURCES]
```

The SHOW DEFAULT command in this example displays the current default device and directory names. The SET DEFAULT command changes these defaults, and the next SHOW DEFAULT command displays the new default device and directory.

SHOW DEVICES

Displays the status of a device on the system.

See the qualifier descriptions for restrictions.

format

SHOW DEVICES [*device-name[:]*]

parameter

device-name[:]

Specifies the name of a device for which information is to be displayed. You can specify a complete device name or only a portion of a device name.

qualifiers

/ALLOCATED

Displays all devices currently allocated to processes.

/BRIEF (default)

Displays brief information about the specified devices.

/FILES

Requires SYSPRV or BYPASS privileges to list read-protected files. Displays a list of the names of all files open on a volume and their associated process name and process identification (PID). The specified device must be a mounted Files-11 volume. If the specified volume is a multivolume set, the files on each volume in the set are listed. If the /SYSTEM qualifier is also specified, only the names of installed files and files opened by the system are displayed. If the /NOSYSTEM qualifier is specified, only those files opened by processes are displayed.

/FULL

Displays a complete list of information about the devices.

/MOUNTED

Displays all devices that currently have volumes mounted on them.

/OUTPUT[=*file-spec*]

/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS\$OUTPUT.

DCL-364 DCL Commands

SHOW DEVICES

/SYSTEM **/NOSYSTEM**

Controls whether the names of installed files and files opened by the system are displayed.

/WINDOWS

Displays the window count and total size of all windows for files open on a volume. The file name and related process name and process identification (PID) are also displayed. The letter C in a display indicates that the file is open with "cathedral windows" (segmented windows).

example

```
$ SHOW DEVICES/FULL DMA0
Disk NODE1$DMA0:, device type RK07, is online, allocated, mounted,
error logging enabled
Error count                0 Operations completed                1257
Owner UIC                   [1,4] Owner process name                VANNOY
Owner process ID            202000C8 Dev Prot S:RWED,O:RWED,G:RWED,W:RWED
Reference count             2 Default buffer size                512
Volume label                JAKE_X239 Relative volume no.                0
Cluster size                1 Transaction count                2
Free blocks                 3741 Maximum files allowed                13447
Extend quantity             5 Mount count                1
Volume status               Process ACP process name                DMA0BACP
File ID cache size         64 Extent cache size                64
Quota cache size           64
```

Volume is subject to mount verification, file high-water marking

In this example, the SHOW DEVICES command requests a full listing of the status of the RK07 device DMA0. The device is located on NODE1 in a VAXcluster.

SHOW DEVICES/SERVED

Displays information on devices served by the MSCP server on this node. The /SERVED qualifier is required.

format

SHOW DEVICES/SERVED

description

The SHOW DEVICES/SERVED command displays information about the MSCP server and the devices it serves. This information is mostly used by system managers.

qualifiers

/ALL

This qualifier displays the information displayed by all of the qualifiers listed below except the ***/OUTPUT*** qualifier.

/COUNT

Displays the number of transfer operations completed, sorted by the size of the transfers, and the number of MSCP operations that have taken place since the MSCP server was started.

/HOST

Displays the names of the processors that have MSCP-served devices on line. SYSGEN's MSCP/HOST command determines how many hosts in the cluster can connect to the MSCP server at one time.

/OUTPUT=[filespec]

Redirects output from your terminal to the specified file. If you do not specify a file, or if you do not use this qualifier, output is sent to SYS\$OUTPUT.

/RESOURCE

Displays information on the resources available to the MSCP server for use in processing I/O requests for the devices it serves. You make these resources available to the MSCP server when you use SYSGEN's MSCP command to start the MSCP server and use the qualifiers listed in the following table:

Qualifier	Item Specified
<i>/BUFFER</i>	The amount of buffer space available to the MSCP server
<i>/FRACTION</i>	The maximum size, in pages, of the buffer granted to an I/O request; for transfers of more data than will fit a buffer of the size specified by this qualifier, several CI transfers are needed
<i>/SMALL</i>	The minimum size, in pages, of the buffer that the MSCP server can grant to an I/O request; if less than this amount of buffer space is available, the I/O request must wait until at least this much buffer space becomes available; when this much space becomes available, the MSCP server grants the request a buffer
<i>/PACKETS</i>	The number of I/O-request packets (CDRPs) available to the MSCP server for processing I/O requests

DCL-366 DCL Commands
SHOW DEVICES/SERVED

example

```
SHOW DEVICES/SERVED
MSCP Served Devices on BOSTON 19-APR-1990 12:34:56.78
Device:           Status      Total Size      Queue Requests
                  AVAIL        340670          Current   Max       Hosts
2$DBA0            AVAIL        340670          0         0         0
2$DMA1            ONLINE       53790           0         0         2
2$DMA0            OFFLINE      53790           0         0         0
```

This example shows the output generated by the command **SHOW DEVICES/SERVED**. The first column in the display shows the names of the devices that are served by the MSCP server. The second column shows the status of the devices. The third column shows the size, in blocks, of the device.

The *Queue Requests* columns show the number of I/O requests currently awaiting processing by that device and the maximum number of I/O requests that have ever been concurrently awaiting processing by that device. The last column in the display shows the number of hosts that have the device on line.

SHOW DISPLAY

Indicates the node where output from a DECwindows application will be displayed.

format

SHOW DISPLAY [*display-device*]

parameters

display-device

Refers to the *display-device* parameter specified with the **SET DISPLAY** command. If you are directing application output to multiple workstations in the same session, you can use logical names to point to each workstation. Using the **SHOW DISPLAY** command, you can specify this logical name as the *display-device* parameter to see where application output will be displayed.

If you do not specify a *display-device* string, the logical name **DECW\$DISPLAY** is used.

example

```

$ SHOW DISPLAY
Device:      WSA1:
Node:        0
Transport:   LOCAL
Server:      0
Screen:      0

$ SET DISPLAY/CREATE/NODE=ZEPHYR
$ SHOW DISPLAY
Device:      WSA2:
Node:        ZEPHYR
Transport:   DECNET
Server:      0
Screen:      0

$ SPAWN/NOWAIT/INPUT=NL: RUN SYS$SYSTEM:DECW$CLOCK

$ SET DISPLAY/NOPERMANENT

$ SHOW DISPLAY
Device:      WSA1:
Node:        0
Transport:   LOCAL
Server:      0
Screen:      0

```

In this example, you are logged in to your workstation, here referred to as node 0. (0 is the standard shorthand notation for representing your node.) You want to run the DECwindows Clock on your workstation and display it on another node, ZEPHYR.

Assuming you are authorized to display applications on ZEPHYR, you redirect the application's output to ZEPHYR with the SET DISPLAY command and enter the SHOW DISPLAY command to verify the location of the redirected display. You then run Clock. When you finish running Clock, you disable the redirected display by entering the SET DISPLAY/NOPERMANENT command. Finally, you enter the SHOW DISPLAY command to verify that any applications subsequently run on your node will also be displayed there.

Note that a new workstation display device, WSA2, is created when you enter the SET DISPLAY/CREATE command. When you cancel the redirected display with the SET DISPLAY/NOPERMANENT command, application output is once again displayed on the workstation display device referred to by WSA1.

SHOW ENTRY

Displays information about a user's batch and print jobs or about specific job entries.

Requires GROUP privilege to display all jobs in your group.
Requires OPER privilege to display all jobs in all groups.

format

SHOW ENTRY [*entry-number,...*]

parameter

[entry-number,...]

Specifies the entry number of the job you want displayed. If no entry number is specified, all your own jobs (or those owned by the user specified with the /USER_NAME qualifier) are displayed.

qualifiers

/BATCH

Selects batch jobs for display. If /USER_NAME is not specified, information about your own jobs is displayed.

/BRIEF (default)

Displays the following information for each job: job name, user name, entry number, job size in blocks (for print jobs), status, queue name, and queue type. The /FULL and /FILES qualifiers override /BRIEF. Specify the /FULL qualifier to obtain more job information.

/BY_JOB_STATUS[=(keyword,...)]

Selects for display only those jobs with the specified status. Specify the status with one or more of the following keywords:

EXECUTING	Requests the display of currently executing jobs.
HOLDING	Requests the display of jobs on hold. Holding status indicates that the job is being held in the queue indefinitely.
PENDING	Requests the display of jobs with pending status. Pending status indicates that the job is waiting its turn to execute.
RETAINED	Requests the display of jobs retained in the queue after execution. Retained status indicates that the job has completed but remains in the queue. For example, a job may be retained in the queue if there was an error during its execution.
TIMED_RELEASE	Requests the display of jobs on hold until a specified time. Timed release status indicates that the job is being held in the queue for execution at a future time.

If no keyword is specified, /BY_JOB_STATUS displays the status of all jobs.

SHOW ENTRY***/DEVICE[=(keyword,...)]***

Selects for display only those print jobs in the queue types specified. Specify the queue type with one or more of the following keywords:

PRINTER	Requests the display of jobs in print queues.
SERVER	Requests the display of jobs in server queues.
TERMINAL	Requests the display of jobs in terminal queues.

If no keyword is specified, **/DEVICE** displays all printer, terminal, and server queues. If **/USER_NAME** is not specified, information about your own jobs is displayed.

/FILES

Adds to the default display the list of full file specifications for each file in each job.

/FULL

Displays the following information for each job: job name, user name, entry number, job status, full file specification associated with each job, date and time of submission, settings specified for the job, queue name, and queue type.

The **/FULL** qualifier overrides the default brief listing format.

/GENERIC

Selects for display only those jobs contained in generic queues. A generic queue holds jobs of a particular type (for example, batch or line printer jobs) and directs them to execution queues for processing. If **/USER_NAME** is not specified, information about your own jobs is displayed.

/OUTPUT[=filespec]***/NOOUTPUT***

Controls where the output of the **SHOW ENTRY** command is sent. By default, the output is sent to the current **SYS\$OUTPUT** device (usually your terminal). To send the output to a file, use the **/OUTPUT** qualifier followed by a file specification. The file specification may not include any wildcard characters. If you enter a partial file specification (for example, specifying only a directory), **SHOW** is the default file name and **LIS** is the default file type. If you enter **/NOOUTPUT**, output is suppressed.

/USER_NAME=username

Selects for display those jobs owned by the specified user. If **/USER_NAME** is not specified, information about your own jobs is displayed. The name must be 1 to 12 alphanumeric characters.

DCL-370 DCL Commands

SHOW ENTRY

example

```
$ SHOW ENTRY/DEVICE=(PRINTER, TERMINAL)
Jobname      Username    Entry    Blocks    Status
-----
FORECAST     JONES      422      12        Printing
  On printer queue LN01$PRINT

MANAGER      JONES      431      4         Printing
  On terminal queue LQ$PRINT
```

In this example, SHOW ENTRY produces a display of your current job entries on all printer and terminal queues.

SHOW ERROR

Displays the error count for all devices with error counts greater than 0.

format

SHOW ERROR

parameters

None.

qualifiers

/FULL

Displays the error count for all devices, including those with no errors. (The error count is either 0 or a number greater than 0.)

/OUTPUT[=file-spec]

/OUTPUT=SYS\$OUTPUT (default)

Specifies the file to which the display is written. By default, the display is written to the current SYS\$OUTPUT device.

example

```
$ SHOW ERROR
```

Displays the error count for all devices with error counts greater than 0:

```
Device    Error Count
CPU       2
MEMORY    1
DBB1      9
```

SHOW INTRUSION

Displays the contents of the break-in database.

Requires the **CMKRNL** and **SECURITY** privileges.

format

SHOW INTRUSION

qualifiers

/OUTPUT[=file-spec]

Directs the output from the **SHOW INTRUSION** command to the file specified with the qualifier. By default, output from the command is displayed to **SYS\$OUTPUT**.

/TYPE=keyword

Selects the type of information from the break-in database that is displayed. The valid keywords are as follows:

ALL	All break-in entries. By default, all entries are displayed.
SUSPECT	Break-in entries for login failures that have occurred but have not yet passed the threshold necessary to be identified as an intruder.
INTRUDER	Break-in entries for which the login failure rate was high enough to warrant evasive action.

example

```
$ SHOW INTRUSION/TYPE=INTRUDER
```

Intrusion	Type	Count	Expiration	Source
TERMINAL	INTRUDER	9	10:29:39.16	AV34C2/LC-1-15:
NETWORK	INTRUDER	7	10:47:53.12	STAR::HAMM

In this example, the **SHOW INTRUSION** command displays all intruder entries currently in the break-in database.

SHOW KEY

Displays the key definitions created with the **DEFINE/KEY** command.

format

SHOW KEY *[key-name]*

DCL-372 DCL Commands

SHOW KEY

parameter

key-name

The name of the key whose definition you want displayed. See the DEFINE/KEY command for a list of valid key names.

qualifiers

/ALL

Displays all key definitions in the current state (or the state specified with the /STATE qualifier). If you use the /ALL qualifier, do not specify a key name.

/BRIEF (default)

/NOBRIEF

Displays only the key definition and state. The /BRIEF and /NOFULL qualifiers are equivalent.

/DIRECTORY

Displays the names of all states for which keys have been defined.

/FULL

/NOFULL (default)

Displays all qualifiers associated with a definition. By default, only the state of the definition and the definition itself are displayed. The /FULL and /NOBRIEF qualifiers are equivalent.

/STATE=(state-name[...])

/NOSTATE

Displays the key definitions for the specified state. If you specify only one state name, you can omit the parentheses. State names can be any appropriate alphanumeric string. State names are created with the DEFINE/KEY command.

example

```
$ DEFINE/KEY/TERMINATE PF1 "ATTACH GEORGE"
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
$ SHOW KEY PF1
DEFAULT keypad definitions:
  PF1 = "ATTACH GEORGE"
$ SHOW KEY/FULL PF1
DEFAULT keypad definitions:
  PF1 = "ATTACH GEORGE" (noecho,terminate,noerase,nolock)
```

The SHOW KEY command in this example displays both the definition and the state for the PF1 key. This is the default display. The SHOW KEY/FULL command displays all qualifiers associated with the key definition.

SHOW LICENSE

Displays all the software product licenses active on the current node. An active license is one that has been registered in the LICENSE database *and* loaded into system memory. To register and activate software product licenses, use the License Management Utility (LICENSE), or VMSLICENSE.COM. Some licenses are registered automatically during product installation.

For a complete description of this utility, see the *VMS License Management Utility Manual*, part of the VMS Base Documentation Set.

To display licenses registered in the LICENSE database, use the LICENSE LIST command, described with the utility.

format

SHOW LICENSE

parameters

None.

qualifier

/OUTPUT[=*filespec*]
/NOOUTPUT

By default, the output of the SHOW LICENSE command is sent to the current SYS\$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification.

You cannot use any wildcard characters for the file specification. If you enter a partial file specification (for example, specifying only a directory), SHOW is the default file name and LIS is the default file type.

If you enter the /NOOUTPUT qualifier, output is suppressed.

example

```
$ SHOW LICENSE
```

```
Active licenses on node WTPOOH:
```

```
DVNETEND
  Producer: DEC
  Units: 0
  Version: 5.2
  Date: (none)
  Termination Date: (none)
  Availability: E (System Integrated Products)
  Activity: 0
  MOD_UNITS
```

DCL-374 DCL Commands

SHOW LICENSE

```
VAX-VMS
  Producer: DEC
  Units: 0
  Version: 5.2
  Date: (none)
  Termination Date: (none)
  Availability: A (VMS Capacity)
  Activity: 0
  MOD_UNITS
  NO_SHARE
```

The SHOW LICENSE command in this example displays all the active licenses on the current node named WTPOOH.

SHOW LOGICAL

Displays translations, the level of translation, and the logical name table for a specified logical name. The SHOW LOGICAL command performs iterative translations.

Requires READ (R) access to the table in which a logical name is cataloged to display information about the logical name.

format

```
SHOW LOGICAL [logical-name[:][,...]]
```

parameter

logical-name[:][,...]

Specifies one or more logical names whose translations you want to display. The asterisk (*) and percent (%) wildcard characters are allowed. However, if a wildcard character is used, iterative translation is not done.

qualifiers

/ACCESS_MODE=mode

Displays names defined in the specified access mode and any inner access modes. You can specify one of the following keywords to indicate the access mode: USER_MODE, SUPERVISOR_MODE, EXECUTIVE_MODE, or KERNEL_MODE.

/ALL (default)

Indicates that all logical names in the specified logical name tables are to be displayed.

/DESCENDANTS

/NODESCENDANTS (default)

Controls whether the system displays names from the specified logical name table and any descendant tables. A descendant table is created by the CREATE/NAME_TABLE command, with the /PARENT_TABLE

qualifier specifying its parent table. If you use the /DESCENDANTS qualifier, you must also use the /TABLE qualifier.

/FULL

Displays more detailed information for the specified logical name. The information includes the access mode, attributes, the translation, and the logical name table.

/GROUP

Indicates that only the group logical name table is to be searched. The /GROUP qualifier is synonymous with /TABLE=LNМ\$GROUP. If you specify the /GROUP qualifier and you do not also specify a logical name, all names in the group table are displayed.

/JOB

Indicates that only the job logical name table is to be searched. The /JOB qualifier is synonymous with /TABLE=LNМ\$JOB. If you specify the /JOB qualifier and you do not also specify a logical name, all names in the job logical name table are displayed.

/OUTPUT[=file-spec]

/NOOUTPUT

By default, the output of the SHOW LOGICAL command is sent to the current SYS\$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

/PROCESS

Indicates that only the process logical name table is to be searched. The /PROCESS qualifier is synonymous with /TABLE=LNМ\$PROCESS. If you specify the /PROCESS qualifier and you do not also specify a logical name, all names in the process table are displayed.

/STRUCTURE

/NOSTRUCTURE (default)

Controls whether the system displays the “family tree” of all accessible logical name tables. If you specify /STRUCTURE, you cannot use any other qualifiers except /ACCESS_MODE, /FULL, and /OUTPUT.

/SYSTEM

Indicates that only the system logical name table is to be searched. The /SYSTEM qualifier is synonymous with /TABLE=LNМ\$SYSTEM. If you specify the /SYSTEM qualifier and you do not also specify a logical name, all names in the system table are displayed.

/TABLE=(name[,...])

Specifies the tables you want to search. If you specify only one table, you can omit the parentheses. Wildcards are allowed. Names with wildcards are used to match table names. Names without wildcards are treated both as table names and table search lists (whichever is appropriate).

DCL-376 DCL Commands

SHOW LOGICAL

example

```
$ SHOW LOGICAL/PROCESS
(LNM$PROCESS_TABLE)
"SYS$COMMAND" = "_TTB4:"
"SYS$DISK" = "WORK6:"
"SYS$DISK" = "WORK6:"
"SYS$ERROR" = "_TTB4:"
"SYS$INPUT" = "_TTB4:"
"SYS$LOGIN" = "WORK6:[ODONNELL]"
"SYS$LOGIN_DEVICE" = "WORK6:"
"SYS$OUTPUT" = "_TTB4:"
"SYS$OUTPUT" = "DBA2:"
"SYS$SCRATCH" = "WORK6:[ODONNELL]"
```

The SHOW LOGICAL command in this example displays all process logical names and their translations. (Note that /TABLE=LNM\$PROCESS would produce the same display as /PROCESS.)

SHOW MAGTAPE

Displays the current characteristics and status of a specified magnetic tape device.

format

```
SHOW MAGTAPE device-name[:]
```

parameter

device-name[:]

Specifies the name of the magnetic tape device for which you want to display the characteristics and status.

qualifier

/OUTPUT[=file-spec]

/NOOUTPUT

Specifies the file to which the display is written; by default, the display is written to the current SYS\$OUTPUT device.

example

```
$ SHOW MAGTAPE MTA0:
MTA0: UNKNOWN, DENSITY=800, FORMAT=Normal-11
      Odd Parity
```

The SHOW MAGTAPE command in this example displays the characteristics of the device MTA0:. The display shows the device type, density, and format (default or normal PDP-11).

It also displays the following characteristics:

Position lost	Write-locked
End-of-tape	Even parity
End-of-file	Odd parity
Beginning-of-tape	

SHOW MEMORY

Displays the availability and usage of those system resources that are related to memory.

format

SHOW MEMORY

qualifiers

/ALL (default)

Displays all available information, that is, information displayed by the ***/FILES***, ***/PHYSICAL_PAGES***, ***/POOL***, and ***/SLOTS*** qualifiers.

/FILES

Displays information about the use of each paging and swap file currently installed.

/FULL

When used with the ***/POOL*** or ***/FILES*** qualifier, displays additional information about the use of each pool area or paging and swap file currently installed. This qualifier is ignored unless the ***/FILES*** or ***/POOL*** qualifier is explicitly specified.

/OUTPUT[=file-spec]

/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter ***/OUTPUT*** without a file specification, the output is sent to the current process default output stream or device, identified by the logical name ***SYS\$OUTPUT***.

/PHYSICAL_PAGES

Displays information about the amount of physical memory and the number of free and modified pages.

/POOL

Displays information about the usage of each dynamic memory (pool) area, including the amount of free space and the size of the largest contiguous block in each area.

/SLOTS

Displays information about the availability of PCB vector slots and balance slots.

DCL-378 DCL Commands

SHOW MEMORY

example

```
$ SHOW MEMORY/SLOTS
```

```
System Memory Resources on 19-APR-1990 16:11:35.31
Slot Usage (slots):          Total1      Free2    Resident3    Swapped4
Process Entry Slots         75          28        46           1
Balance Set Slots           70          26        44           0
```

Slot Usage (slots)

Displays the use of process entry slots and balance slots.

- ¹Total Displays the number of process entry slots (the value of the SYSGEN parameter MAXPROCESSCNT) and balance slots (the value of the SYSGEN parameter BALSETCNT) permanently allocated when the system was bootstrapped.
- ²Free Displays the number of slots currently not in use.
- ³Resident Displays the number of slots currently used by memory-resident processes. The number of balance slots in use can never be any larger than the number of process entry slots in use because the SWAPPER and NULL processes have process entry slots but do not require balance slots.
- ⁴Swapped Displays the number of slots used by outswapped processes. For process entry slots, this number includes all processes that have been partially outswapped. For balance slots, this number includes those processes that have had their process bodies outswapped but have process headers that are still resident.

SHOW NETWORK

Displays the availability of the local node as a member of the network¹ and the addresses and names of all nodes that are currently accessible to the local node. The SHOW NETWORK command also displays link and cost relationships between the local node and other nodes in the network.

format

```
SHOW NETWORK
```

qualifier

```
/OUTPUT[=file-spec]  
/NOOUTPUT
```

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified

¹ DECnet-VAX is available under separate license.

by the logical name SYS\$OUTPUT. If you enter /OUTPUT with a partial file specification (for example, specifying only a directory), SHOW is the default file name and LIS is the default file type. If you enter a file specification, it may not include any wildcard characters. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW NETWORK
```

```
VAX/VMS Network Status for local node 2.161 ARAKIS on 19-APR-1990 09:18:03.07
The next hop to the nearest area router is node 2.62 ZEUS.
```

Node	Links	Cost	Hops	Next Hop to Node
2.161 ARAKIS	0	0	0	Local -> 2.161 ARAKIS
2.1 RAEL	0	8	1	UNA-0 -> 2.1 RAEL
2.2 PANGEA	0	8	1	UNA-0 -> 2.2 PANGEA
2.3 TWDEE	0	10	2	UNA-0 -> 2.63 AURORA
2.4 TWDUM	0	8	1	UNA-0 -> 2.4 TWDUM
2.11 NEONV	0	8	1	UNA-0 -> 2.11 NEONV
2.63 AURORA	0	8	1	UNA-0 -> 2.63 AURORA

Total of 7 nodes.

If your local node is a nonrouting or end node and you enter the SHOW NETWORK command, the following message is displayed:

```
This is a nonrouting node, and does not have any network
information. The designated router for node _nodename is
node_number_name.
```

SHOW PRINTER

Displays the current settings for a printer.

format

```
SHOW PRINTER device-name[:]
```

parameter

device-name[:]

Specifies the name of the printer for which settings are to be displayed.

qualifier

/OUTPUT[=*file-spec*]

/NOOUTPUT

By default, the output of the SHOW PRINTER command is sent to the current SYS\$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

DCL-380 DCL Commands

SHOW PRINTER

example

```
$ SHOW PRINTER LPA0:
Printer LPA0:, device type LP11, is online, allocated, spooled
Error count          0 Operations completed          880
Owner process "SYMBIONT_0001" Owner UIC              [0,0]
Owner process ID     21C0008D Dev Prot S:RWLP,O:RWLP,G:RWLP,W:RWLP
Reference count      2 Default buffer size          132
Page width           132 Page Length                66
No Carriage_return  Formfeed          Lowercase
No Passall          No Wrap            Printall
No Fallback
Intermediate device: STAR$DBA1:
Associated queue: LN01$PRINT
```

The SHOW PRINTER command in this example displays the settings for the printer LPA0.

SHOW PROCESS

Displays information about a process and its subprocesses. If no qualifier is entered, only a subset of information is displayed: the time, the process terminal, the user name and UIC, the node name, the process name and the process identification, priority, default directory, and allocated devices.

Requires GROUP privilege to show other processes in the same group. Requires WORLD privilege to show processes outside your group.

format

```
SHOW PROCESS [[node-name:]process-name]
```

parameter

process-name

Specifies the name of the process about which information is to be displayed. Process names can be up to 23 alphanumeric characters long in the following format:

```
[node-name:]process-name
```

- The node name can have as many as 6 alphanumeric characters.
- The colons count for 2 characters.
- The process name can have as many as 15 characters.

A local process name can look like a remote process name. Therefore, if you specify ATHENS::SMITH, the system checks for a process named ATHENS::SMITH on the local node before checking node ATHENS for a process named SMITH.

Process names are linked to group numbers. The specified process must have the same group number in its user identification code (UIC) as the current process.

qualifiers

/ACCOUNTING

Displays the accumulated accounting statistics for the process.

/ALL

Displays the basic subset of information as well as accounting statistics, privileges, quotas, and subprocesses. Displays memory use for the current process.

/CONTINUOUS

Displays continuously updated information about the local process in a VAXcluster environment. You cannot use the */CONTINUOUS* qualifier to display information about a process on another node in a VAXcluster environment.

While the continuous display is running, you can press the V key to display a map of the pages in the virtual address space of the process. To terminate the continuous display, press the E key. To return to the original display, press the space bar. The */CONTINUOUS* qualifier may not be used with the */OUTPUT* qualifier.

/IDENTIFICATION=pid

Requires GROUP or WORLD privilege to access processes other than your own. Displays information about the process with the specified PID (process identification). The PID is assigned by the system when the process is created. When you specify a PID, you can omit the leading zeros. If you specify the */IDENTIFICATION* qualifier, you cannot use the process name parameter. If, in addition, you specify the */MEMORY* qualifier, the PID value must be that of the current process.

/MEMORY

Displays the process's use of dynamic memory areas. The */MEMORY* qualifier is allowed only for the current process.

/OUTPUT[=file-spec]

/NOOUTPUT

By default, the output of the SHOW PROCESS command is sent to the current SYS\$OUTPUT device (usually your terminal). To send the output to a file, use the */OUTPUT* qualifier followed by a file specification. If you enter */NOOUTPUT*, output is suppressed. The */OUTPUT* qualifier may not be used with the */CONTINUOUS* qualifier.

/PRIVILEGES

Displays current privileges for the process.

DCL-382 DCL Commands

SHOW PROCESS

/QUOTAS

Displays, for each resource, either a quota or a limit. The values displayed for quotas reflect any quota reductions resulting from subprocess creation. The values displayed for limits reflect the resources available to a process at creation.

/SUBPROCESSES

Displays the current subprocesses in hierarchical order.

example

```
$ SHOW PROCESS
19-APR-1990 15:35:19.39  User: MALIK   Process ID: 28200364
                          Node: OCALA   Process name: MALIK
Terminal:                 RTA5:
User identifier: [VMS,MALIK]
Base priority:           4
Default file spec: WORK5:[MALIK]
Devices allocated: RTA5:
```

The SHOW PROCESS command in this example is entered on NODE ATHENS by the user MALIK. The system displays the subset of information for the owned process on node OCALA.

SHOW PROTECTION

Displays the current file protection to be applied to all new files created during the terminal session or batch job. You can change the default protection at any time with the SET PROTECTION command.

format

SHOW PROTECTION

example

```
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
$ SET PROTECTION=(GROUP:RWED, WORLD:RE)/DEFAULT
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RWED, WORLD=RE
```

The SHOW PROTECTION command in this example requests a display of the current protection defaults and the user identifiers; the SET PROTECTION/DEFAULT command changes the file access allowed to other users in the same group and to miscellaneous system users. The next SHOW PROTECTION command shows the modified protection defaults.

SHOW QUEUE

Displays information about queues and the jobs that are currently in queues.

Requires GROUP privilege to show all jobs in your group.
Requires OPER privilege to show all jobs in all groups.

format

SHOW QUEUE *[queue-name]*

parameter

queue-name

Specifies the name of the queue for which you want information displayed. Wildcard characters (* and %) are allowed. The default value for the queue-name parameter is the asterisk wildcard (*). If no queue name is specified, information on all queues is displayed.

qualifiers

/ALL_JOBS

Displays all the jobs in the specified queues. If you do not specify a queue name, the ***/ALL_JOBS*** qualifier displays all job entries on all queues. To modify the display, combine this qualifier with the ***/BY_JOB_STATUS*** qualifier.

/BATCH

Displays all batch queues. Use the ***/BATCH*** qualifier in conjunction with other qualifiers to display specific information about particular batch queues.

/BRIEF

Displays a one line description of each queue and the jobs that are in it. This information includes the name, type, and status of each queue. It also includes the user name, entry number, and status for each job. The ***/FULL*** and ***/FILES*** qualifiers override ***/BRIEF***.

/BY_JOB_STATUS=(keyword-list)

Displays jobs with the specified status. Specify the status with one or more of the following keywords:

EXECUTING	Requests the display of currently executing jobs.
HOLDING	Requests the display of jobs on hold. Holding status indicates that the job is being held in the queue indefinitely.
PENDING	Requests the display of jobs with pending status. Pending status indicates that the job is waiting its turn to execute.

DCL-384 DCL Commands

SHOW QUEUE

RETAINED	Requests the display of jobs retained in the queue after execution. Retained status indicates that the job has completed, but it remains in the queue. For example, a job may be retained in the queue if there was an error during its execution.
TIMED_RELEASE	Requests the display of jobs on hold until a specified time. Timed release status indicates that the job is being held in the queue for execution at a future time.

Note that if you specify the qualifier without a keyword, the system will only display queues that actually contain jobs.

/DEVICE[=(keyword-list)]

Displays a particular type of queue. Use the /DEVICE qualifier in conjunction with other qualifiers to display specific information about particular device queues.

Specify the type of device queue with one or more of the following keywords:

PRINTER	Requests the display of all print queues.
SERVER	Requests the display of all server queues.
TERMINAL	Requests the display of all terminal queues.

You can specify more than one keyword. If you do not specify a keyword, /DEVICE displays all printer, terminal, and server queues.

/FILES

Adds to the display the list of files associated with each job.

/FULL

Displays complete information about queues, the jobs contained in queues, and the files associated with the jobs. See /BRIEF.

/GENERIC

Displays all generic queues. A generic queue is not an execution queue. Its function is to hold jobs of a particular type (line printer jobs, for example) and direct them to execution queues for processing.

Use the /GENERIC qualifier in conjunction with other qualifiers to display specific information about particular generic queues. For example, use the /GENERIC qualifier along with the /BATCH qualifier to specify information about generic batch queues. Use the /GENERIC qualifier along with the /DEVICE qualifier to determine information concerning generic output queues.

/OUTPUT[=file-spec] ***/NOOUTPUT***

By default, the output of the SHOW QUEUE command is sent to the current SYS\$OUTPUT device (usually your terminal). To send the output

to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

/SUMMARY

Displays the total number of executing jobs, pending jobs, holding jobs, retained jobs, and timed release jobs for each queue. For output queues, the total block count for pending jobs is also shown.

example

```
$ SHOW QUEUE/FULL CAXTON_LPA0
```

```
Printer queue CAXTON_LPA0, on CAXTON::CAXTON_LPA0, mounted form
80_COLS (stock=BLUE)
  /BASE_PRIORITY=100
  /DEFAULT=(FEED,FLAG,FORM=40_COLS (stock=WHITE),TRAILER=ONE)
  /NOENABLE_GENERIC Lowercase /OWNER=[1,4] /PROTECTION=(S:E,O:D,G:R,W:W)
```

Jobname	Username	Entry	Blocks	Status
-----	-----	-----	-----	-----
ACCOUNT	MARTIN	880	10	Printing
Submitted	9-AUG-1988 12:49	/FORM=80_COLS (stock=BLUE) /PRIORITY=100		
REPORT	MARTIN	858	4	Pending
Submitted	8-AUG-1988 17:27	/PRIORITY=100		

The SHOW QUEUE command in this example lists any current job entry you have on the printer queue CAXTON_LPA0. The /FULL qualifier lists the submission information, the full file specification, and the current settings for both the job and the queue.

SHOW QUEUE/CHARACTERISTIC

Displays information about queue characteristics defined for the system. A characteristic is a user-defined attribute of a batch or output queue, such as ink color.

format

SHOW QUEUE/CHARACTERISTIC

[characteristic-name]

parameter

characteristic-name

Specifies the name of a characteristic. Wildcard characters (* and %) are allowed. The default value for the characteristic-name parameter is the asterisk wildcard (*). Thus, information about all characteristics is displayed when you do not specify a characteristic name.

qualifier

/OUTPUT[=filespec]
/NOOUTPUT

By default the output of the SHOW QUEUE/CHARACTERISTIC command is sent to the current SYS\$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW QUEUE/CHARACTERISTIC *INK
Characteristic name          Number
-----
REDINK                       0
BLUEINK                      6
BROWNINK                     25
```

The SHOW QUEUE/CHARACTERISTIC command in this example displays the name and number of all characteristics that end with INK.

SHOW QUEUE/FORM

Displays information about forms defined for the system. Forms define the size and type paper and the layout of text that are used for print jobs.

format

SHOW QUEUE/FORM *[form-name]*

parameter

form-name

Specifies the name of the form. Wildcard characters are allowed. The default value for the form-name parameter is an asterisk (*) which means that the names of all forms on the system are displayed.

qualifiers

/BRIEF (default)

Displays a brief description (form names, numbers, and descriptions) about the forms on the system.

/FULL

Displays a full description (including paper size and margin settings) about the forms on the system.

/OUTPUT[=file-spec]
/NOOUTPUT

By default the output of the SHOW QUEUE/FORM command is sent to the current SYS\$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification. If you enter /NOOUTPUT, output is suppressed.

example

```
$ SHOW QUEUE/FORM/FULL
Form name                               Number   Description
-----
132_51_STD (stock=DEFAULT)              102     132 by 51 (standard short)
      /LENGTH=51 /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /TRUNCATE /WIDTH=132
40_66_STD (stock=DEFAULT)               103     40 by 66 (standard labels)
      /LENGTH=66 /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /WIDTH=40
BLUE_PAPER_STOCK (stock=DIGITAL_8X11_STOCK1412TEA)
      22222     blue paper, DEC order# 22222
      /LENGTH=66 /MARGIN=(BOTTOM=6) /STOCK=DIGITAL_8X11_STOCK1412TEA
      /TRUNCATE /WIDTH=80
DEFAULT                                 0       System-defined default
      /LENGTH=66 /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /TRUNCATE /WIDTH=132
LN01_LANDSCAPE (stock=DEFAULT)          105     132 by 66 (landscape)
      /LENGTH=66 /STOCK=DEFAULT /WIDTH=132
LN01_LANDSCAPE_INDENTED (stock=DEFAULT)
      107     132 by 65 (landscape)
      /LENGTH=65 /SETUP=(LN01_TOP_MARGIN_150) /STOCK=DEFAULT /WIDTH=132
LN01_PORTRAIT (stock=DEFAULT)           106     80 by 60 (portrait)
      /LENGTH=60 /SETUP=(LN01_PORTRAIT) /STOCK=DEFAULT /WIDTH=80
MEMO (stock=DEFAULT)                   110     LN03 indented memo format
      /LENGTH=64 /MARGIN=(TOP=2,LEFT=5) /STOCK=DEFAULT /TRUNCATE /WIDTH=80
```

This SHOW QUEUE/FORM command also displays the names of all form types and stock for the system. By using the /FULL qualifier, you can see what image size has been set for each form type.

SHOW QUOTA

Displays the current disk quota that is authorized for a specific user on a specific disk. This display includes a calculation of the amount of space available and the amount of overdraft that is permitted.

Requires READ (R) access to the quota file in order to display the quotas of other users.

format

SHOW QUOTA

DCL-388 DCL Commands
SHOW QUOTA

qualifiers

/DISK[=*device-name*::]

Specifies the disk whose quotas are to be examined. By default, the current default disk (defined by SYS\$DISK) is examined.

/USER=*uic*

Specifies which user's quotas are to be displayed. By default, the current user's quotas are displayed.

example

```
$ SHOW QUOTA /USER=[360,007]/DISK=XXX1:  
%SYSTEM-F-NODISKQUOTA, no disk quota entry for this UIC
```

The SHOW QUOTA command in this example displays the fact that the user with UIC [360,007] has no disk quota allocation on device XXX1.

SHOW RMS_DEFAULT

Displays the current default values for the multiblock count, the multibuffer count, the network transfer size, the prolog level, and the extend quantity.

format

SHOW RMS_DEFAULT

parameters

None.

qualifier

/OUTPUT[=*file-spec*]

/NOOUTPUT

Specifies the file to which the display is written (default is SYS\$OUTPUT). Wildcard characters are not allowed.

example

```
$ SHOW RMS_DEFAULT
```

	MULTI- BLOCK COUNT		MULTIBUFFER COUNTS						NETWORK BLOCK COUNT
			Indexed	Relative	Disk	Sequential Magtape	Unit Record		
Process	0		0	0	0	0	0		0
System	16		0	0	0	0	0		8
			Prolog	Extend	Quantity				
Process	0		0		0				
System	0		0		0				

The **SHOW RMS_DEFAULT** command in this example shows a system multiblock count of 16 and a network block count of 8. These are typical values.

SHOW STATUS

The **SHOW STATUS** command in this example displays the current status of your process.

format

SHOW STATUS

parameters

None.

example

```
$ SHOW STATUS
Status on 19-APR-1990 12:56:48.68           Elapsed CPU : 0 00:00:55.02
Buff. I/O : 5117   Cur. ws. : 300           Open files : 1
Dir. I/O : 458   Phys. Mem. : 162           Page Faults : 8323
```

Displays the status of your process. The information includes the following:

- Current time and date
- Elapsed CPU time used by the current process
- Buffered I/O count
- Current working set size
- Open file count
- Direct I/O count
- Current amount of physical memory occupied
- Number of page faults

SHOW SYMBOL

Displays the value of the specified symbol.

format

SHOW SYMBOL [*symbol-name*]

parameter

symbol-name

Specifies the name of the symbol whose value you want to display. You must specify a symbol name unless you use the /ALL qualifier. Wildcard characters are allowed in the symbol-name parameter.

qualifiers

/ALL

Displays the current values of all symbols in the specified symbol table (/LOCAL or /GLOBAL). If you specify /ALL and do not specify either /LOCAL or /GLOBAL, the SHOW SYMBOL command displays the contents of the local symbol table for the current command level.

/GLOBAL

Searches only the global symbol table for the specified symbol name. If you specify both the /ALL and /GLOBAL qualifiers, all names in the global symbol table are displayed.

/LOCAL

Searches only the local symbol table for the current command level for the specified symbol name. If you specify both the /ALL and /LOCAL qualifiers, all names in the local symbol table for the current command level are displayed.

/LOG (default)

/NOLOG

Controls whether the system generates an informational message if the symbol value has been truncated. The value is truncated if it exceeds 255 characters.

example

```
$ SHOW SYMBOL/GLOBAL/ALL
TIME == "SHOW TIME"
LOG == "@LOG"
$RESTART == "FALSE"
$SEVERITY == "1"
$STATUS == "%X00000001"
```

The SHOW SYMBOL command in this example displays all the symbols defined in the global symbol table. Note that the symbols \$RESTART, \$STATUS, and \$SEVERITY, which are maintained by the system, are also displayed.

SHOW SYSTEM

Displays status information about current processes: the time, process name and identification, processing state, priority, total process I/O, cumulative processor time used, cumulative page faults, amount of physical memory being used, and type of process.

format

SHOW SYSTEM

parameters

None.

qualifiers

/BATCH

Displays all batch jobs for the local system. When used with ***/CLUSTER***, displays all batch jobs in the VAXcluster environment.

/CLUSTER

Displays all processes on all nodes in a VAXcluster.

/FULL

Displays the user identification code (UIC) in addition to the default information. The UIC is displayed underneath the process name.

/NETWORK

Displays all network processes in the system.

/NODE[=(name,...)]

Displays all the processes on the specified node or nodes. If you enter ***/NODE*** without a value, the qualifier displays all the processes on the local node of a VAXcluster environment.

/OUTPUT[=file-spec]

/NOOUTPUT

By default, the output of the **SHOW SYSTEM** command is sent to the current **SYS\$OUTPUT** device (usually your terminal). To send the output to a file, use the ***/OUTPUT*** qualifier followed by a file specification. If you enter ***/NOOUTPUT***, output is suppressed.

/PROCESS (default)

Displays all processes in the system.

/SUBPROCESS

Displays all subprocesses in the system.

DCL-392 DCL Commands

SHOW SYSTEM

example

```
$ SHOW SYSTEM
VAX/VMS 5.2 on node KRYPTON 19-APR-1990 17:45:47.78 Uptime 2 21:53:59
  Pid   Process Name   State  Pri    I/O      CPU   Page flts Ph.Mem
27400201 SWAPPER        HIB    16     0 0 00:29:52.05    0    0
27401E03 DOCBUILD      LEF    4   37530 0 00:05:47.62  96421  601
27402604 BATCH_789     LEF    4   3106 0 00:00:48.67   4909  2636B
27401C05 BATCH_60    LEF    6    248 0 00:00:06.83   1439  1556B
27400207 ERRFMT       HIB    8   6332 0 00:00:41.83    89   229
27400208 CACHE_SERVER   HIB   16   2235 0 00:00:05.85    67   202
27400209 CLUSTER_SERVER HIB    8   4625 0 00:22:13.28   157   448
2740020C JOB_CONTROL   HIB   10  270920 0 01:07:47.88  5163  1384
2740020D CONFIGURE    HIB    9    125 0 00:00:00.53   104   264
.
.
27400E8D Sir Lancelot    LEF    5     226 0 00:00:07.87   4560  697
2740049A STILES        LEF    4     160 0 00:00:02.69    534  477
27401EA0 BATCH_523   CUR   4 4 17470 0 03:25:49.67   8128  5616 B
274026AF YURYAN        CUR   6 4 14045 0 00:02:03.24  20032  397
274016D5 WEINER     LEF    6     427 0 00:00:09.28   5275  1384
27401ED6 BULMER_1      HIB    5     935 0 00:00:10.17   3029  2204S
274012D7 BATCH_689  LEF    4   49216 0 00:14:18.36   7021  3470 B
274032D9 DECS$MAIL     LEF    4     2626 0 00:00:51.19   4328  3087 B
274018E3 SERVER_0021 LEF    6     519 0 00:00:07.07   1500  389N
274016E8 NMAIL_0008    HIB    4   10955 0 00:00:55.73   5652  151
274034EA KAIKOW    LEF    4     2132 0 00:00:23.85   5318  452
274022EB S. Whiplash    CUR   6 4 492 0 00:00:12.15   5181  459
274018EF DwMail       LEF    5  121386 0 00:28:00.97  7233  4094
27401AF0 EMACSS$RTA43 LEF    4   14727 0 00:03:56.54   8411  4224 S
27400CF4 CULVER       HIB    5   25104 0 00:06:07.76  37407  1923
274020F5 GILLIAM      LEF    7   14726 0 00:02:10.74  34262  1669
27400CF6 mr. mike      LEF    9   40637 0 00:05:15.63  18454  463
```

The SHOW SYSTEM command in this example displays all processes on the system.

The information in this example includes the following:

- Process identification code (PID)—A 32-bit binary value that uniquely identifies a process.
- Process name—A 1- to 15-character string used to identify a process.
- Process state—The activity level of the process, such as COM (computing), HIB (hibernation), LEF (local event flag) wait, or CUR (if the process is current). If a multiprocessing environment exists, the display shows the CPU ID of the processor on which any current process is executing.

Note that the SHOW SYSTEM command examines the processes on the system without stopping activity on the system. In this example process information changed during the time that SHOW SYSTEM collected the data to be displayed. As a result, this display includes

two processes, named YURYAN and S. Whiplash, with the state CUR on the same CPU, CPU ID 6 in the example.

- Current priority—The priority level assigned to the process (the higher the number, the higher the priority).¹
- Total process I/O count¹—The number of I/O operations involved in executing the process. This consists of both the direct I/O count and the buffered I/O count.
- Charged CPU time¹—The amount of CPU time that a process has used thus far.
- Number of page faults¹—The number of exceptions generated by references to pages which are not in the process's working set.
- Physical memory occupied¹—The amount of space in physical memory that the process is currently occupying.
- Process indicator—Letter B indicates a batch job; letter S indicates a subprocess; letter N indicates a network process.
- User identification code (UIC)—An 8-digit octal number assigned to a process. This is only displayed if the /FULL qualifier is specified.

```
$ SHOW SYSTEM /CLUSTER
VAX/VMS V5.2 on node ALPES 19-APR-1990 09:09:58.61 Uptime 0 2:27:11
Pid      Process Name   State Pri I/O  CPU              Page flts Ph. Mem
31E00041 SWAPPER        HIB  16  0  0 00:00:02.42      0          0
31E00047 CACHE_SERVER   HIB  16  58  0 00:00:00.26      80         36
31E00048 CLUSTER_SERVER CUR   9 156  0 00:00:58.15    1168        90
31E00049 OPCOM         HIB   7 8007  0 00:00:33.46    5506       305
31E0004A AUDIT_SERVER   HIB   9  651  0 00:00:21.17    2267        22
31E0004B JOB_CONTROL    HIB  10 1030  0 00:00:11.02     795        202
```

The SHOW SYSTEM command in this example shows all processes on all nodes of the cluster.

¹ This information is displayed only if the process is currently in the balance set; if the process is not in the balance set, these columns contain the following message:

DCL-394 DCL Commands

SHOW SYSTEM

```
$ SHOW SYSTEM /NODE=EON
VAX/VMS V5.2 on node EON 19-APR-1990 09:19:15.33 Uptime 0 02:29:07
Pid      Process Name  State  Pri  I/O  CPU      Page flts Ph. Mem
36200041 SWAPPER      HIB    16   0   0 00:00:12.03 0      0
36200046 ERRFMT       HIB    8   263 0 00:00:05.89 152    87
36200047 CACHE_SERVER CUR    16   9   0 00:00:00.26 80     51
36200048 CLUSTER_SERVER CUR    8   94  0 00:00:30.07 340    68
36200049 OPCOM        HIB    6  2188 0 00:02:01.04 1999   177
3620004A AUDIT_SERVER HIB    10  346  0 00:00:10.42 1707   72
```

The SHOW SYSTEM command in this example shows all processes on the node EON.

SHOW TERMINAL

Displays the current characteristics of a specific terminal. Each characteristic corresponds to an option of the SET TERMINAL command.

format

```
SHOW TERMINAL [device-name[:]]
```

parameter

device-name[:]

Specifies the name of the terminal for which you want the characteristics displayed. The default is your terminal (SYS\$COMMAND).

qualifiers

/OUTPUT[=file-spec]

/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS\$OUTPUT. If you enter /OUTPUT with a partial file specification (for example, specifying only a directory), SHOW is the default file name and LIS the default file type. If you enter a file specification, it may not include any wildcard characters. If you enter /NOOUTPUT, output is suppressed.

/PERMANENT

Requires LOG_IO or PHY_IO privilege. Displays the permanent characteristics of the terminal.

example

```
$ SHOW TERMINAL
Terminal: _TTE4:            Device_Type: VT102            Owner: FRANKLIN
Physical Terminal: _LTA49
  Input:  9600            LFill:  0            Width:  80            Parity: None
  Output: 9600            CRfill: 0            Page:   24
Terminal Characteristics:
  Interactive            Echo                    Type_ahead            No Escape
  No Hostsync            Ttsync                 Lowercase             Tab
  Wrap                    Scope                   No Remote             Eightbit
  Broadcast              No Readsynchron        No Form                Fulldup
  No Modem                No Local_echo          No Autobaud            Hangup
  No Brdcstmbx            No DMA                  No Altypeahd           Set_speed
  Line Editing            Overstrike editing    No Fallback            No Dialup
  No Secure server        No Disconnect          No Psthru              No Syspassword
  No SIXEL Graphics      Soft Characters        Printer port           Numeric Keypad
  ANSI_CRT                No Regis                No Block_mode         Advanced_video
  Edit_mode                DEC_CRT                DEC_CRT2              No DEC_CRT3
```

In this example, the **SHOW TERMINAL** command displays the characteristics of this specific terminal. If you are displaying statistics about a terminal allocated to another user, the input, output, LFill, CRfill, width, page, and parity statistics are not shown.

SHOW TIME

Displays the current date and time. The **DAY** element is optional.

format

SHOW [DAY]TIME

parameters

None.

example

```
$ SHOW TIME
19-APR-1990 00:03:45
```

The **SHOW TIME** command in this example displays the current date and time.

SHOW TRANSLATION

Displays the first translation found for the specified logical name. You can specify the tables that are searched.

Requires READ (R) access to a logical name table to display information about any logical name cataloged in that table.

format

SHOW TRANSLATION *logical-name*

parameter

logical-name

Specifies the logical name whose translation you want to display.

qualifier

/TABLE=name

Searches the specified table. The default is */TABLE=LNMDCL_ LOGICAL*.

example

```
$ SHOW TRANSLATION/TABLE=LNMSYSTEM USER  
USER = "DBA2:" (LNMSYSTEM_TABLE)
```

The SHOW TRANSLATION command in this example displays the translation for the logical name USER. Because a table name is specified, the SHOW TRANSLATION command does not use the default search order. Only the specified table, LNMSYSTEM, is searched. LNMSYSTEM is the system logical name table.

SHOW USERS

Displays the user name and node name (in a VAXcluster environment) of interactive, subprocess, and batch users on the system.

format

SHOW USERS [*username*]

parameter

username

Specifies the user about whom you want information. Wildcard characters are allowed. If you specify a string, all users whose user names begin with the string are displayed. If you omit the username parameter, a list of all interactive, subprocess, and batch users is displayed.

qualifier

/BATCH (default)

Displays all batch users in the VAXcluster environment. To restrict the display to users on specific nodes, use **/BATCH** with the **/NODE** qualifier.

/CLUSTER (default)

Displays all users on all nodes in a VAXcluster environment.

/FULL

Displays the user name, the node name, the process name, the process identification code (PID), terminal names (both virtual and physical), and port information of all interactive, subprocess, and batch users on the system.

/INTERACTIVE (default)

Displays all interactive users in the VAXcluster environment. To restrict the display to users on specific nodes, use **/INTERACTIVE** with the **/NODE** qualifier.

/NETWORK

Displays all network users in the VAXcluster environment. To restrict the display to users on specific nodes, use **/NETWORK** with the **/NODE** qualifier.

/NODE[=(name,...)]

Displays all interactive, subprocess, and batch users on the specified node or nodes. If you enter **/NODE** without a value, the qualifier displays all the interactive, subprocess, and batch users on the local node.

/OUTPUT[=file-spec]

/NOOUTPUT

By default, the output of the **SHOW USERS** command is sent to the current **SYS\$OUTPUT** device (usually your terminal). To send the output to a file, use the **/OUTPUT** qualifier followed by a file specification. If you enter **/NOOUTPUT**, output is suppressed.

/SUBPROCESS

Displays all subprocess users in the VAXcluster environment. To restrict the display to users on specific nodes, use **/SUBPROCESS** with the **/NODE** qualifier.

DCL-398 DCL Commands

SHOW USERS

example

```
$ SHOW USERS *MAR*
      VAX/VMS User Processes at 19-APR-1990 14:06.16.24
Total number of users = 3, number of processes = 10
```

Username	Node	Interactive	Subprocess	Batch
LMARTIN	ATHENS	-	-	1
LMARTIN	RUMAD	5	2	
MARRA	ATHENS	1		
MARSHALL	OCALA	1		

The SHOW USERS command in this example displays the user name and node names of all users whose user names contains the string MAR.

```
$ SHOW USERS YETTO
      VAX/VMS User Processes at 19-APR-1990 08:59:38.76
Total number of users = 1, number of processes = 2
```

Username	Node	Interactive	Subprocess	Batch
YETTO	SPHAWK	2		

The SHOW USERS command in this example displays the user name and node name of the user YETTO.

SHOW WORKING_SET

Displays the working set limit, quota, and extent assigned to the current process.

format

SHOW WORKING_SET

qualifier

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS\$OUTPUT.

example

```
$ SHOW WORKING_SET
Working Set      /Limit= 180   /Quota= 350       /Extent= 1200
Adjustment enabled  Authorized Quota= 350  Authorized Extent= 1200
```

In this example, the response to the SHOW WORKING_SET command indicates that the current process has a working set limit of 180 pages, a quota of 350 pages and that the current quota is equal to the authorized limit (350 pages). It also shows that the current process has a working set extent of 1200 and that the current extent is equal to the authorized limit (1200).

SORT

Invokes the Sort/Merge Utility (SORT) to reorder the records in a file into a defined sequence and to create either a new file of the reordered records or an address file by which the reordered records can be accessed. For a complete description of the Sort/Merge Utility, including more information about the SORT command, see the Reference Section.

format

SORT *input-file-spec[,...] output-file-spec*

SPAWN

Creates a subprocess of the current process.

The RESOURCE_WAIT state is required to spawn a process. Requires TMPMBX or PRMMBX user privilege. The SPAWN command does not manage terminal characteristics. The SPAWN and ATTACH commands cannot be used if your terminal has an associated mailbox.

format

SPAWN [*command-string*]

parameter

command-string

Specifies a command string of less than 132 characters that is to be executed in the context of the created subprocess. When the command completes execution, the subprocess terminates and control returns to the parent process. If both a command string and the /INPUT qualifier are specified, the specified command string executes before additional commands are obtained from the /INPUT qualifier.

qualifiers

/CARRIAGE_CONTROL ***/NOCARRIAGE_CONTROL***

Determines whether carriage return/line feed characters are prefixed to the subprocess's prompt string. By default, SPAWN copies the current setting of the parent process.

/CLI=cli-file-spec ***/NOCLI***

Specifies the name of a command language interpreter (CLI) to be used by the subprocess. The default CLI is the same as the parent process (defined in SYSUAF). If you specify /CLI, the attributes of the parent process are copied to the subprocess.

DCL-400 DCL Commands
SPAWN

/INPUT=file-spec

Specifies an input file containing one or more DCL commands to be executed by the spawned subprocess. File type defaults to COM and no wildcards are allowed in the file specification. Once processing of the input file is complete, the subprocess is terminated. If both a command string and the */INPUT* qualifier are specified, the specified command string executes before additional commands are obtained from the */INPUT* qualifier. If neither is specified, SYS\$INPUT is assumed (in which case a SPAWN/NOWAIT command is aborted if CTRL/Y is pressed to abort something running in your parent process).

/KEYPAD (default)

/NOKEYPAD

Copies keypad key definitions and the current keypad state from the parent process.

/LOG (default)

/NOLOG

Displays the assigned subprocess name and any messages indicating transfer of control between processes.

/LOGICAL_NAMES (default)

/NOLOGICAL_NAMES

Copies process logical names and logical name tables to the subprocess. By default, all process logical names and logical name tables are copied to the subprocess except those explicitly marked CONFINE or created in executive or kernel mode.

/NOTIFY

/NONOTIFY (default)

Controls whether a message is broadcast to your terminal notifying you that your subprocess has completed or aborted. This qualifier should not be used unless you specify the */NOWAIT* qualifier. */NOTIFY* cannot be specified when the SPAWN command is executed from within a noninteractive process.

/OUTPUT=file-spec

Specifies the output file to which the results of the SPAWN operation are written. No wildcards can be used in the file specification. (Do not specify SYS\$COMMAND as a file specification for */OUTPUT* when using the */NOWAIT* qualifier; both parent and subprocess output will be displayed simultaneously on your terminal.)

/PROCESS=subprocess-name

Specifies the name of the subprocess to be created. The default subprocess name format is username_n.

/PROMPT[=string]

Specifies the prompt string for DCL to use in the subprocess. The default is the prompt of the parent process. The string must be enclosed in quotation marks if it contains spaces, special characters, or lowercase characters.

/SYMBOLS (default)
/NOSYMBOLS

Determines whether global and local symbols (except \$RESTART, \$SEVERITY, and \$STATUS) are passed to the subprocess.

/TABLE=command-table

Specifies the name of an alternate command table to be used by the subprocess.

/WAIT (default)
/NOWAIT

Requires that you wait for the subprocess to terminate before you enter another DCL command. The /NOWAIT qualifier allows you to enter new commands while the subprocess is running. (Use the /OUTPUT qualifier with the /NOWAIT qualifier to avoid displaying both parent and subprocess output on the terminal simultaneously.)

example

```
$ RUN MYPROG
.
.
.
$ CTRL/Y
$ SPAWN MAIL
%DCL-S-SPAWNED, process SMITH_1 spawned
%DCL-S-ATTACHED, terminal now attached to process SMITH_1
MAIL> READ
.
.
.
MAIL> EXIT
%DCL-S-RETURNED, control returned to process SMITH
$ CONTINUE
```

The SPAWN command in this example allows you to enter the VMS Mail Utility without terminating the currently running program. After you exit from MAIL, control is returned to the parent process.

START/CPU

Starts the specified secondary processor or processors in a VMS multiprocessing system. The /CPU qualifier is required.

Applies only to VMS multiprocessing systems. Requires change mode to kernel (CMKRNL) privilege.

format

START/CPU [*cpu-id,...*]

parameter

[*cpu-id,...*]

Decimal value representing the identity of a processor in a VMS multiprocessing system. In a VAX 8300 system, for instance, the CPU ID is the VAXBI node number of the processor; in a VAX 8800, the CPU ID of the left processor is 1 and that of the right processor is 0. If you do not specify a CPU ID and do not include the /ALL qualifier, the START/CPU command selects a single available processor to join the multiprocessing system.

description

The START/CPU command starts a secondary processor in a VMS multiprocessing system.

You can issue a START/CPU command only for processors in the STOPPED or TIMEOUT state, as represented by the SHOW CPU command. Otherwise, the START/CPU command has no effect.

qualifier

/ALL

Selects all remaining processors in the system's available set to join the multiprocessing system.

START/QUEUE

Starts or restarts the specified queue after it has been initialized. You also can use this command to change the attributes of the specified queue. The /QUEUE qualifier is required.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

START/QUEUE *queue-name[:]*

parameter

queue-name[:]

Specifies the name of the queue to be started or restarted.

qualifiers

/ALIGN[=(option[,...])]

Prints alignment pages to aid in aligning printer forms. Use this qualifier only when restarting an output execution queue from a paused state.

After the alignment is complete, the queue enters a paused state until you restart it by reentering the **START/QUEUE** command. Printing resumes from the point where alignment data started; that is, the task is backspaced over the pages printed for alignment.

Possible options are as follows:

MASK Specifies that input data are masked by replacing alphabetic characters with x's and numbers with 9's; nonalphanumeric characters are not masked. Mask characters allow you to prevent the printing of sensitive information. If you omit the **MASK** option, data are printed unaltered.

n Specifies the number of alignment pages to print. The value of *n* can be from 1 through 20. By default, one page of alignment data is printed.

/BACKWARD=*n*

Restarts a print queue *n* pages before the current page; *n* defaults to 1. If you omit the page value, printing resumes at the top of the current page. Use this qualifier only when restarting an output execution queue from a paused state.

/BASE_PRIORITY=*n*

Specifies the base process priority at which jobs are initiated from a batch execution queue. By default, if you omit the qualifier, jobs are initiated at the same priority as the base priority established by **DEFPRI** at system generation (usually 4). The base priority specifier can be any decimal value from 0 through 15.

/BATCH

/NOBATCH

Specifies that you are starting or restarting a batch queue. The **/NOBATCH** qualifier cancels the effect of a previous **/BATCH** qualifier on the same command. It is supported in this release for compatibility with VMS Version 4.n. The **/[NO]BATCH** qualifier of the **INITIALIZE/QUEUE** command has superseded the **/[NO]BATCH** qualifier of the **START/QUEUE** command. Digital recommends that you use the **INITIALIZE/QUEUE/[NO]BATCH** command to determine queue type.

DCL-404 DCL Commands
START/QUEUE

Digital also recommends that you update command procedures that use START/QUEUE/[NO]BATCH.

/BLOCK_LIMIT=(*[lowlim,]uplim*)

/NOBLOCK_LIMIT

Limits the size of print jobs that can be processed on an output execution queue. This qualifier allows you to reserve certain printers for certain size jobs. You must specify at least one of the parameters. The *lowlim* parameter is a decimal number referring to the minimum number of blocks that are accepted by the queue for a print job. If a print job is submitted that contains fewer blocks than the *lowlim* value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is decreased sufficiently, the job is processed.

The *uplim* parameter is a decimal number referring to the maximum number of blocks that are accepted by the queue for a print job. If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is increased sufficiently, the job is processed.

/CHARACTERISTICS=(*characteristic*[,...])

/NOCHARACTERISTICS

Specifies one or more characteristics for processing jobs on an execution queue. If a queue does not have all the characteristics that have been specified for a job, the job remains pending. If you specify only one characteristic, you can omit the parentheses. Each time you specify */CHARACTERISTICS*, all previously set characteristics are canceled. Only the characteristics specified with the qualifier are established for the queue. Queue characteristics are installation-specific. The characteristic parameter can be either a value from 0 through 127 or a characteristic name that has been defined by the *DEFINE/CHARACTERISTIC* command.

/CLOSE

Prevents jobs from being entered in the queue through *PRINT* or *SUBMIT* commands or as a result of requeue operations. To allow jobs to be entered, use the */OPEN* qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled). When a queue is marked closed, jobs executing continue to execute. Jobs already pending in the queue continue to be candidates for execution.

/CPUDEFAULT=time

Defines the default CPU time limit for jobs in this batch execution queue. You can specify time as delta time, 0, *INFINITE*, or *NONE*. You can specify up to 497 days of delta time.

If the queue does not have a specified CPUMAXIMUM time limit and the value established in the user authorization file (UAF) has a specified CPU time limit of NONE, either the value 0 or the keyword INFINITE allows unlimited CPU time. If you specify NONE, the CPU time value defaults to the value specified either in the UAF or by the SUBMIT command (if included). CPU time values must be greater than or equal to the number specified by the SYSGEN parameter PQL_MCPULM.

/CPUMAXIMUM=time

Defines the default CPU time limit for all jobs in this batch execution queue. You can specify time as delta time, 0, INFINITE, or NONE. You can specify up to 497 days of delta time.

If the queue does not have a specified CPUMAXIMUM time limit and the value established in the user authorization file (UAF) has a specified CPU time limit of NONE, either the value 0 or the keyword INFINITE allows unlimited CPU time. If you specify NONE, the CPU time value defaults to the value specified either in the UAF or by the SUBMIT command (if included). CPU time values must be greater than or equal to the number specified by the SYSGEN parameter PQL_MCPULM. The time cannot exceed the CPU time limit set by the /CPUMAXIMUM qualifier.

/DEFAULT=(option[,...])
/NODEFAULT

Establishes defaults for certain options of the PRINT command. Defaults are specified by the list of options. If you specify only one option, you can omit the parentheses. After you set an option for the queue with the /DEFAULT qualifier, you do not have to specify that option in your PRINT commands. If you do specify these options in your PRINT command, the values specified with the PRINT command override the values established for the queue with the /DEFAULT qualifier. For information on establishing mandatory queue attributes, see the description of the /SEPARATE qualifier.

You cannot use the /DEFAULT qualifier with the /GENERIC qualifier.

Possible options are as follows:

[NO]BURST[=keyword]

Controls whether two file flag pages with a burst bar between them are printed preceding output. If you specify the value ALL (default), these flag pages are printed before each file in the job. If you specify the value ONE, these flag pages are printed once before the first file in the job.

[NO]FEED

Specifies whether a form-feed is inserted automatically at the end of a page.

DCL-406 DCL Commands
START/QUEUE

[NO]FLAG[=keyword]	Controls whether a file flag page is printed preceding output. If you specify the value ALL (default), a flag page is printed before each file in the job. If you specify the value ONE, a flag page is printed once before the first file in the job.
FORM=type	Specifies the default form for an output execution queue. If a job is submitted without an explicit form definition, this form is used to process the job. See also /FORM_MOUNTED.
[NO]TRAILER[=keyword]	Controls whether a file trailer page is printed following output. If you specify the value ALL (default), a trailer page is printed after each file in the job. If you specify the value ONE, a trailer page is printed once after the last file in the job.

When you specify the BURST option for a file, the [NO]FLAG option does not add or subtract a flag page from the two flag pages that are printed preceding the file.

/DESCRIPTION=string
/NODESCRIPTION

A string of up to 255 characters used to provide operator-supplied information about the queue.

Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (").

The /NODESCRIPTION qualifier removes any descriptive text that may be associated with the queue.

/DISABLE_SWAPPING
/NODISABLE_SWAPPING

Controls whether batch jobs executed from a queue can be swapped in and out of memory.

/ENABLE_GENERIC
/NOENABLE_GENERIC

Specifies whether files queued to a generic queue that does not specify explicit queue names with the /GENERIC qualifier can be placed in this execution queue for processing. For more information, see the description of the /GENERIC qualifier.

/FORM_MOUNTED=type

Specifies the mounted form for an output execution queue. If the stock of the mounted form does not match the stock of the default form, as indicated by the qualifier /DEFAULT=FORM, all jobs submitted to this queue without an explicit form definition enter a pending state. If a job is submitted with an explicit form and the stock of the explicit form does not match the stock of the mounted form, the job enters a pending state. In both cases, the jobs remain pending state until the stock of the mounted

form of the queue matches the stock of the form associated with the job. To specify the form type, use either a numeric value or a form name that has been defined by the DEFINE/FORM command. Form types are installation-specific. You cannot use the /FORM_MOUNTED qualifier with the /GENERIC qualifier.

/FORWARD=n

Advances the specified number of pages before resuming printing the current file in the current job; the default is 1. If you omit the page value, printing resumes at the top of the next page. Use this qualifier only when restarting an output execution queue from a paused state.

/GENERIC[(queue-name[,...])]
/NOGENERIC

Specifies a generic queue. Also specifies that jobs placed in this queue can be moved for processing to compatible execution queues. The /GENERIC qualifier optionally accepts a list of target execution queues that have been previously defined. For a generic batch queue, these target queues must be batch execution queues. For a generic output queue, these target queues must be output execution queues, but can be of any type (printer, server, or terminal). If you do not specify any target execution queues with the /GENERIC qualifier, jobs can be moved to any execution queue that (1) is initialized with the /ENABLE_GENERIC qualifier, and (2) is the same type (batch or output) as the generic queue. To define the queue as a generic batch or output queue, you use the /GENERIC qualifier with either the /BATCH or /DEVICE qualifier. If you specify neither /BATCH nor /DEVICE on creation of a generic queue, by default the queue becomes a generic printer queue.

/JOB_LIMIT=n

Specifies the number of batch jobs that can be executed concurrently from the queue. Specify a number in the range 0 through 255.

/LIBRARY=file-name
/NOLIBRARY

Specifies the file name for the device control library. When you initialize an output execution queue, you can use the /LIBRARY qualifier to specify an alternate device control library. You can use only a file name as the parameter of the /LIBRARY qualifier. The system always assumes that the file is located in SYS\$LIBRARY and that the file type is TLB.

/NEXT

Aborts the currently suspended print job and begins processing of the first pending job in the queue.

DCL-408 DCL Commands
START/QUEUE

/ON=[node::]device[:] (*printer, terminal, server queue*)

/ON=node:: (*batch queue*)

Specifies the node or device, or both, on which this execution queue is located. For batch execution queues, you can specify only the node name. For output execution queues, you can include both the node name and the device name.

/OPEN

Allows jobs to be entered in the queue through PRINT or SUBMIT commands or as the result of requeue operations. To prevent jobs from being entered in the queue, use the /CLOSE qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled).

/OWNER_UIC=uic

Requires OPER privilege or CONTROL and EXECUTE access to the queue. Enables you to change the user identification code (UIC) of the queue.

/PROCESSOR=filename

/NOPROCESSOR

Allows you to specify your own print symbiont for an output execution queue. You can use any valid file name as a parameter of the /PROCESSOR qualifier. The system supplies the device and directory name SYS\$SYSTEM and the file type EXE. If you use this qualifier for an output queue, it specifies that the symbiont image to be executed is SYS\$SYSTEM:filename.EXE.

/PROTECTION=(ownership[:access],...)

Requires OPER privilege or CONTROL and EXECUTE access to the queue. Specifies the protection of the queue. Ownership categories are SYSTEM, OWNER, GROUP, WORLD; each category can be abbreviated to its first character. Access categories are R (READ), W (WRITE), E (EXECUTE), and D (DELETE); a null access specification means no access.

/RECORD_BLOCKING

/NORECORD_BLOCKING

Determines whether the symbiont can concatenate (or block together) output records for transmission to the output device. If you specify /NORECORD_BLOCKING, the symbiont sends each formatted record in a separate I/O request to the output device. For the standard VMS print symbiont, record blocking can have a significant performance advantage over single-record mode.

/RETAIN[=option]
/NORETAIN

Holds jobs in the queue in a retained status after they have executed. The */NORETAIN* qualifier enables you to reset the queue to the default. Possible options are as follows:

ALL Holds all jobs in the queue after execution
ERROR Holds in the queue only jobs that complete unsuccessfully

/SCHEDULE=[NO]SIZE

Specifies whether pending jobs in an output queue are scheduled for printing based on the size of the job. When the */SCHEDULE=SIZE* qualifier is in effect, shorter jobs are printed before longer ones. When */SCHEDULE=NOSIZE* is in effect, jobs are printed in the order they were submitted, regardless of size.

If you enter this command while there are pending jobs in any queue, its effect on future jobs is unpredictable.

/SEARCH="search-string"

Specifies that printing is to resume at the page containing the specified string. The search for the string moves forward, beginning on the page following the current page. During the search, consecutive tabs and spaces are treated as a single space, and character case is ignored. The string can be from 1 through 63 characters and must be enclosed in quotation marks. Use this qualifier only when restarting an output execution queue from a paused state.

/SEPARATE=(option[,...])
/NOSEPARATE

Specifies the mandatory queue attributes, or job separation options, for an output execution queue. Job separation options cannot be overridden by the *PRINT* command.

You cannot use the */SEPARATE* qualifier with the */GENERIC* qualifier.

The job separation options are as follows:

[NO]BURST Specifies whether two job flag pages with a burst bar between them are printed at the beginning of each job.
[NO]FLAG Specifies whether a job flag page is printed at the beginning of each job.

DCL-410 DCL Commands

START/QUEUE

[NO]TRAILER	Specifies whether a job trailer page is printed at the end of each job.
[NO]RESET=(module[,...])	Specifies one or more device control library modules that contain the job reset sequence for the queue. The specified modules from the queue's device control library (by default <code>SYS\$LIBRARY:SYSDEVCTL</code>) are used to reset the device each time a job reset occurs. The RESET sequence occurs after any file trailer and before any job trailer. Thus, all job separation pages are printed when the device is in its RESET state.

When you specify **/SEPARATE=BURST**, the **[NO]FLAG** separation option does not add or subtract a flag page from the two flag pages that are printed preceding the job.

For information on establishing queue attributes that can be overridden, see the description of the **/DEFAULT** qualifier.

/TERMINAL ***/NOTERMINAL***

Indicates that the output queue is a terminal queue. The **/NOTERMINAL** qualifier cancels the effect of a previous **/TERMINAL** qualifier on the same command. The **[NO]DEVICE** qualifier of the **INITIALIZE/QUEUE** command has superseded the **[NO]TERMINAL** qualifier. Digital recommends that you use the **INITIALIZE/QUEUE** command to determine queue type. Digital also recommends that you use this qualifier to update command procedures that use **START/QUEUE/[NO]TERMINAL**.

/TOP_OF_FILE

Resumes printing at the beginning of the file that was current when the output execution queue paused. Use this qualifier only when restarting an output execution queue from a paused state.

/WSDEFAULT=n

Defines for a batch job a working set default, the default number of physical pages that the job can use. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

If you specify **0** or **NONE**, the working set default value defaults to the value specified in the UAF or by the **SUBMIT** command (if included).

/WSEXTENT=n

Defines for the batch job a working set extent, the maximum amount of physical memory that the job can use. The job uses the maximum amount of physical memory only when the system has excess free pages. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

If you specify 0 or NONE, the working set extent value defaults to the value specified in the UAF or by the SUBMIT command (if included).

/WSQUOTA=n

Defines for a batch job a working set quota, the amount of physical memory that is guaranteed to the job. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

Working set default, working set quota, and working set extent values are included in each user record in the system user authorization file (UAF). You can specify working set values for individual jobs and/or for all jobs in a given queue.

example

```
$ STOP/QUEUE LPA0
$ START/QUEUE/TOP_OF_FILE LPA0
```

The STOP/QUEUE command in this example suspends operation of the printer queue LPA0. Then the START/QUEUE/TOP_OF_FILE command resumes operation. The file that was being printed when the queue was stopped is started again from the beginning.

START/QUEUE/MANAGER

Starts the queue manager for the batch/print facility and opens the queue file. After the system is bootstrapped, you must execute this command before you can execute any other queue management or job submission command. The /QUEUE qualifier is optional, but the /MANAGER qualifier is required.

Requires both OPER and SYSNAM privileges.

format

START/QUEUE/MANAGER *[file-spec]*

parameter

[file-spec]

Specifies the name of the queue file. This file contains information about batch and print jobs, queues, form definitions, and characteristics. The file specification parameter is used in VAXcluster systems. The default file specification is SYS\$SYSTEM:JBCSYSQUE.DAT. Any elements that you omit from the file specification default to those of SYS\$SYSTEM:JBCSYSQUE.DAT. No wildcard characters are permitted in the file specification.

DCL-412 DCL Commands

START/QUEUE/MANAGER

qualifiers

/BUFFER_COUNT=*n*

Specifies the number of buffers in a local buffer cache to allocate for performing I/O operations to the queue file. Specify a positive integer in the range of 1 through 1500, or 0. If you specify 0, the default value of 50 is used.

/EXTEND_QUANTITY=*n*

Specifies the number of blocks by which the queue file is extended, when necessary. This value is also used as the initial allocation size when the queue file is created. Specify a positive integer in the range of 10 through 65,535, or 0. If you specify 0, the default value of 100 is used.

/NEW_VERSION

/NON_NEW_VERSION (default)

Specifies that a new version of the queue file be created to supersede an existing version. If you specify a new version all jobs in the previous version are lost. The new queue file contains no information until you enter a subsequent INITIALIZE/QUEUE command.

/RESTART

/NORESTART (default)

The **/RESTART** qualifier specifies that the queue manager be restarted automatically on recovery from a job controller abort. In addition, batch and output queues are restored to the states that existed prior to the interruption of service. The queue file that is opened is the queue file that was open before the abort. When the job controller incurs an internal fatal error, the process aborts and restarts itself. By default, the queue manager is not restarted. Intervention by a user with OPERATOR privilege is necessary to restart the queue manager and to restore the queuing environment using START/QUEUE/MANAGER and appropriate START/QUEUE commands. Note that in order to prevent a looping condition, the job controller does not restart the queue manager if it detects an error within two minutes of starting the queue manager.

example

```
$ START/QUEUE/MANAGER DUA5:[SYSQUE]
```

The START/QUEUE/MANAGER command in this example opens the queue file JBCSYSQUE.DAT on the cluster-accessible disk volume DUA5, in directory SYSQUE. You must mount the disk before you enter the START/QUEUE/MANAGER command.

STOP

Terminates execution of a command, an image, a command procedure, a command procedure that was interrupted by CTRL/Y, or a detached process or subprocess.

Requires GROUP privilege to stop other processes in the same group. Requires WORLD privilege to stop processes outside your group.

format

STOP [*process-name*]

parameter

process-name

Requires that the process be in your group.

Specifies the name of the process to be deleted. The process name can have from 1 to 15 alphanumeric characters. The specified process must have the same group number in its user identification code (UIC) as the current process; you cannot use the process-name parameter to stop a process outside of your group. To stop a process outside of your group, you must use the qualifier */IDENTIFICATION=pid*.

qualifier

/IDENTIFICATION=pid

Specifies the system-assigned process identification code (PID). */IDENTIFICATION* can be used in place of the process name parameter.

example

```
$ RUN/PROCESS_NAME=LIBRA LIBRA
%RUN-S-PROC_ID, identification of created process is 0013340D
.
.
.
$ STOP LIBRA
```

The RUN command in this example creates a subprocess named LIBRA to execute the image LIBRA.EXE. Subsequently, the STOP command causes the image to exit and deletes the process.

STOP/CPU

Stops the specified secondary processor or processors in a VMS multiprocessing system. The /CPU qualifier is required.

Applies only to VMS multiprocessing systems. Requires change mode to kernel (CMKRNL) privilege.

format

STOP/CPU [*cpu-id*,...]

parameter

cpu-id

Decimal value representing the identity of a processor in a VMS multiprocessing system. In a VAX 8300 system, for instance, the CPU ID is the VAXBI node number of the processor; in a VAX 8800, the CPU ID of the left processor is 1 and that of the right processor is 0. If you do not specify a CPU ID, the STOP/CPU command selects a processor in the current active set to stop.

description

The STOP/CPU command removes a secondary processor from the active set in a VMS multiprocessing system. If the secondary processor is not executing a process when the STOP/CPU command is issued, it enters the STOPPED state. If the secondary is executing a process at the time, it continues to execute the current process until it becomes a candidate for rescheduling on another processor in the system. When this occurs, the secondary enters the STOPPED state.

The VMS operating system subjects a processor to a set of checks when it is the object of a STOP/CPU command. As a result, you may not be permitted to stop certain processors that are vital to the functioning of the system. In these cases, there is usually a process in the system that can execute only on the processor you intend to stop. You can determine this by issuing a SHOW CPU/FULL command. In unusual circumstances, you can bypass the checking mechanism by using the /OVERRIDE_ CHECKS qualifier in the command.

The STOP/CPU command has no effect if its object processor is already in the STOPPED state when it is issued.

qualifiers

/ALL

Stops all eligible secondary processors in the system's active set.

/OVERRIDE_CHECKS

Directs the STOP/CPU command to bypass a series of checks that determine whether the specified processor is eligible for removal from the active set.

STOP/QUEUE

The STOP/QUEUE command causes the specified execution queue to pause. All jobs currently executing in the queue are suspended (until the queue is restarted with the START/QUEUE command), and no new jobs can be initiated. The /QUEUE qualifier is required.

Requires OPER privilege or EXECUTE (E) access to the queue.

format

STOP/QUEUE *queue-name[:]*

parameter

queue-name[:]

Specifies the name of the queue that you want to pause.

example

```
$ STOP/QUEUE TEXTBATCH
```

```
.  
. .  
. . .
```

```
$ START/QUEUE/BLOCK_LIMIT=500 TEXTBATCH
```

The STOP/QUEUE command in this example halts all batch jobs that are currently executing on the queue TEXTBATCH and places that queue in the paused state. Later the START/QUEUE command releases the queue from the paused state. All the jobs that were halted resume processing, but the START/QUEUE command now limits any further jobs to 500 blocks or smaller.

STOP/QUEUE/ABORT

Aborts a job that is printing or processing on an output queue, deletes it from the queue, and begins processing the first pending job in the queue. The /QUEUE qualifier is optional, but the /ABORT qualifier is required.

Requires OPER privilege, EXECUTE (E) access to the queue, or DELETE (D) access to the current job.

format

STOP/QUEUE/ABORT *queue-name[:]*

parameter

queue-name[:]

Specifies the name of the queue containing the job you want to abort.

example

```
$ STOP/QUEUE/ABORT LPA0
```

This example aborts the current print job on the queue LPA0. The print symbiont begins to process the first pending job in the queue. Assuming there is no problem with the printer, the current page of the file completes printing. If the printer queue has been set up to put trailer pages at the end of jobs, a trailer page is printed after the current page is completed.

STOP/QUEUE/ENTRY

Aborts one or more jobs that are executing on a batch queue or printing on an output queue, deletes them from the queue, and begins processing the first pending job in the queue. The /QUEUE qualifier is optional, but the /ENTRY qualifier is required.

Requires OPER privilege, EXECUTE (E) access to the queue, or DELETE (D) access to the current job.

format

STOP/QUEUE/ENTRY=(*entry-number[,...]*) [*queue-name[:]*]

parameters

entry-number[,...]

Specifies the entry number (or a list of entry numbers) of jobs to be deleted.

[queue-name[:]]

Specifies the name of the queue that contains the jobs that you want to abort.

example

```
$ STOP/QUEUE/ENTRY=365 SYS$BATCH
```

The STOP/QUEUE/ENTRY command in this example aborts batch job number 365 currently executing on the SYS\$BATCH queue and begins the first pending job in the queue.

STOP/QUEUE/MANAGER

Performs an orderly shutdown of the system job queue manager on the node from which the command is entered. The /QUEUE qualifier is optional, but the /MANAGER qualifier is required.

Requires both OPER and SYSNAM privileges.

format

```
STOP/QUEUE/MANAGER
```

parameters

None.

example

```
$ STOP/QUEUE/MANAGER
```

The STOP/QUEUE/MANAGER command in this example performs a shutdown of all queues on the node from which the command is entered.

STOP/QUEUE/NEXT

Stops the specified queue after all executing jobs have completed processing. No new jobs can be initiated; the START/QUEUE command restarts the queue. The /QUEUE qualifier is optional, but you must specify the /NEXT qualifier.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

```
STOP/QUEUE/NEXT queue-name[:]
```

parameter

queue-name[:]

Specifies the name of the queue that you want to stop.

example

```
$ STOP/QUEUE/NEXT LPA0  
$ SHOW QUEUE/ALL LPA0  
Printer queue LPA0  
$ DELETE/QUEUE LPA0
```

This example shows how to delete the printer queue LPA0. First, the STOP/QUEUE/NEXT command is entered, which stops the printer after the current job is printed. Then the SHOW QUEUE/ALL command is entered to ensure that no jobs are pending in the queue. The screen display shows that no jobs are pending. Finally, the DELETE/QUEUE command is entered to delete the printer queue LPA0.

STOP/QUEUE/REQUEUE

Stops the current jobs on the specified queue and requeues them for later processing. The queue does not stop; processing of the first pending job in the queue begins. The /QUEUE qualifier is optional, but the /REQUEUE qualifier is required. The /ENTRY qualifier is required to requeue batch jobs.

Requires OPER privilege, EXECUTE access to the queue or DELETE access to the current job.

format

```
STOP/QUEUE/REQUEUE[=queue-name] queue-name[:]  
STOP/QUEUE/REQUEUE/ENTRY=(entry-number[,...])  
[=queue-name] queue-name[:]
```

parameters

***queue-name*[:]**

Specifies the name of the queue that contains the jobs that you want to stop. When you specify a queue name as a parameter for the /REQUEUE qualifier, the jobs are requeued to that queue. Otherwise, the jobs are requeued in the current queue.

***entry-number*[,...]**

Specifies the entry number (or a list of entry numbers) of the jobs you want to requeue. If you specify only one entry number, you can omit the parentheses.

The system assigns a unique entry number to each queued print or batch job in the system. By default, the PRINT and SUBMIT commands display the entry number when they successfully queue a job for processing. These commands also create or update the local symbol \$ENTRY to reflect the entry number of the most recently queued job. To find a job's entry number, enter the SHOW ENTRY or SHOW QUEUE command.

qualifiers

/ENTRY=(entry-number[,...])

Specifies the entry number of one or more jobs you want to abort. If you specify only one entry number, you can omit the parentheses.

The system assigns a unique entry number to each queued print or batch job in the system. By default, the PRINT and SUBMIT commands display the entry number when they successfully queue a job for processing. These commands also create or update the local symbol \$ENTRY to reflect the entry number of the most recently queued job. To find a job's entry number, enter the SHOW ENTRY or SHOW QUEUE command.

You must use the /ENTRY qualifier when you enter the STOP/QUEUE/REQUEUE command for a batch queue. Entry numbers specified must match entry numbers of executing jobs.

/HOLD

Places the aborted job or jobs in a hold state for later release with the SET ENTRY/RELEASE or SET ENTRY/NOHOLD command.

/PRIORITY=n

Requires OPER or ALTPRI privilege to raise the priority value above the value of the SYSGEN parameter MAXQUEPRI. Changes the priority of the requeued job or jobs. The n parameter can be from 0 to 255; the default value of the n parameter is the same as the priority value that the job or jobs had when it was stopped.

example

```
$ STOP/QUEUE/REQUEUE=LPB0 LPA0
```

In this example, the current print job on queue LPA0 is stopped and requeued to queue LPB0. If the print symbiont sent checkpoint information about the print job to the job controller, printing resumes on LPB0 at the last checkpoint recorded.

STOP/QUEUE/RESET

Abruptly stops the queue and returns control to the system. Any jobs currently executing are stopped immediately. The START/QUEUE command restarts the queue. Current jobs that can be restarted (all print jobs and any batch jobs submitted with the /RESTART qualifier) are requeued for processing. Current jobs that cannot be restarted are aborted and must be resubmitted for processing. The /QUEUE qualifier is optional, but you must specify the /RESET qualifier.

Requires OPER privilege or EXECUTE (E) access to the specified queue.

format

STOP/QUEUE/RESET *queue-name[:]*

parameter

queue-name[:]

Specifies the name of the queue you want to reset.

example

```
$ STOP/QUEUE/RESET TEXBATCH
```

The STOP/QUEUE/RESET command in this example stops the TEXBATCH queue. Any current job that was submitted with the /RESTART qualifier is requeued for processing when the queue is restarted. Current jobs that did not specify /RESTART must be resubmitted to the queue.

SUBMIT

Queues one or more files containing command procedures to a batch queue.

Requires OPER privilege, EXECUTE (E) access to the queue, or WRITE (W) access to the queue.

format

SUBMIT *file-spec[,...]*

parameter

file-spec[,...]

Specifies one or more files containing command procedures. Wildcard characters are allowed in the directory specification, file name, file type, and version number fields. The default file type is that of the preceding file. If no previous file specification contains an explicit file type, the default file type is COM. If you specify a node name, you must use the /REMOTE qualifier.

qualifiers

/AFTER=time

/NOAFTER

Requests that the job be held until after a specific time. If the specified time has passed already, the job is processed immediately. You can specify time as either an absolute time or as a combination of absolute and delta times.

/BACKUP
/NOBACKUP

Modifies the interpretation of the time value specified with the */BEFORE* or */SINCE* qualifier. */BACKUP* selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: */CREATED*, */EXPIRED*, and */MODIFIED*. If you specify none of these four time qualifiers, the default is */CREATED*.

/BEFORE[=time]
/NOBEFORE

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: *TODAY* (default), *TOMORROW*, or *YESTERDAY*. Specify one of the following qualifiers with */BEFORE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

/BY_OWNER[=uic]
/NOBY_OWNER

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CHARACTERISTICS=(characteristic[,...])

Specifies the name or numbers of one or more characteristics to be associated with the the job. Characteristics can refer to such things as color of ink. If you specify only one characteristic, you can omit the parentheses.

/CLI=filename

Specifies the command language interpreter (CLI) to be used to process the job. The file specification assumes the device name *SYS\$SYSTEM:* and the file type *EXE* (*SYS\$SYSTEM:filename.EXE*).

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each submit operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL

RET

You can use any combination of uppercase and lowercase letters for word responses. You can abbreviate word responses to one or more letters (for example, *T*, *TR*, or *TRU* for *TRUE*), but these abbreviations must be

DCL-422 DCL Commands

SUBMIT

unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CPUTIME=time

Defines a CPU time limit for the batch job. You can specify time as delta time, 0, INFINITE, or NONE.

/CREATED (default) ***/NOCREATED***

Modifies the interpretation of the time value specified with the /BEFORE or /SINCE qualifier. The qualifier /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/DELETE ***/NODELETE (default)***

Positional qualifier. Controls whether files are deleted after processing. If you specify the /DELETE qualifier after the SUBMIT command name, all files in the job are deleted after processing. If you specify the /DELETE qualifier after a file specification, only that file is deleted after it is processed.

/EXCLUDE=(file-spec[,...]) ***/NOEXCLUDE***

Excludes the specified files from the submit operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED ***/NOEXPIRED***

Modifies the interpretation of the time value specified with the /BEFORE or /SINCE qualifier. The /EXPIRED qualifier selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/HOLD***/NOHOLD (default)***

Controls whether the job is made available for immediate processing. The */HOLD* qualifier holds the job until it is released by the *SET ENTRY/RELEASE* or *SET ENTRY/NOHOLD* command.

/IDENTIFY (default)***/NOIDENTIFY***

Displays the job name, the queue name, the entry number, and the status of the job when it is queued.

/KEEP***/NOKEEP***

Controls whether the log file is deleted after it is printed; */NOKEEP* is the default unless */NOPRINTER* is specified.

/LOG_FILE[=file-spec]***/NOLOG_FILE***

Names the log file. No wildcards are allowed in the file specification. You can use the */LOG_FILE* qualifier to write the log file to a different device. Logical names in the file specification are translated in the context of the process that submits the job.

/MODIFIED***/NOMODIFIED***

Modifies the interpretation of the time value specified with the */BEFORE* or */SINCE* qualifier. */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/NAME=job-name

Names the job (and possibly the batch job log file). The job name must be 1 to 39 alphanumeric characters. If characters other than alphanumeric characters, underscores, or dollar signs are used in the name, enclose the name in quotation marks. The default job name is the name of the first file in the job.

/NOTIFY***/NONOTIFY (default)***

Controls whether a message is broadcast to your terminal session when the job is completed or aborted.

/PARAMETERS=(parameter[,...])

Provides the values of up to eight optional parameters (equated to the symbols P1 through P8, respectively, for each command procedure in the job). The symbols are local to the specified command procedure. If the parameter contains spaces, special characters, or lowercase characters,

SUBMIT

enclose it in quotation marks. The size of the parameter can be from 1 to 255 characters.

/PRINTER[=queue-name](default)

/NOPRINTER

Queues the job log file for printing when your job is completed. */PRINTER* allows you to specify a particular print queue; the default print queue is `SYS$PRINT`. If you specify */NOPRINTER*, */KEEP* is assumed.

/PRIORITY=n

Requires **OPER** or **ALTPRI** privilege to specify a priority greater than the value of the **SYSGEN** parameter **MAXQUEPRI**. Specifies the job-scheduling priority for the batch job with respect to other jobs in the same queue. The value of *n* is an integer in the range of 0 through 255. The default value is the value of the **SYSGEN** parameter **DEFQUEPRI**.

/QUEUE=queue-name[:]

Identifies the batch queue on which the job is entered. The default queue is `SYS$BATCH`.

/REMOTE

Queues the job to `SYS$BATCH` on the remote node specified. When you use */REMOTE*, you *must* include the node name in the file specification. You can specify only the following qualifiers with */REMOTE*: */BACKUP*, */BEFORE*, */BY_OWNER*, */CONFIRM*, */CREATED*, */EXCLUDE*, */EXPIRED*, */MODIFIED*, and */SINCE*.

/RESTART

/NORESTART (default)

Indicates whether the job restarts after a system failure or after a `STOP/QUEUE/REQUEUE` command.

/SINCE[=time]

/NOSINCE

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: **TODAY** (default), **TOMORROW**, or **YESTERDAY**. Specify one of the following qualifiers with */SINCE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

/USER=username

Requires **CMKRNL** privilege and **R** (read) and **W** (write) access to the user authorization file (**UAF**). Allows you to submit a job on behalf of another user. The job runs exactly as if that user had submitted it.

/WSDEFAULT=n

Defines for a batch job a working set default, the default number of physical pages that the job can use. If the queue on which the job executes has a nonzero default working set, the smaller of the specified job and queue values is used. If the queue on which the job executes has a working set default of 0, the smaller of the specified job and UAF values (value established in the user authorization file) is used. If you specify 0 or NONE, the specified queue or UAF value is used. Working set default values must range between the numbers specified by the SYSGEN parameters PQL_MWSDEFAULT and WSMAX.

/WSEXTENT=n

Defines for the batch job a working set extent, the maximum amount of physical memory that the job can use. The job uses the maximum amount of physical memory only when the system has excess free pages. If the queue on which the job executes has a nonzero working set extent, the smaller of the specified job and queue values is used. If the queue on which the job executes has a working set extent of 0, the smaller of the specified job and the value established in the user authorization file (UAF) is used. If you specify 0 or NONE, the specified queue or UAF value is used. Working set extent values must range between the numbers specified by the SYSGEN parameters PQL_MWSEXTENT and WSMAX.

/WSQUOTA=n

Defines for the batch job a working set quota, the amount of physical memory that the job is guaranteed. If the queue on which the job executes has a nonzero working set quota, the smaller of the specified job and queue values is used. If the queue on which the job executes has a working set quota of 0, the smaller of the specified job or the value established in the user authorization file (UAF) is used. If you specify 0 or NONE, the specified queue or UAF value is used. Working set quota values must range between the numbers specified by the SYSGEN parameters PQL_MWSQUOTA and WSMAX.

example

```
$ DEFINE JUNE WORKZ:[JONES]ANNUAL_REPORT.COM  
$ SUBMIT JUNE
```

Job ANNUAL_REPORT (queue SYS\$BATCH, entry 229) started on ZOO_BATCH

In this example, the logical name JUNE is created and equated to ANNUAL_REPORT.COM with the DEFINE command. Using the logical name JUNE, the user submits ANNUAL_REPORT.COM to the batch queue. Note that the system translates the logical name JUNE to ANNUAL_REPORT.COM before ANNUAL_REPORT.COM is submitted to the batch queue. Also, the log file produced is named ANNUAL_REPORT.COM rather than JUNE.COM.

Note also that the job is submitted to the generic queue SYS\$BATCH, but runs on the execution queue ZOO_BATCH.

SUBROUTINE

Defines the beginning of a subroutine in a command procedure. The SUBROUTINE command must be the first executable statement in a subroutine. For more information about the SUBROUTINE command, refer to the description of the CALL command.

SYNCHRONIZE

Holds the process issuing the command until the specified job completes execution.

format

SYNCHRONIZE *[job-name]*

parameter

[job-name]

Specifies the name of the job as defined when the job was submitted. To specify a job that does not have a unique name, use the /ENTRY qualifier to specify the entry number. If you specify both the job name and the /ENTRY qualifier, the job name is ignored.

qualifiers

/ENTRY=entry-number

Identifies the job by the system-assigned entry number.

If you specify both the job-name parameter and the /ENTRY qualifier, the job name is ignored.

/QUEUE=queue-name[:]

Names the queue containing the job. If you use the /QUEUE qualifier, you must specify either the job-name parameter or the /ENTRY qualifier. If you specify the job-name parameter, the default queue is SYS\$BATCH. If you specify the /ENTRY qualifier, there is no default queue.

example

```
$ SUBMIT/NAME=TIMER          COMP.COM
Job TIMER (queue SYS$BATCH, entry 214) started on queue SYS$BATCH
$ SYNCHRONIZE /ENTRY=214
```

In this example, a batch job named **TIMER** is submitted. Then the **SYNCHRONIZE** command is entered interactively. This command places the interactive process in a wait state until entry number 214 (**TIMER**) completes. You cannot enter subsequent commands from your terminal session until the **SYNCHRONIZE** command completes and your process is released from the wait state.

TYPE

Displays the contents of a file or group of files on the current output device.

format

TYPE *file-spec[,...]*

parameter***file-spec[,...]***

Specifies one or more files to be displayed. If you specify a file name and not a file type, the file type defaults to **LIS**. The **TYPE** command displays all files that satisfy the file description.

Wildcard characters are allowed in place of the directory name, file name, file type, or file version number field. Either commas or plus signs can be used to separate two or more files. The files are displayed in the order listed.

qualifiers***/BACKUP***

Modifies the time value specified with the **/BEFORE** or **/SINCE** qualifier. **/BACKUP** selects files according to the dates of their most recent backups. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: **/CREATED**, **/EXPIRED**, and **/MODIFIED**. If you specify none of these four time qualifiers, the default is **/CREATED**.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: **TODAY** (default), **TOMORROW**, or **YESTERDAY**. Specify one of the following qualifiers with **/BEFORE** to indicate the time attribute to be used as the basis for selection: **/BACKUP**, **/CREATED** (default), **/EXPIRED**, or **/MODIFIED**.

/BY_OWNER[=*uic*]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

/CONFIRM

/NOCONFIRM (default)

Controls whether a request is issued before each TYPE operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	CTRL/Z
1	0	ALL

<RET>

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers are NO, FALSE, 0, and <RET>. QUIT or CTRL/Z indicates that you want to stop processing the command at that point. When you respond with ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

/CREATED (default)

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /CREATED selects files based on their dates of creation. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /EXPIRED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/EXCLUDE=(*file-spec*[,...])

Excludes the specified files from the TYPE operation. You can include a directory but not a device in the file specification. Wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you provide only one file specification, you can omit the parentheses.

/EXPIRED

Modifies the time value specified with the /BEFORE or /SINCE qualifier. /EXPIRED selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: /BACKUP, /CREATED, and /MODIFIED. If you specify none of these four time qualifiers, the default is /CREATED.

/MODIFIED

Modifies the time value specified with the */BEFORE* or */SINCE* qualifier. */MODIFIED* selects files according to the dates on which they were last modified. This qualifier is incompatible with the other qualifiers that also allow you to select files according to time attributes: */BACKUP*, */CREATED*, and */EXPIRED*. If you specify none of these four time modifiers, the default is */CREATED*.

/OUTPUT[=file-spec]
/NOOUTPUT

Controls where the output of the command is sent. If you specify */OUTPUT=file-spec*, the output is sent to the specified file, rather than to the current output device, *SYS\$OUTPUT*.

/PAGE
/NOPAGE (default)

Controls whether output from the *TYPE* command is displayed one screen at a time.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as an absolute time, a combination of absolute and delta times, or as one of the following keywords: *TODAY* (default), *TOMORROW*, or *YESTERDAY*. Specify one of the following qualifiers with */BEFORE* to indicate the time attribute to be used as the basis for selection: */BACKUP*, */CREATED* (default), */EXPIRED*, or */MODIFIED*.

example

```
$ TYPE LETTER*.MEM  
April 19, 1990
```

.
.
.

CTRL/Y

Interrupt

```
$ SHOW TIME  
19-APR-1990 15:48:07
```

```
$ CONTINUE
```

Sincerely yours,

.
.
.

In this example, the TYPE command displays all files whose names begin with the word LETTER and have the file type MEM. While the files are being displayed, the user presses CTRL/Y to interrupt the TYPE operation and display the time. After entering the SHOW TIME command, the user enters the CONTINUE command to resume the TYPE operation.

UNLOCK

Makes an improperly closed file accessible.

format

UNLOCK *file-spec[,...]*

parameter

file-spec[,...]

Specifies the name of the file to be unlocked. Wildcard characters are allowed. If you include two or more file specifications, separate them with either commas or plus signs.

qualifiers

/CONFIRM

/NOCONFIRM (*default*)

For each file being unlocked, displays a query to which you must respond Y (YES) or T (TRUE) to unlock the file. Any other response aborts the unlock operation.

/LOG

/NOLOG (*default*)

Controls whether the UNLOCK command displays the file specification of each file being unlocked.

example

```
$ TYPE TST.OUT
%TYPE-E-OPENIN, error opening DISK1:[STEVE]TST.OUT;3 as input
-SYSTEM-W-FILELOCKED, file is deaccess locked
$ UNLOCK TST.OUT
$ TYPE TST.OUT
```

In this example, the request to type the output file TST.OUT returns an error message indicating that the file is locked. The UNLOCK command unlocks it. Then the TYPE command is reentered to display the contents of the file.

VIEW

Invokes the DDIF Viewer, which lets you view a compound document file on a character cell terminal or DECwindows display. For a complete description of the DDIF Viewer, including more information about the **VIEW** command and its qualifiers, see the *VMS Compound Document Architecture Manual*.

format

VIEW *input-file-spec*

WAIT

Puts your process into a wait state for the specified amount of time. The **WAIT** command is used in a command procedure to delay processing of either the procedure itself or a set of commands in the procedure.

format

WAIT *delta-time*

parameter***delta-time***

Specifies a delta time interval in the following format. (A delta time is an offset from the current time to a time in the future.)

hour:minute:second.hundredth

The fields on the format line indicate the following:

hour	An integer in the range 0 through 59
minute	An integer in the range 0 through 59
second	An integer in the range 0 through 59
hundredth	An integer in the range 0 through 99.

The colons and period are required delimiters; also, the delta time must begin with the number of hours and not a colon. Note that the days field, usually included in the delta time format, must be omitted here.

example

```
$ LOOP:  
$ RUN ALPHA  
$ WAIT 00:10  
$ GOTO LOOP
```

In this example, the command procedure executes the program image ALPHA. After the RUN command executes the program, the WAIT command delays execution of the GOTO command for 10 minutes. Note that 00 is specified for the number of hours, because the time specification cannot begin with a colon. After 10 minutes, the GOTO command executes, and the procedure transfers control to the label LOOP and executes the program ALPHA again. The procedure loops until it is interrupted or terminated.

If the procedure is executed interactively, terminate it by pressing CTRL/C or CTRL/Y and entering the STOP command or another DCL command that runs a new image in the process. If the procedure is executed in a batch job, enter the DELETE/ENTRY command to terminate it.

WRITE

Writes the specified data as one record to an open file specified by a logical name.

All qualifiers must precede all *data-item* expressions.

format

WRITE *logical-name expression[,...]*

parameters

logical-name

Specifies the logical name assigned to the output file. Use the logical name assigned by the OPEN command. In interactive mode, specify the process-permanent files identified by the logical names SYS\$INPUT, SYS\$OUTPUT, SYS\$ERROR, and SYS\$COMMAND.

expression[,...]

Specifies data to be written as a single record to the output file. You can specify data items using character string expressions, which may be symbol names, character strings in quotation marks, literal numeric values, or a lexical function. You can specify a list of expressions separated by commas; the command interpreter concatenates the items into one record and writes the record to the output file. The maximum size of any record that can be written is less than 1024 bytes. If, however, you specify the /SYMBOL qualifier, the maximum record size is 2048 bytes.

qualifiers

/ERROR=label

Transfers control on an I/O error to the location specified by *label* (in a command procedure). If no error routine is specified and an error occurs during the writing of the file, the current ON condition action is taken. */ERROR* overrides any ON condition action specified. If an error occurs and control passes successfully to the target label, the reserved global symbol \$STATUS retains the error code.

/SYMBOL

Causes the expression to be interpreted and its expanded value placed in a 2048-byte (instead of a 1024-byte) buffer before the write operation is performed. If you specify multiple expressions, their values are concatenated and placed in the 2048-byte buffer. Use the */SYMBOL* qualifier to write a very large record. Each expression specified must be a symbol.

/UPDATE

Replaces the last record read with the record specified with the expression parameter. You must be able to read and write to a file to use the */UPDATE* qualifier. Use the **WRITE/UPDATE** command only after a **READ** command. The **WRITE/UPDATE** command modifies the last record you have read.

example

```
$ OPEN/WRITE OUTPUT_FILE TESTFILE.DAT
$ INQUIRE ID "Assign Test-id Number"
$ WRITE/ERROR=WRITE_ERROR OUTPUT_FILE "Test-id is ",ID
$ WRITE/ERROR=WRITE_ERROR OUTPUT_FILE ""
$ !
$ WRITE_LOOP:
    .
    .
    .
$ GOTO WRITE_LOOP
$ END_LOOP:
$ !
$ CLOSE OUTPUT_FILE
$ PRINT TESTFILE.DAT
$ EXIT
$ !
$ WRITE_ERROR:
$ WRITE SYS$OUTPUT "There was a WRITE error."
$ CLOSE OUTPUT_FILE
$ EXIT
```

In this example, the open command opens the file TESTFILE.DAT; the **INQUIRE** command requests an identification number to be assigned to a particular run of the procedure. The number entered is equated to the symbol ID. The **WRITE** commands write a text line concatenated with the symbol name ID and a blank line.

WRITE

The lines between the label `WRITE_LOOP` and `END_LOOP` process information and write additional data to the file. When the processing is finished, control is transferred to the label `END_LOOP`. The `CLOSE` and `PRINT` commands at this label close the output file and queue a copy of the file to the system printer.

The label `WRITE_ERROR` is used as the target of the `/ERROR` qualifier on the `WRITE` command; if an error occurs when a record is being written, control is transferred to the label `WRITE_ERROR`.

DIGITAL Standard Runoff (DSR) Commands

DIGITAL Standard Runoff (DSR) is a text-formatting facility consisting of DSR commands, DSR flags, the DCL command RUNOFF, the DSR Table of Contents Utility, and the DSR Indexing Utility. You enter DSR commands and flags in a file along with the text you want to format. The output file that results from DSR processing is a formatted document. Neither the DSR commands nor the DSR flags appear in the final document.

The following steps summarize the process of producing a document with DSR:

1. Use a text editor such as EDT to create or edit a file that contains DSR commands, DSR flags, and text.
2. Use the RUNOFF command to process your file and format the text according to DSR defaults and DSR commands that you enter.
3. Print the formatted document.

1 DSR Command Format

A DSR command consists of the following parts:

- A **Control flag** (.) that introduces a DSR command. Begin a command in column 1 unless it follows other DSR commands on the same line.
- A **keyword** that immediately follows the Control flag to specify the command function. A keyword can be a single word or several words separated by spaces. The letters of a keyword may be entered in uppercase, lowercase, or both. Keywords may be abbreviated to uniqueness.
- An **argument** that provides additional information for some commands. Use commas or spaces to separate multiple arguments (for example, .LAYOUT 1,3).

Many commands have optional arguments. If you do not enter a value for the argument, DSR supplies a predetermined standard numeric or alphabetic value. This standard value is known as a default.

- A **terminator** that ends the command or string of commands. Commands are most commonly terminated by the end of the line. However, you can terminate a command with a semicolon (;). You can terminate a command and begin a comment with an exclamation point (!). Or you can terminate a command and begin another one with a period (.

2 Entering DSR Commands

You can put each DSR command on a separate line or you can put several DSR commands on the same line. You must always type the Control flag (.) in column 1 of a line. The following example shows a single command on each line:

```
.BLANK  
.LEFT MARGIN 0  
.INDENT 10
```

To put more than one DSR command on a line, you must follow these rules:

- You must type the first command in column 1 of a line.
- You can put one command after another if all commands on the line either take no values or take numeric values.
- You can (except where explicitly disallowed) include a command that takes an alphabetic argument, as long as it is the last command on the line.
- You must precede each command with a Control flag (.).

The following example shows multiple commands on a single line:

```
.BLANK.LEFT MARGIN 0.INDENT 10
```

There are exceptions to these rules. Some commands that take alphabetic values (such as the .DISPLAY commands) can appear anywhere in a line of commands. Other commands take text, but they must be followed by a semicolon (;) for another command to follow on the same line. Some commands that take text after them (.TITLE and .CHAPTER) cannot be followed by any other commands. DSR Commands describes the formats of the individual commands.

To terminate a command or line of commands, you usually enter a carriage return. However, you can terminate a command by typing a semicolon if you want to enter text on the same line with the command. The text must immediately follow the semicolon. For example:

```
We sail the ocean blue,  
.BLANK;And our saucy ship's a beauty.
```

In the preceding example, the semicolon after the .BLANK command tells DSR that the command is terminated and that text now follows. DSR inserts a blank line between the two lines of text, as shown in the following output:

```
We sail the ocean blue,  
  
And our saucy ship's a beauty.
```


3 DSR Commands

This chapter contains an alphabetical list of all DSR commands with a description of each command.

.APPENDIX

The `.APPENDIX` command specifies the beginning of an appendix, assigns an identifying letter to it, and allows you to supply a title. Successive `.APPENDIX` commands assign identifying letters in alphabetical order. (See also `.NUMBER APPENDIX` and `.DISPLAY APPENDIX`.)

.AUTOJUSTIFY, .NO AUTOJUSTIFY

When you enter `.AUTOJUSTIFY`, the following commands automatically execute `.JUSTIFY` (as well as `.FILL`) commands:

`.APPENDIX`
`.CHAPTER`
`.HEADER LEVEL`
`.NOTE`

If you disable automatic justification by entering `.NO AUTOJUSTIFY`, DSR does not disturb either the justify/no-justify or the fill/no-fill states that are in effect (whether by default or as a result of a previous `.JUSTIFY` or `.NO JUSTIFY` command) at the time you use one of these commands. Whichever state is in effect remains in effect when you enter `.NO AUTOJUSTIFY`. (See also `.JUSTIFY`, `.NO JUSTIFY`, `.FILL`, and `.NO FILL`.)

.AUTOPARAGRAPH, .NO AUTOPARAGRAPH

The `.AUTOPARAGRAPH` and `.NO AUTOPARAGRAPH` commands turn the automatic paragraph capability on and off. If `.AUTOPARAGRAPH` is in effect, you do not have to insert `.PARAGRAPH` commands each time you want to format a paragraph. When you start a line with a space or tab or insert a blank line, DSR automatically formats a new paragraph, using the values of `.PARAGRAPH` or `.SET PARAGRAPH`. You can specify values for `.PARAGRAPH` or you can use the default values (see `.PARAGRAPH`). `.AUTOPARAGRAPH` functions the same way `.AUTOTABLE` does, except that `.AUTOTABLE` starts a new paragraph each time a line does *not* start with a space or tab (see `.AUTOTABLE`).

.AUTOSUBTITLE, .NO AUTOSUBTITLE

The .AUTOSUBTITLE command causes DSR to use .HEADER LEVEL titles for running-head subtitles. Subtitles therefore can change according to the section title that applies to a given page. The .NO AUTOSUBTITLE command cancels the .AUTOSUBTITLE function. (See .HEADERS ON, .SUBTITLE, and .HEADER LEVEL.)

.AUTOTABLE, .NO AUTOTABLE

The .AUTOTABLE and .NO AUTOTABLE commands turn the automatic paragraph capability on and off. If .AUTOTABLE is in effect, DSR formats a new paragraph for each line that does not start with a space or tab. It is formatted according to .PARAGRAPH or .SET PARAGRAPH values, whether they are specified or supplied by default (see .PARAGRAPH). .AUTOTABLE functions the same way that .AUTOPARAGRAPH does, except that .AUTOPARAGRAPH starts a new paragraph for each line that starts with a space or tab (see .AUTOPARAGRAPH).

.BLANK

The .BLANK command inserts exactly the number of blank lines that you specify. It is different from .SKIP, which inserts a multiple of the number of blank lines specified in the .SPACING command (see .SKIP and .SPACING).

.BREAK

The .BREAK command ends the current line immediately, without filling or justifying. Enter .BREAK when .FILL is in effect and you want a few short lines of text with no blank lines in between.

.CENTER (.CENTRE)

The .CENTER command centers a single line of text around a character position on a line (compare with .RIGHT).

.CHAPTER

The .CHAPTER command specifies the beginning of a chapter, numbers it, and allows you to supply a chapter title. Successive .CHAPTER commands number the chapters sequentially. (See also .NUMBER CHAPTER and .DISPLAY CHAPTER.)

.CONTROL CHARACTERS, .NO CONTROL CHARACTERS

The .CONTROL CHARACTERS command causes DSR to accept control characters as normal text in your input file. The characters that are affected by this command are the characters in the DEC Multinational Character set with the following decimal values; 1-31, 128-159, and 255. The control characters 0 (NULL) and 127 (DEL) can only be inserted into a document by using the accept flag (␣). A form feed (ASCII 12) must be preceded by the accept flag if used in column 1.

The .NO CONTROL CHARACTERS command does not accept control characters as normal text.

.DATE, .NO DATE

The .DATE and .NO DATE commands control whether the current date appears in running heads. The date appears in the following format: 22 August 1988. The .SUBTITLE command must be included for .DATE to be effective. (See also .HEADERS ON and .SET DATE.)

.DISPLAY APPENDIX

The .DISPLAY APPENDIX command allows you to specify the form that the lettering (or numbering) of appendixes will take. The form you specify appears in the title, the page numbers, and the first character of header level numbers throughout the appendix. This command does not change any values; it only affects the way the values are displayed. (See also .APPENDIX and .NUMBER APPENDIX.)

.DISPLAY CHAPTER

The `.DISPLAY CHAPTER` command allows you to specify the form that the numbering (or lettering) of chapters will take. The form you specify appears in the title, the page numbers, and the first character of header level numbers throughout the chapter. This command does not change any values; it only affects the way the values are displayed. (See also `.CHAPTER` and `.NUMBER CHAPTER`.)

.DISPLAY ELEMENTS

The `.DISPLAY ELEMENTS` command allows you to specify the form that sequential numbering or lettering of items in a list will take. This command does not change any values; it only affects the way the values are displayed. (See also `.LIST`, `.END LIST`, and `.NUMBER LIST`.)

.DISPLAY LEVELS

The `.DISPLAY LEVELS` command allows you to specify the form that sequential numbering (or lettering) of section headers will take. You can control the form of individual numbers within a section number for a header (that is, those numbers preceding or following a dot). This command does not change any values; it only affects the way the values are displayed. (See also `.HEADER LEVEL`, `.NUMBER LEVEL`, and `.STYLE HEADERS`.)

Default Header Level Numbering

	Nonchapter	Chapter n	Appendix A
<code>.HEADER LEVEL 1</code>	1	n.1	A.1
<code>.HEADER LEVEL 2</code>	1.1	n.1.1	A.1.1
<code>.HEADER LEVEL 3</code>	1.1.1	n.1.1.1	A.1.1.1

.DISPLAY NUMBER

The **.DISPLAY NUMBER** command allows you to specify the form that sequential numbering (or lettering) of pages will take. This command does not change any values; it only affects the way the values are displayed. (See also **.HEADERS ON**, **.NUMBER PAGE**, **.NO NUMBER**, **.LAYOUT**, **.NUMBER RUNNING**, and **.NO PAGING**.)

.DISPLAY SUBPAGE

The **.DISPLAY SUBPAGE** command allows you to specify the form that sequential lettering (or numbering) of subpage characters will take. Subpage characters are the characters that are appended to the page numbers of subpages. This command does not change any values; it only affects the way the values are displayed. (See also **.SUBPAGE** and **.NUMBER SUBPAGE**.)

.ENABLE BAR, .DISABLE BAR, .BEGIN BAR, .END BAR

The bar commands control the insertion of vertical bars (|) at the beginning of lines of text. The bars (usually called change bars) are normally inserted to indicate where changes in text have occurred since the previous edition of a document. You can specify a character other than the default character (vertical bars) to indicate changes.

The **.ENABLE BAR** command shifts all text following it three spaces to the right to make room for the bars on the left. The width of the lines of actual text is not altered.

The **.BEGIN BAR** command causes DSR to start inserting vertical bars at the beginning of lines.

The **.END BAR** command causes DSR to stop putting vertical bars at the beginning of lines.

The **.DISABLE BAR** command disables the bar commands but does not shift the lines of text back to their original position.

.ENABLE BOLDING, .DISABLE BOLDING

The **.ENABLE BOLDING** and **.DISABLE BOLDING** commands enable and disable the bolding function. You can perform bolding only if recognition of the Bold flag (*) is turned on and the bold function is enabled.

.ENABLE HYPHENATION, .DISABLE HYPHENATION

The **.ENABLE HYPHENATION** and **.DISABLE HYPHENATION** commands enable and disable the hyphenation function.

You can use hyphenation to close up excessive spacing between words. Extra spaces often are placed between words when margins are narrow and a line contains several long words.

.ENABLE INDEXING, .DISABLE INDEXING

These commands enable and disable the operation of the indexing commands (**.INDEX** and **.ENTRY**), and the Index flag (>).

.ENABLE OVERSTRIKING, .DISABLE OVERSTRIKING

The **.ENABLE OVERSTRIKING** and **.DISABLE OVERSTRIKING** commands enable and disable the overstrike function.

You use the Overstrike flag (%) to create special characters that are not available on the terminal by overstriking any printing character with another. For example, you can overstrike a 7 with a hyphen to create a European 7.

.ENABLE TOC, .DISABLE TOC

These commands enable and disable DSR's collection of information for the table of contents.

.ENABLE UNDERLINING, .DISABLE UNDERLINING

The `.ENABLE UNDERLINING` and `.DISABLE UNDERLINING` commands enable and disable the underline function. You can perform underlining only if recognition of the Underline flag (`&`) is turned on and the underline function is enabled.

.ENTRY

The `.ENTRY` command creates an index entry without a page number reference. It is usually used for "See . . ." or "See also . . ." index entries.

.FIGURE DEFERRED, .FIGURE

The `.FIGURE DEFERRED` command leaves room on a page for you to insert a figure later. You specify the number of blank lines you need, and DSR leaves that amount of space on the current page if there is enough room.

If there is not enough room on the current page, `.FIGURE DEFERRED` first adds enough text to complete the page and then puts the required number of blank lines at the top of the next page.

The `.FIGURE` command is the same as `.FIGURE DEFERRED` except that, if there is not enough room on the current page, DSR ends the page immediately and then puts the blank lines at the top of the next page.

.FILL, .NO FILL

The `.FILL` command causes DSR to treat line endings exactly like spaces (see also `.NO SPACE`). Line-filling is the accumulation of words on a line until the addition of one more word would exceed the right margin. If `.NO FILL` is in effect, line endings in the input file are duplicated in the output file (see also `.KEEP`).

.FIRST TITLE

The **.FIRST TITLE** command allows running-head information to appear on the first page of a document with no chapters. (See also **.HEADERS ON**, **.LAYOUT**, **.TITLE**, **.SUBTITLE**, and **.AUTOSUBTITLE**.)

.FLAGS ACCEPT, .NO FLAGS ACCEPT

The **.FLAGS ACCEPT** and **.NO FLAGS ACCEPT** commands turn on and turn off recognition of the Accept flag character (_).

.FLAGS ALL, .NO FLAGS ALL

The **.FLAGS ALL** and **.NO FLAGS ALL** commands function as master switches for all other **.FLAGS/NO FLAGS** flag-name command settings, except the **.FLAGS/NO FLAGS COMMENT** and **.FLAGS/NO FLAGS CONTROL** commands.

The **.FLAGS ALL** and **.NO FLAGS ALL** commands turn on and turn off recognition of all flags without disturbing other flag command settings. (An analogy for flag recognition is turning on a master switch [entering **.FLAGS ALL**] — those lights whose switches are in the ON position will go on and those whose switches are in the OFF position will not go on.) See also **.ENABLE/.DISABLE BOLDING**, **HYPHENATION**, **INDEXING**, **OVERSTRIKING**, and **UNDERLINING** commands.

.FLAGS BOLD, .NO FLAGS BOLD

The **.FLAGS BOLD** and **.NO FLAGS BOLD** commands turn on and turn off recognition of the Bold flag character (*).

.FLAGS BREAK, .NO FLAGS BREAK

The **.FLAGS BREAK** and **.NO FLAGS BREAK** commands turn on and turn off recognition of the Break flag character (|).

.FLAGS CAPITALIZE, .NO FLAGS CAPITALIZE

The **.FLAGS CAPITALIZE** and **.NO FLAGS CAPITALIZE** commands turn on and turn off recognition of the Capitalize flag character (<).

.FLAGS COMMENT, .NO FLAGS COMMENT

The **.FLAGS COMMENT** and **.NO FLAGS COMMENT** commands turn on and turn off recognition of the Comment flag character (!).

.FLAGS CONTROL, .NO FLAGS CONTROL

These commands control recognition of the Control flag character (the dot that begins a DSR command). You can enter **.FLAGS CONTROL** to change the character that precedes the commands from a dot to a character of your choice. You can enter **.NO FLAGS CONTROL** to turn off recognition of the Control flag character.

NOTE: There is no way to reenable recognition of the Control flag once you enter the **.NO FLAGS CONTROL** command.

.FLAGS HYPHENATE, .NO FLAGS HYPHENATE

The **.FLAGS HYPHENATE** and **.NO FLAGS HYPHENATE** commands turn on and turn off recognition of the Hyphenate flag character (=).

.FLAGS INDEX, .NO FLAGS INDEX

These commands respectively turn on and turn off recognition of the Index flag character (>).

.FLAGS LOWERCASE, .NO FLAGS LOWERCASE

The **.FLAGS LOWERCASE** and **.NO FLAGS LOWERCASE** commands turn on and turn off recognition of the Lowercase flag character (\).

.FLAGS OVERSTRIKE, .NO FLAGS OVERSTRIKE

The **.FLAGS OVERSTRIKE** and **.NO FLAGS OVERSTRIKE** commands enable and disable recognition of the Overstrike flag character (%).

.FLAGS PERIOD, .NO FLAGS PERIOD

The **.FLAGS PERIOD** and **.NO FLAGS PERIOD** commands turn on and turn off recognition of the Period flag character (+).

.FLAGS SPACE, .NO FLAGS SPACE

The **.FLAGS SPACE** and **.NO FLAGS SPACE** commands turn on and turn off recognition of the Space flag character (#).

.FLAGS SUBINDEX, .NO FLAGS SUBINDEX

The **.FLAGS SUBINDEX** and **.NO FLAGS SUBINDEX** commands turn on and turn off recognition of the Subindex flag (>). You can also use the **.FLAGS SUBINDEX** command to change the Subindex flag to another character. If you enter **.NO FLAGS SUBINDEX**, the command will cause a right angle bracket (>) to be printed as part of your indexed text, instead of causing subindexing.

.FLAGS SUBSTITUTE, .NO FLAGS SUBSTITUTE

The **.FLAGS SUBSTITUTE** and **.NO FLAGS SUBSTITUTE** commands turn on and turn off recognition of the Substitute flag character (\$). The default Substitute flag character (\$) or any replacement character you specify must be used in pairs.

.FLAGS UNDERLINE, .NO FLAGS UNDERLINE

The **.FLAGS UNDERLINE** and **.NO FLAGS UNDERLINE** commands turn on and turn off recognition of the Underline flag character (&).

.FLAGS UPPERCASE, .NO FLAGS UPPERCASE

The **.FLAGS UPPERCASE** and **.NO FLAGS UPPERCASE** commands turn on and turn off recognition of the Uppercase flag (^).

.FOOTNOTE, .END FOOTNOTE

The **.FOOTNOTE** command places the text following it at the bottom of the current page if there is room. If there is not enough room on the current page for the entire footnote, DSR places the entire note at the bottom of the next page.

The **.END FOOTNOTE** command ends the footnote and restores any case, fill, justify, spacing, or margin settings that you might have changed within the footnote.

The right margin of the footnote will be the same as the right margin in effect for the document at the time the footnote is created. If you change the right margin of the document but want the right margin of all footnotes to be the same, enter the **.RIGHT MARGIN** command immediately after each **.FOOTNOTE** command to set the same right margin for each footnote.

The left margin setting of the footnote is defaulted to 0.

.HEADER LEVEL

The **.HEADER LEVEL** command allows you to specify both a section number and a section title. Successive **.HEADER LEVEL** commands of the same value (all **.HEADER LEVEL 1**'s for example) cause the section numbers to increase sequentially. This happens at all six levels of headers. If your current section is in Chapter 2 and is numbered 2.5.2.4, then the following numbering would result depending upon the **.HEADER LEVEL** command you used:

- **.HL3** (or **.HL** without a value) would number the next section 2.5.2.5
- **.HL2** would number the next section 2.5.3
- **.HL1** would number the next section 2.6

DSR-14 DIGITAL Standard Runoff (DSR) Commands

.HEADER LEVEL

(See also `.DISPLAY LEVELS`, `.NUMBER LEVEL`, `.SET LEVEL`, and `.STYLE HEADERS`.)

Following is a summary of default header level numbering for three levels of three different types of documents:

Default Header Level Numbering

	Nonchapter	Chapter n	Appendix A
<code>.HEADER LEVEL 1</code>	1	n.1	A.1
<code>.HEADER LEVEL 2</code>	1.1	n.1.1	A.1.1
<code>.HEADER LEVEL 3</code>	1.1.1	n.1.1.1	A.1.1.1

.HEADERS ON, .NO HEADERS

The `.HEADERS ON` and `.NO HEADERS` commands restore and cancel, respectively, the capability of having one or two lines of information at the top of a page. These lines indicate the content of the page and the page number. They are called running heads, which you should not confuse with section heads (specified with `.HEADER LEVEL` commands).

.HEADERS UPPER, .HEADERS LOWER, .HEADERS MIXED

The `.HEADERS UPPER/LOWER/MIXED` commands specify the case of the word *page* that precedes the page number. The commands produce, respectively, `PAGE`, `page`, and `Page`. In an index, these commands also affect the word *index* that is part of the page number, for example, `Page Index-3`. The command normally takes effect on the next page.

.IF, .IFNOT, .ELSE, .ENDIF

The `.IF`, `.IF NOT`, `.ELSE`, and `.ENDIF` commands (also known as the conditional commands) cause portions of a DSR file to be processed or not processed, according to conditions that you specify. The commands refer to the `/VARIANT` qualifier that you specify on the DSR command line. (See also `/DEBUG=CONDITIONALS` and `.VARIABLE`.)

.INDENT

The **.INDENT** command causes the first line of text following it to begin at a position relative to the left margin.

.INDEX

The **.INDEX** command creates an index entry with a page number reference.

.JUSTIFY, .NO JUSTIFY

The **.JUSTIFY** command causes DSR to insert exactly enough space between words so that the last character reaches the right margin. The **.NO JUSTIFY** command disables justification.

.KEEP, .NO KEEP

The **.KEEP** command allows you to keep in the output file blank lines that are present in the input file when **.NO FILL** is in effect. Normally, multiple blank lines in the input file are discarded in the output file while **.NO FILL** is in effect. **.NO KEEP** also discards blank lines when **.NO FILL** is in effect. (See also **.LITERAL**.)

.LAYOUT

The **.LAYOUT** command rearranges running-head and running-foot information on pages. (See the **.HEADERS ON** command.) When the default **.LAYOUT** operates, page numbering is not displayed on the first page, it starts on page 2.

.LEFT MARGIN

The **.LEFT MARGIN** command sets the left margin to the specified position.

.LIST, .END LIST

The **.LIST** command specifies the beginning of a list by resetting the left margin farther to the right, by setting a **.SKIP** command value to take effect before each item in the list, and by executing **.TEST PAGE**. Use the **.LIST ELEMENT** command to specify each item in the list. **.LIST ELEMENT** commands also give you numbers or letters in sequence in the left margin or let you substitute a single character of your choice for each of the numbers or letters (for example, the lowercase letter o, which is known as a "bullet"). (See also **.DISPLAY ELEMENTS** and **.NUMBER LIST**.)

The **.END LIST** command ends a list, restoring any fill, justify, case, margin, or spacing settings that were in effect before you entered the most recent **.LIST** command. You can also specify a value with **.END LIST** that puts blank lines after the last item in the list (as with **.SKIP**).

.LIST ELEMENT

The **.LIST ELEMENT** command specifies the beginning of each item in a list. If you specify a character in a **.LIST** command, it appears, followed by two spaces, before each item. Otherwise, a sequence of numbers or letters, as defined in the **.DISPLAY ELEMENTS** command, appears when you enter successive **.LIST ELEMENT** commands. If you have not entered the **.DISPLAY ELEMENTS** command, you will get a sequence of decimal numbers, each followed by a period and two spaces. (See **.LIST**, **.END LIST**, **.DISPLAY ELEMENTS**, and **.NUMBER LIST**.)

.LITERAL

The **.LITERAL** command allows you to have your text formatted exactly as you have typed it. This means that you will get a blank line in the output file wherever a blank line occurs in the input file. (If the value specified by the **.SPACING** command is anything other than one, you will get the spacing value that you specified.)

Commands are not recognized when **.LITERAL** is in effect and are treated as ordinary text if you enter them. DSR flags are also treated as normal text. Tab stops set prior to the **.LITERAL** command, however, are still in effect within the block of **.LITERAL** text (see **.TAB STOPS**). You must enter **.END LITERAL** when you want DSR to resume normal formatting.

.NO SPACE

The **.NO SPACE** command prevents the insertion of the end-of-line space for one line of text only, causing the characters at the end of one line and the beginning of the next to be adjacent.

Without the **.NO SPACE** command, when **.FILL** is in effect, DSR treats the end of an input line exactly like a space. That is, it inserts a space in the output file at the place where each input line ended (this is the meaning of "fill\nospace").

If you ever have occasion to use this command, you should enter it immediately after the end-of-line space that you want to affect.

.NOTE, .END NOTE

The **.NOTE** command highlights a portion of text by narrowing the margin settings, centering the text on the page, and printing a title centered over the text.

The **.END NOTE** command restores the fill, justify, case, margin, and spacing settings that were in effect just before you entered the **.NOTE**.

.NUMBER APPENDIX

The **.NUMBER APPENDIX** command allows you to specify an identifying letter with which a sequence of appendixes will begin. The next **.APPENDIX** command starts the sequence. Subsequent **.APPENDIX** commands cause appendixes to be lettered in alphabetic order. See also **.DISPLAY APPENDIX**.

.NUMBER CHAPTER

The **.NUMBER CHAPTER** command allows you to specify the number with which a sequence of chapters will begin. The next **.CHAPTER** command starts the sequence. Subsequent **.CHAPTER** commands will cause each chapter to be numbered one higher than the previous chapter. (See also **.DISPLAY CHAPTER**.)

.NUMBER LEVEL

The **.NUMBER LEVEL** command allows you to specify the beginning number of a sequence of headers. Enter this command immediately before the first **.HEADER LEVEL** command that you want to affect. Subsequent **.HEADER LEVEL** commands will each be one higher than the preceding one according to its level (see **.HEADER LEVEL**). (See also **.STYLE HEADERS** and **.DISPLAY LEVELS**.)

Default Header Level Numbering

	Nonchapter	Chapter n	Appendix A
.HEADER LEVEL 1	1	n.1	A.1
.HEADER LEVEL 2	1.1	n.1.1	A.1.1
.HEADER LEVEL 3	1.1.1	n.1.1.1	A.1.1.1

.NUMBER LIST

The **.NUMBER LIST** command allows you to specify, anywhere in a list, the number with which a sequence of items in a list will begin. Enter this command just before the **.LIST ELEMENT** command that you want to affect. Subsequent list elements will each have a number that is one greater than the number for the preceding **.LIST ELEMENT** command. (See also **.DISPLAY ELEMENTS**, with which you can specify the form the number will take.)

.NUMBER PAGE, .NO NUMBER

The **.NO NUMBER** command suspends normal page numbering. The **.NUMBER PAGE** command resumes normal page numbering, having kept track of the numbering while **.NO NUMBER** was in effect; or it allows you to specify the beginning of a new number sequence by specifying a number for the next page. (See also **.NUMBER RUNNING**, **.DISPLAY NUMBER**, **.NO PAGING**, and **.HEADERS ON**.)

.NUMBER RUNNING

The **.NUMBER RUNNING** command allows you to specify the beginning of a new sequence of running page numbers. This command affects page numbers only if you have entered a **.LAYOUT** command with an *n1* value of 3. (See **.LAYOUT**, **.HEADERS ON**, and **.NO NUMBER**.)

.NUMBER SUBPAGE

The **.NUMBER SUBPAGE** command allows you to specify the beginning of a new sequence of subpage numbers, for example, 1-16A, 1-16B, 1-16C, and so on. This command affects only the letters that the **.SUBPAGE** command appends to the normally numeric page number. **.NUMBER SUBPAGE** takes effect on the next page. (See also **.SUBPAGE** and **.DISPLAY SUBPAGE**.)

.PAGE

The **.PAGE** command starts a new page.

.PAGE SIZE

The **.PAGE SIZE** command sets the page "frame" by specifying the page length (the maximum number of lines of text on a page) and the page width for the running heads. (Compare with **.RIGHT MARGIN**, which sets the text width.) The width component of **.PAGE SIZE** and the value established by **.RIGHT MARGIN** are separate values.

.PAGING, .NO PAGING

The **.PAGING** command enables paging. The **.NO PAGING** command disables it.

.PARAGRAPH

.PARAGRAPH

The **.PARAGRAPH** command controls spacing and page placement associated with the creation of paragraphs. It executes **.TEST PAGE**, followed by **.SKIP** and **.INDENT**. (See also **.SET PARAGRAPH**.)

.PERIOD, .NO PERIOD

DSR normally adds an extra space after any of the following punctuation marks in your text: period (.), colon (:), question mark (?), and exclamation point (!).

The **.NO PERIOD** command cancels the extra space that DSR inserts after any of the punctuation marks listed above. The **.NO PERIOD** command is used to differentiate between punctuation used in the syntax of a sentence and punctuation used for other purposes.

The **.PERIOD** command restores the routine insertion of an extra space following any of the punctuation marks listed above.

.REPEAT

The **.REPEAT** command allows you to specify up to 150 characters to be printed a specified number of times, either horizontally or vertically.

.REQUIRE

The **.REQUIRE** command allows you to process several DSR files at the same time and merge them in an output file.

.RIGHT

The **.RIGHT** command positions a single line of text relative to the right margin. (See also **.CENTER**.)

.RIGHT MARGIN

The **.RIGHT MARGIN** command sets the right margin to the position that you specify. This is the position to which a line of text normally extends. If **.JUSTIFY** is in effect, the **.RIGHT MARGIN** value is the position against which text is justified. If **.NO JUSTIFY** is in effect, the **.RIGHT MARGIN** value specifies the maximum number of characters on any text line. (Compare with **.PAGE SIZE**, which sets the page width for running heads.)

.SAVE, .RESTORE

These commands maintain the formatting context of a document for the user. The files produced by the DSR utilities make changes to the formatting context. In order not to disturb the user's context, the RNT and RNX files execute **.SAVE** and **.RESTORE** commands.

.SAVE stores information about the current RUNOFF formatting context; this includes DSR defaults and DSR commands and flags issued by the user.

.RESTORE restores the formatting information saved by the last-issued **.SAVE** command.

.SEND TOC

The **.SEND TOC** command allows you to insert DSR commands, DSR flags, and text into the table of contents (RNT) file. The items that you insert affect the appearance of the table of contents. For example, you can send emphasis flag characters to cause bolding and underlining in the table of contents.

.SET DATE, .SET TIME

The **.SET DATE** and **.SET TIME** commands let you specify a date and time to be inserted in your file when you issue the Substitute flag pair, **\$\$**, with any of the appropriate date or time parameters. **.SET DATE** also sets the date for the **.DATE** command, which causes the date to appear in running heads.

.SET LEVEL

The **.SET LEVEL** command allows you to preset the level of the next section head without entering a **.HEADER LEVEL** command (see **.HEADER LEVEL**).

.SET PARAGRAPH

The **.SET PARAGRAPH** command allows you to set values for **.PARAGRAPH** without entering **.PARAGRAPH**. The **.SET PARAGRAPH** command can be especially useful if you plan to execute **.AUTOPARAGRAPH** or **.AUTOTABLE**. (See **.PARAGRAPH**.)

.SKIP

The **.SKIP** command inserts a multiple of the number of blank lines that has been specified by the **.SPACING** command. Contrast this with **.BLANK**, which inserts only the number of blank lines specified with the **.BLANK** command itself. (See **.BLANK**.)

.SPACING

The **.SPACING** command changes the amount of spacing between lines of text.

.STYLE HEADERS

The **.STYLE HEADERS** command changes the formats of the various levels of section heads (**.HEADER LEVEL n**). Do not confuse the numbers that identify the header level (in the range of 1 through 6) with the numbers that get printed just to the left of the header title (3.5.2, for example). See **.HEADER LEVEL**. (See also **.NUMBER LEVEL** and **.DISPLAY LEVEL**.)

Default Header Level Numbering

	Nonchapter	Chapter n	Appendix A
.HEADER LEVEL 1	1	n.1	A.1
.HEADER LEVEL 2	1.1	n.1.1	A.1.1
.HEADER LEVEL 3	1.1.1	n.1.1.1	A.1.1.1

.SUBPAGE, .END SUBPAGE

The **.SUBPAGE** command begins a new page and a new format for page numbering. It numbers the new page by keeping the previous page number and appending the letter A to it. For example, if the previous page is 10, the first subpage is 10A and the next page becomes 10B unless you have entered an **.END SUBPAGE** in the meantime. (See also **.NUMBER SUBPAGE**, **.DISPLAY SUBPAGE**, **.HEADERS ON**, **.LAYOUT**, and **.PAGE**.)

The **.END SUBPAGE** command begins a new page and goes back to normal page numbering. If you entered the **.END SUBPAGE** command on page 2-8D, for example, the next page would be numbered 2-9.

.SUBTITLE, .NO SUBTITLE

The **.SUBTITLE** command allows you to specify a subtitle for a running head (see **.HEADERS ON**). When using the default **.LAYOUT** command, the subtitle appears on the second line of every page (except page 1) at the leftmost position on a line (character position 0), regardless of the left margin setting. The **.NO SUBTITLE** command cancels the **.SUBTITLE** command. (See also **.AUTOSUBTITLE**, **.TITLE**, **.FIRST TITLE**, and **.LAYOUT**.)

.TAB STOPS

The **.TAB STOPS** command changes the current positions of tab stops. Each tab character in the input file advances the print carriage to the right to the next tab stop.

.TEST PAGE

The **.TEST PAGE** command allows you to keep a specified amount of text entirely on a single page. If there is not enough room on the current page to accommodate that amount, DSR ends the current page and puts the entire text on the next page.

.TITLE

The **.TITLE** command allows you to specify a title for a running head (see **.HEADERS ON**). This title normally appears at the top of every page but the first, at the leftmost position on the line (character position 0), regardless of the **.LEFT MARGIN** setting. (See also **.FIRST TITLE**, **.SUBTITLE**, and **.LAYOUT**.)

.VARIABLE

The **.VARIABLE** command allows you to specify a character that corresponds to the name you have given the commands and text in an **.IF** (or **.IFNOT**) block. This identifying character is placed in the left margin when you process your file with the **/DEBUG** or **/DEBUG=CONDITIONALS** command line qualifier.

.XLOWER, .XUPPER

The **.XLOWER** command allows you to control the case of index entries specified by the **.INDEX** and the **.ENTRY** commands, or by the **Index** flag (**>**). The case of the index entries will match exactly the case that you enter when you make the index entry.

The **.XUPPER** command lets DSR control the case of index entries. If **.XUPPER** is in effect (as it is by default), DSR capitalizes the first character of every index entry, and drops everything else in the entry to lowercase.

EDT Keypad Commands

This section contains descriptions of the EDT keypad commands, qualifiers, and specifiers. The command descriptions are given in alphabetical order.

ADVANCE Function

description

Pressing ADVANCE sets the direction for subsequent editing work to forward (to the right of the cursor and down toward the end of the buffer).

ADVANCE is the default direction and remains in effect until you press BACKUP.

You can also use the ADVANCE key to set the direction of and process the FIND function. Press the GOLD and FIND keys, enter the string you want to locate, then press ADVANCE to move the cursor forward to find the string.

APPEND Function

description

Pressing APPEND deletes the select range from the current buffer and adds it to the end of the PASTE buffer. The previous contents of the PASTE buffer are not deleted.

BACKSPACE Function CTRL/H

description

Pressing BACKSPACE causes the cursor to move to the beginning of the current line. If the cursor is already at the beginning of a line, pressing BACKSPACE moves it to the beginning of the previous line.

The BACKSPACE key and CTRL/H always have the same preset function in EDT. When you redefine the BACKSPACE key, you redefine CTRL/H (except for terminals with LK201 keyboards when they are operating in VT200 mode). To redefine the BACKSPACE key using the line-mode DEFINE KEY command, enter DEFINE KEY CONTROL H. To find the definition of the BACKSPACE key, enter SHOW KEY CONTROL H. For terminals with LK201 keyboards, use DEFINE KEY FUNCTION 24 and SHOW KEY FUNCTION 24 for the F12 key.

BACKUP Function

description

Pressing BACKUP sets the direction for subsequent editing work to backward (to the left of the cursor toward the beginning of the buffer).

You can use the BACKUP key to set the direction of and process the FIND function. Press the GOLD and FIND keys, enter the string you want to locate, and then press BACKUP to move the cursor backward to find the string.

To change EDT's direction to forward, use ADVANCE. The RESET function also sets EDT's direction to forward.

BOTTOM Function

description

Pressing BOTTOM moves the cursor to the end of the buffer, after the last character position in the buffer. The cursor is positioned at the end of buffer ([EOB]) mark.

CHAR (Character) Function

description

Pressing CHAR (character) moves the cursor one character in the current direction (forward or backward, depending on whether ADVANCE or BACKUP is in effect).

This key is not on the VT52 keypad. However, the LEFT and RIGHT arrow keys can be used to move the cursor one character position.

CHNGCASE (Change Case) Function

description

Pressing CHNGCASE (change case) changes the case of letters in your text. Uppercase letters become lowercase; lowercase letters become uppercase. The number of letters affected by this function depends on several factors: active select range, cursor location, SET SEARCH parameter, and repeat count. The following chart shows what happens when you use CHNGCASE under various conditions:

EDT Keypad Commands EDT-3

CHNGCASE (Change Case) Function

Conditions	Results
SELECT RANGE ACTIVE	Changes the case of every letter in the select range. All lowercase letters become uppercase; all uppercase letters become lowercase.
NO SELECT RANGE ACTIVE	
1a. SET SEARCH BEGIN in effect. Cursor on first character of current search string. Repeat count = 0 or 1.	Changes the case of every letter in the search string. All lowercase letters become uppercase; all uppercase letters become lowercase. Moves the cursor to the character to the right of the search string.
1b. SET SEARCH END in effect. Cursor to the right of current search string. Repeat count = 0 or 1.	Changes the case of every letter in the search string. All lowercase letters become uppercase; all uppercase letters become lowercase. Moves the cursor to the first character of the search string.
2a. Current direction is forward. *Cursor not at active end of current search string. Repeat count = 0 or 1.	Changes the case of the letter that the cursor is on. Moves the cursor one column to the right.
2b. Current direction is backward. *Cursor not at active end of current search string. Repeat count = 0 or 1.	Changes the case of the letter to the left of the cursor. The cursor remains on the altered letter.
3a. Repeat count greater than 1. Current direction is forward.	Changes the case of the number of letters given in the repeat count. Moves the cursor to the right of the last altered character.
3b. Repeat count greater than 1. Current direction is backward.	Changes the case of the number of letters given in the repeat count. Moves the cursor to the leftmost altered letter.
* Active end of select range:	
SET SEARCH BEGIN	Positions cursor on first character of current search string.
SET SEARCH END	Positions cursor one character to the right of current search string.

COMMAND Function

description

Pressing COMMAND enables you to enter a line-mode command while EDT is still in keypad mode. When you press GOLD and COMMAND, EDT prompts you with *Command:* at the bottom of the screen. Enter the line-mode command you want to use, for example EXIT, then press the ENTER key on the keypad to send the command to EDT.

Use CTRL/Z to shift from keypad to line editing if you want to enter several line-mode commands in a row.

You can enter two or more line-mode commands on the same line by separating the commands with semicolons. If you want to put nokeypad commands after the line-mode command, use CHANGE ;nokeypad-commands form. You can use EDT macros with the COMMAND function just as you would any other line-mode command.

CTRL/A (Control A) Function

description

Pressing CTRL/A establishes the tab position at the present cursor position and resets the indentation level count to be the quotient of the cursor position divided by the SET TAB value. To use CTRL/A, the current cursor position must be a multiple of the SET TAB value. For example, if the SET TAB value is 5, you can use CTRL/A for cursor locations at positions 5, 10, 15, 20, and so on. If the cursor is at some other column, for example 13, and you press CTRL/A, EDT prints an error message.

CTRL/A does not move text. You must use the TAB function to indent a line. CTRL/A works only if SET TAB is in effect. EDT's default is SET NOTAB.

CTRL/C (Control C) Function

description

Pressing CTRL/C interrupts certain operations before EDT finishes processing them. You can use CTRL/C to stop a runaway search through a long file or to stop a long repeat count. CTRL/C can also be used to halt certain EDT operations. For example, you can use CTRL/C to stop EDT from displaying a whole buffer when you use the line-mode TYPE command to move to another buffer.

When EDT aborts the operation, it prints the message "Aborted by CTRL/C." If EDT cannot stop a particular process, it prints the message "CTRL/C ignored."

Note: You cannot redefine the CTRL/C function.

CTRL/D (Control D) Function

description

Pressing CTRL/D decreases the TAB level count one tab setting. The tab level count is the multiple of the SET TAB value that determines the tab indentation level. Suppose the SET TAB value is 5, the tab level count is 3, and the current indentation level is 15. If you press CTRL/D and then the TAB key, the subsequent text will be moved over 10 columns. The SET TAB value is still 5, but the tab level count is now 2 and the current indentation level is 10.

CTRL/D does not move text. You must use the TAB function to indent a line. CTRL/D only works if SET TAB is in effect. EDT's default is SET NOTAB.

CTRL/E (Control E) Function

description

Pressing CTRL/E increments the tab level count by 1. The tab level count is the multiple of the SET TAB value that determines the tab indentation level. Suppose the SET TAB value is 5, the tab level count is 2, and the current indentation level is 10. If you press CTRL/E and then the TAB key, the subsequent text will be moved over 15 columns. The SET TAB value is still 5, but the tab level count is now 3 and the current indentation level is 15.

CTRL/E does not move text. You must use the TAB function to indent a line. CTRL/E only works if SET TAB is in effect. EDT's default is SET NOTAB.

CTRL/K (Control K) Function description

Pressing CTRL/K starts the key definition process in keypad mode. You can create key definitions using both nokeypad commands and predefined function keys. You can define a key by using only nokeypad commands, by pressing predefined function keys, or by combining nokeypad commands with pressing predefined function keys.

Five types of keys can be defined or redefined:

- A keypad key with or without GOLD

All keypad keys can be redefined.

If the current definition of a key is GOLD, you must use the line-mode DEFINE KEY command to redefine that key. GOLD is the default definition for the top left-hand keypad key (PF1 on VT100; blue on VT52). If you are defining a key to have the RESET function, you must enter the word RESET in the definition line.

- CONTROL with a keyboard character, with or without GOLD

EDT does not allow you to redefine CTRL/C. Some CONTROL character combinations are system commands and, for that reason, cannot be redefined. These include O, P, Q, S, X, Y, and [.

Note that you cannot press CTRL/U to enter its definition in the definition line.

- GOLD with a keyboard character

GOLD can be used with any keyboard character except the digits 0 through 9 and the minus sign.

- The DELETE key with or without GOLD

The DELETE key can be redefined by itself or with GOLD.

You cannot press DELETE or <X> to enter its definition in the definition line.

- FUNCTION keys on the LK201 keyboard

These include the six keys located above the arrow keys on the terminal's "editing" keypad as well as keys F6 through F20 on the function key row across the top of the keyboard.

To define a key in keypad-mode, first press CTRL/K. EDT displays the message:

Press the key you wish to define

EDT Keypad Commands EDT-7

CTRL/K (Control K) Function

Press the key or key sequence you want to define. EDT then displays the message:

Now enter the definition and press ENTER

Enter your definition and/or press existing keypad function keys and then press the ENTER key as instructed. (The DO key on LK201 keyboards does not process a CTRL/K key definition.) The process is now complete. If you want to review the definition of any key, use the line-mode SHOW KEY command.

Any nokeypad command can be used in a key definition. The same entity specifiers that are available in nokeypad mode are available for key definitions. Most preset EDT function keys have nokeypad definitions. (GOLD and RESET are exceptions.) Use the SHOW KEY command to see these definitions.

When you look at the definitions for most preset functions, notice that they end with a period. This period is the definition for the ENTER function. When you complete your definition with a period, EDT processes the command as soon as you press the key or key sequence. If the period is omitted, EDT stores the command and does not show the results until you press RETURN, ENTER, or another function key with the period at the end of its definition. For example, if you define GOLD/P to be 2DL., when you press the GOLD key and then the P key, two lines will disappear from your screen. If the definition is 2DL, no change appears on the screen after pressing GOLD and then P. But as soon as you press another function key, the two lines vanish. Be sure to use the period key on the main keyboard, not the one on the keypad, to complete your definitions.

CTRL/L (Control L) Function

description

Pressing CTRL/L inserts a form feed character (<FF>) into your text. You can also use CTRL/L to enter a form feed in search strings and SET commands.

CTRL/M (Control M) Function

description

Pressing CTRL/M inserts a carriage return character (<CR>) into your text. You can also use CTRL/M to enter a carriage return character (<CR> or ^M) in strings and SET commands. CTRL/M is not identical to an EDT line terminator. However, in keypad mode, you can use CTRL/M to mean a line terminator in search and substitute strings.

When you redefine the CTRL/M key sequence, you also automatically redefine the RETURN key. It is recommended that you do not alter the preset definition of CTRL/M for that reason.

CTRL/R (Control R) Function

description

Pressing CTRL/R (in keypad mode) refreshes the screen display. This function has no effect on the text you are editing. It simply clears and redraws the screen, eliminating any extraneous characters or messages that might have appeared on the screen but are not part of the current text you are editing. Note that CTRL/R performs the same function as CTRL/W in keypad-mode.

CTRL/T (Control T) Function

description

Pressing CTRL/T indents the lines in a select range that must contain only whole lines. After creating the select range, press CTRL/T to move the select range lines over one tab stop to the right. Use a repeat count to indent the lines more than one tab stop. To move the text one tab stop to the left, press GOLD and then the minus sign (-) before CTRL/T. You can move the lines several tab stops to the left by using both the minus sign and a repeat count.

CTRL/T works only if SET TAB is in effect. EDT's default is SET NOTAB. To determine the current SET TAB value, use the SHOW TAB command. Note that CTRL/T is not affected by the tab level count, nor does that count have any effect on how far text is indented.

When the DCL command SET CONTROL=T is in effect, you cannot use CTRL/T in EDT to perform tabbing. If DCL is set to NOCONTROL=T (the default), CTRL/T will perform tabbing in EDT. GOLD/T always performs its tabbing function in EDT, unless you have redefined the key sequence.

CTRL/U (Control U) Function

description

Pressing CTRL/U deletes everything from the character to the left of the cursor to the beginning of the line. If the cursor is in the middle or at the end of the line and CTRL/U is pressed, EDT deletes the characters between the cursor and the beginning of that line. If the cursor is at the beginning of a line when CTRL/U is pressed, the line above the cursor is deleted. Text deleted by CTRL/U is stored in the delete line buffer. Use UND L to insert or restore the deleted text.

CTRL/U can be used to cancel a COMMAND, FIND, or CTRL/K operation. For example, if you have pressed GOLD/COMMAND and have started to enter a line-mode command, you can press CTRL/U to return the cursor to the text. If you have pressed GOLD/FIND and have started to enter a search string, you can also press CTRL/U to return the cursor to the text. The string in the search buffer remains the same as it was before you pressed GOLD/FIND. Similarly, if you are in the process of creating a key definition with CTRL/K, you can press CTRL/U to cancel the definition process.

CTRL/X always performs the same function as CTRL/U, regardless of the definition assigned to CTRL/X.

CTRL/W (Control W) Function

description

Pressing CTRL/W or GOLD/W refreshes the screen display. This function has no effect on the text you are editing; it simply clears and redraws the screen, eliminating any extraneous characters or messages that have appeared on the screen but are not part of the current text you are editing. Note that CTRL/W performs the same function as CTRL/R in keypad mode.

CTRL/Z (Control Z) Function

description

Pressing CTRL/Z shifts EDT from keypad mode to line mode. After you have pressed CTRL/Z, the line mode asterisk prompt (*) appears indicating that EDT is ready to accept line-mode commands. To resume keypad editing, use the line-mode CHANGE command.

CUT Function

description

Pressing CUT removes the active select range from the current buffer and stores it in the PASTE buffer. You can use CUT to delete large or small sections of text. When you use CUT in conjunction with the PASTE function, you can move or copy text from one place in the current buffer to another place in that buffer.

When you use CUT to delete only part of a line, EDT adds a line terminator at the end of the text being stored in the PASTE buffer. The line terminator is necessary because EDT cannot store partial lines in the PASTE buffer. When you use the PASTE function, EDT removes the added line terminator. Thus, when you insert the text, you do not have an extra line terminator.

The steps for moving and copying text are described under the keypad PASTE function.

DEL C (Delete Character) Function

description

Pressing DEL C (delete character) deletes the character on which the cursor is positioned. The cursor stays in the same position, but the remaining characters on the line shift one position to the left.

The deleted character is stored in the delete character buffer. Only one character at a time can occupy that buffer. Each time you delete a character with DEL C or with the DELETE function, the contents of the delete character buffer are overwritten. Remember that the delete character buffer is inaccessible to you and that its name does not appear in the list displayed by the SHOW BUFFER command.

Use UND C to restore or insert the contents of the delete character buffer into your text.

DEL C deletes the character the cursor is on; the DELETE function always deletes the character to the left of the cursor. (D-C. is the nokeypad definition for DELETE.)

DEL EOL (Delete to End of Line) Function

description

Pressing DEL EOL (delete to end of line) deletes everything on a line from the character the cursor is on up to, but not including, the line terminator. The cursor remains in the same position as it was before DEL EOL was pressed. If the cursor is on a line terminator, DEL EOL deletes that line terminator and all the text up to the next line terminator.

The characters deleted from the line are placed in the delete line buffer. Each time DEL EOL, DEL L, or CTRL/U is used, the contents of that buffer are overwritten. Use UND L to restore or insert the contents of the buffer in your text.

When you use DEL EOL, EDT deletes the characters up to the line terminator to the right of the cursor. DEL L deletes those same characters, but also deletes the line terminator and positions the cursor on the first character of the next line. CTRL/U deletes the text from the character to the left of the cursor to the beginning of the line.

DELETE Function

description

Pressing the DELETE key deletes the character to the left of the cursor. If the cursor is at the beginning of a line, pressing DELETE deletes the preceding line terminator.

When a character is deleted using the DELETE key, that character is placed in the delete character buffer. The contents of the buffer are overwritten each time a character is deleted either by the DELETE function or by DEL C. Use UND C to restore or insert the contents of the delete character buffer into the text you are editing.

Use the DELETE key to edit the text you enter in response to EDT prompts such as **Search for:** or **Command:**. These deleted characters are not stored in the delete character buffer.

The difference between DELETE and DEL C is that DEL C deletes the character that the cursor is on; DELETE deletes the character to the left of the cursor.

DEL L (Delete Line) Function

description

Pressing DEL L (delete line) deletes everything on a line starting with the character that the cursor is on up to and including the line terminator. The cursor position remains unchanged on the screen.

If the cursor is on the first character of the line when you press DEL L, the entire line is deleted and the cursor is positioned on the first character of the following line.

The characters deleted by DEL L, DEL EOL, or CTRL/U are stored in the delete line buffer. Each time a line or piece of line is deleted, the contents of the delete line buffer are overwritten. Use UND L to restore or insert the contents of the delete line buffer into the text you are editing.

DEL L always deletes the line terminator to the right of the cursor; DEL EOL deletes only the characters up to that line terminator. CTRL/U deletes the text from the character to the left of the cursor to the beginning of the line.

DEL W (Delete Word) Function

description

Pressing DEL W (delete word) deletes words or parts of words. When the cursor is at the beginning of the word, the entire word and the space following it are deleted. If the cursor is in the middle of the word, only the character that the cursor is on and those characters to the right of the cursor, up to and including the spaces that come after, are deleted. The characters to the left of the cursor in that word remain in the text. If the word being deleted is at the end of a line, all characters up to, but not including, the line terminator are deleted.

The characters deleted by DEL W and LINEFEED (F13 on LK201 keyboards) are stored in the delete-word buffer. Each time DEL W or LINEFEED is used, the contents of the delete-word buffer are overwritten. Use UND W to restore or insert the contents of the delete-word buffer into the text you are editing.

DEL W always deletes the cursor character and the remaining characters in the word to the right of the cursor. LINEFEED deletes the word or part of the word to the left of the cursor.

DO Function (LK201 only)

description

Pressing DO processes searches and line editing commands in keypad mode. Although DO has the same definition as ENTER, you cannot use the DO key to enter a key definition with CTRL/K.

When you receive a prompt from EDT in keypad mode, you can use DO to send EDT the information you enter in response to the prompt. The two preset EDT functions that have prompts are COMMAND and FIND.

To use COMMAND, press the GOLD and COMMAND keys. When EDT displays the *Command:* prompt, enter the line mode command. Then press DO to send the command to EDT for processing.

To use FIND, press either the LK201 FIND key or the GOLD and FIND keys on the numeric keypad. When EDT displays the *Search for:* prompt, enter the search string. Then press DO to send the string to EDT so it can perform the search.

DOWN Arrow

description

Pressing the DOWN arrow key moves the cursor down one line toward the bottom of the buffer regardless of EDT's direction.

When you use the DOWN arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of that line. If you continue to use the DOWN arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for the DOWN arrow and resets it the next time you use the function.

ENTER Function

description

Pressing ENTER processes searches, line editing commands, and key definitions in keypad mode. EDT generally uses the ENTER function to process keypad editing functions.

When you receive a prompt from EDT in keypad mode, use ENTER to send EDT the information you type in response to the prompt. The two preset EDT functions that have prompts are COMMAND and FIND.

To use COMMAND, press the GOLD and COMMAND keys. When EDT displays the *Command:* prompt, enter the line-mode command. Then press ENTER to send the command to EDT for processing.

To use FIND, press the GOLD and FIND keys. When EDT displays the *Search for:* prompt, enter the search string. Then press ENTER to send the string to EDT so it can perform the search.

You are asked to press the ENTER key when you complete a keypad definition using CTRL/K in keypad mode. When the message "Now enter the definition terminated by ENTER" appears, type the definition and then press the ENTER key.

EOL (End of Line) Function

description

Pressing EOL (end of line) moves the cursor to the end of the current line if the direction is forward. If the current direction is backward, the cursor moves to the end of the previous line. If the cursor is already at the end of a line, EOL moves it to the end of the next or previous line, depending on the current direction. Use BACKSPACE (F12 on LK201 keyboards) to move the cursor to the beginning of a line.

FILL Function (VT100)

description

Pressing FILL takes a select range of lines and reorganizes the text so that the maximum number of whole words can fit within the current line width. The default line width for EDT is the terminal width that the operating system passes to EDT. Use the line-mode SHOW SCREEN command to find the current screen and line width. The valid screen width values for screen-mode editing are 80 and 132. (The 132 screen width is only valid for VT100-series terminals with AVO — advanced

video option.) If your screen width is set to 80, EDT will fill lines to column 79; if your screen width is 132, EDT will fill lines to column 131.

If you want to use a line length other than 80 or 132 for filling text, you must use the line-mode SET WRAP command. The SET WRAP command also affects the line length that EDT uses for inserting text in keypad mode. EDT uses the SET SCREEN value to determine the line length for filling text only if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses the wrap value, regardless of the SET SCREEN width. You can use the SHOW WRAP command to find out the current wrap value or setting.

The filling process considers a blank line to be a break between paragraphs. Even if there are spaces on the blank line, EDT fills the text up to the blank line and then resumes filling with the next line that contains text.

The nokeypad definition for FILL on VT100-series terminals and for CTRL/F on VT52 terminals is FILLSR.

The FILL function is available on all VT100-series terminals. You must use CTRL/F on VT52 terminals to perform the FILL function.

FIND Function

description

Pressing FIND sets up a search procedure. When you press GOLD and then FIND, EDT displays the prompt *Search for:* at the bottom of the screen. Enter the string you want to locate. Then press ENTER to process the search in the current direction.

After you have typed in your search string, you can press ADVANCE instead of ENTER to search toward the end of the buffer, or you can press BACKUP to search backward toward the top. The direction you use to process FIND becomes EDT's current direction.

EDT can perform searches in several ways. The defaults are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search in the portion of the buffer between the cursor position and the beginning or end of the buffer, depending on the direction of the search. Use the SET SEARCH command to change the way EDT performs searches. The SHOW SEARCH command tells you which search parameters are currently in effect.

FNDNXT (Find Next) Function

description

After a search string has been established by FIND, you can use FNDNXT (find next) to locate the next occurrence of that string. The direction for FNDNXT is always the current EDT direction. You can change directions without affecting the search string.

The search string established by FIND remains in effect until you use FIND again or use some other EDT function that overwrites the contents of the search buffer.

EDT can perform searches in several ways. The defaults are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search on the portion of the buffer between the cursor position and the beginning or end of the buffer, depending on the direction of the search. Use the SET SEARCH command to change the way EDT performs searches. The SHOW SEARCH command tells you which search parameters are currently in effect.

GOLD Function

description

Pressing GOLD together with other keypad and keyboard keys performs various editing functions. GOLD is like the SHIFT key in that it does nothing by itself.

When used with a keypad key, GOLD causes EDT to perform that key's alternate function. For example, to use the COMMAND function, you must first press GOLD and then the 7 key on the keypad. If you do not press GOLD, EDT performs the PAGE function. Using EDT's key definition facility, you can redefine any GOLD/keypad sequence to perform a different function during your EDT session.

The define key feature allows you to designate a GOLD/keyboard key sequence to perform a keypad editing function for the duration of your editing session. You can also use GOLD in combination with a CTRL/character sequence and with the DELETE key to define new keypad-mode functions. GOLD/FUNCTION key sequences can be defined on terminals that have LK201 keyboards.

GOLD can be used with keyboard number keys to designate the number of times for EDT to repeat a keypad editing function. First press GOLD, next the keyboard number keys, and then the keypad function keys that you want EDT to repeat. When EDT's direction is set to forward, you can use GOLD followed by a minus sign (-) to change EDT's direction to backward temporarily. This feature allows you to process an EDT function in the opposite direction, without having to reset EDT's direction. For example, you can use GOLD/-2 with WORD to have the cursor back up two words without changing EDT's direction. The maximum number of times you can repeat a function with the GOLD/repeat feature is 32,767.

When you use the SPECINS function, you first press GOLD, then the keyboard digits for the decimal equivalent value of the character you want to insert. Then press GOLD again — this time to access the alternate function on the keypad function key — and finally the SPECINS key.

GOLD is the nokeypad definition for GOLD. Note that there is no period at the end of the definition because GOLD is not a nokeypad command. You must use the line-mode DEFINE KEY command to redefine a key that has GOLD as its definition.

HELP Function description

Pressing HELP provides information on EDT's preset keypad and control functions. Pressing HELP puts you in touch with EDT's HELP facility; it has no effect on your editing session. When you exit from HELP, the screen is redrawn exactly as it was before you pressed HELP and the cursor is in the same position as before.

When you press HELP, EDT displays a diagram of the keypad functions and a list of preset control-key functions. For help on a particular keypad function key, press the appropriate keypad key. For information on a GOLD/keypad sequence, press only the keypad key. Information for both the primary and alternate functions of that key will be displayed. For information on a control key sequence, press both the CTRL and keyboard keys after you are in the keypad HELP facility. For help on a GOLD/keyboard key sequence, press only the keyboard key; do not press GOLD.

To exit from HELP, press the spacebar.

If you have access to more than one HELP file, use the SET HELP command to change HELP files. The SHOW HELP command displays the name of the HELP file that is currently available for your editing session.

To define another key to perform the HELP function, use the nokeypad HELP command.

LEFT Arrow

description

Pressing the LEFT arrow moves the cursor one character to the left, regardless of EDT's direction.

If the cursor is at the first character position of a line, pressing LEFT arrow moves the cursor to the line terminator of the previous line.

LINE Function

description

Pressing LINE moves the cursor to the beginning of the next line if the direction is forward or to the beginning of the current line if the direction is backward. If the cursor is at the beginning of a line and the direction is backward, the cursor moves to the beginning of the previous line.

LINEFEED Function

description

Pressing LINEFEED deletes the word or characters in a word to the left of the cursor up to the beginning of the previous word. It is similar to DEL W, which deletes the word or characters in a word to the right of the cursor up to the beginning of the next word.

If the cursor is on a space when LINEFEED is pressed, the word preceding the space is deleted, usually leaving two spaces in a row. If the cursor is at the end or in the middle of a word, all characters in that word to the left of the cursor are deleted. The letter that the cursor is on remains in the text.

When the cursor is at the beginning of a word, the preceding word and space are deleted by LINEFEED. If the cursor is at the beginning of a line, LINEFEED deletes the preceding line terminator.

All characters deleted by LINEFEED are stored in the delete word buffer. Each time DEL W or LINEFEED is used, the contents of the delete word buffer are overwritten. Use UND W to insert or restore the contents of the delete word buffer in your text.

The LINEFEED key and CTRL/J always have the same preset function in EDT. When you redefine the LINEFEED key, you redefine CTRL/J (except for terminals with LK201 keyboards when they are operating in VT200 mode). To redefine the LINEFEED key using the line-mode DEFINE KEY command, enter DEFINE KEY CONTROL J. To find out what the

definition of the LINEFEED key is, enter SHOW KEY CONTROL J. For terminals with LK201 keyboards, use DEFINE KEY FUNCTION 25 and SHOW KEY FUNCTION 25 for the F13 key.

OPEN LINE Function

description

Pressing OPEN LINE inserts a line terminator in the text you are editing at the current cursor position and makes the line terminator the new cursor character. If the cursor is initially at the beginning of a line, the text on that line is moved down so that the cursor is on the blank line.

When the cursor is in the middle of a line, the text to the right of the cursor and the cursor character itself move to a new line. The cursor is now on the line terminator that OPEN LINE inserts. When the cursor is at the end of a line, a line terminator is added, creating a blank line below the current line.

RETURN and CTRL/M also insert line terminators in your text. However, neither of these functions moves the cursor to the inserted line terminator. The cursor remains on the same character.

PAGE Function

description

Pressing PAGE moves the cursor to a position at the right of the next page marker in your text. The cursor is always located after the page marker, but the direction that EDT moves to find the page marker depends on the current direction. In order to use PAGE, the text you are editing must have PAGE boundary markers. The default page marker is the form feed character (CTRL/L, decimal value 12, displayed by EDT as <FF>).

If you have no page markers in your buffer, the PAGE entity is the same as the whole buffer. When EDT's direction is forward, PAGE moves the cursor to the end of buffer ([EOB]) mark. If the current direction is backward, PAGE moves the cursor to the beginning of the buffer.

You can use the SET ENTITY PAGE command to define any string of characters as the page marker for the duration of your editing session. The marker can be either a single character that you insert in the text, such as an exclamation point (!), or a series of characters, such as a RUNOFF header level (.HL1).

If you are using the default page marker, you can use SET TEXT PAGE to have EDT display some other text in place of the <FF> page marker for the duration of your EDT session.

PASTE Function

description

Pressing **PASTE** in conjunction with **CUT** or **APPEND** copies or moves text within a buffer. **PASTE** copies the text currently residing in the **PASTE** buffer into the current buffer. The **PASTE** buffer contents are inserted to the left of the cursor regardless of EDT's current direction. **PASTE** has no effect on the contents of the **PASTE** buffer.

To move text from one place in your buffer to another, you need to use **SELECT**, **CUT**, and **PASTE** in the following order:

1. Use **SELECT** to create a select range of the text you want to move.
2. Press **CUT** to delete the text from the current buffer and put it into the **PASTE** buffer.
3. Move the cursor to the location where you want to insert the deleted text.
4. Press **PASTE** to have EDT copy the **PASTE** buffer text into your current buffer to the left of the cursor.

You can use **SELECT**, **CUT**, and **PASTE** to copy text that exists in one place in your buffer to a second location. Follow the same procedure as for moving text, but add an additional step between the second and third steps:

- 2a. Press **PASTE** to have EDT restore the deleted text in its original location.

Each time you use **CUT**, EDT overwrites the contents of the **PASTE** buffer. If you want to add more text to the buffer before you insert it in the new location, you can use **APPEND**. **APPEND** deletes the select range text from its current location and adds it to the end of the **PASTE** buffer. When you press **PASTE**, both the text you deleted with **CUT** and the text you deleted with **APPEND** are inserted to the left of the cursor.

It is possible to edit the **PASTE** buffer. By entering the line-mode **FIND=buffer** command, you can enter the **PASTE** buffer, make your changes, and then return to your original buffer. Now, when you use **PASTE**, the revised buffer contents are inserted at the cursor location.

When you use **CUT** to put part of a line into a buffer, EDT adds a line terminator at the end of the text since EDT buffers cannot hold partial lines. **PASTE** removes the added line terminator so that when you insert the text, you do not have an extra line terminator.

You can use the line-mode **FIND** command to move from one buffer to another during your EDT session. Then you can use **PASTE** to put the contents of the **PASTE** buffer in that buffer.

REPLACE Function

description

Pressing **REPLACE** deletes the text in the select range and replaces it with the contents of the **PASTE** buffer. **REPLACE** enables you to delete different blocks of text and replace them all with the same text. **EDT** stores the deleted text in a buffer called **DELETE**. If the buffer does not exist, **EDT** creates it. If you have created a buffer called **DELETE**, **EDT** overwrites the text you had in that buffer with the newly deleted text. Each time you use **REPLACE**, **EDT** overwrites the text in the **DELETE** buffer. The **DELETE** buffer can be entered and edited and its name appears on the **SHOW BUFFER** list.

You can use **CUT** to put the replacement text into the **PASTE** buffer, or you can move to the **PASTE** buffer with the line-mode **FIND** command and insert the text directly there.

RESET Function

description

Pressing **RESET** changes several conditions of your editing session:

- Cancels an active select range
- Sets **EDT**'s current direction to **ADVANCE**
- Sets **EDT** to the default **DMOV** state

RESET also has a special function within the **CTRL/K** key definition facility. Namely, you can use **RESET** to delete the text on the definition line if you want to start your definition over again.

RESET is the nokeypad definition for **RESET**. Note that there is no period at the end of the definition. This is because **RESET** is not a nokeypad command.

RETURN Function

description

Pressing RETURN adds a line terminator to the text you are editing. The new line terminator is inserted to the left of the current cursor position. The cursor remains on the same character where it was before you pressed RETURN. If the cursor is at the beginning of the line, a blank line is created above the current cursor line.

When the cursor is in the middle of a line, RETURN moves the cursor character and all the text to the right of the cursor to a new line. When the cursor is at the end of a line, RETURN adds a line terminator, creating a blank line below the current line. The cursor is then positioned at the beginning of the new blank line. OPEN LINE also inserts a line terminator in your text, but it positions the cursor on the new line terminator.

You can redefine the RETURN key, although this is not recommended. When you redefine the RETURN key, you also redefine CTRL/M. To find out the definition of the RETURN key, enter SHOW KEY CONTROL M.

RIGHT Arrow

description

Pressing the RIGHT arrow moves the cursor one character to the right, regardless of EDT's direction.

If the cursor is on a line terminator, RIGHT arrow moves the cursor to the first character on the next line.

SECT (Section) Function

description

Pressing SECT (section) moves the cursor one section — 16 lines — toward the end or beginning of the buffer, depending on EDT's current direction. The cursor is always placed at the beginning of the new current line regardless of its previous position.

SELECT Function

description

Pressing SELECT sets up a select range for use with keypad functions such as APPEND, CHNGCASE, CUT, FILL, REPLACE, SUBS, and CTRL/T. Start with the cursor at one end of the text you want selected. Next press SELECT to mark that position as the beginning of the select range. Then, using the arrow keys and/or function keys that move the cursor, mark the other end of the text being selected. Now you are ready to press a function key that uses a select range.

The RESET function cancels the select range. If you have included more text than you wanted in the select range, you can move the cursor back toward the position initially marked by SELECT, using arrow keys and cursor moving functions, thus reducing the size of the range. Adjusting select ranges on VT100 terminals is easy because EDT shows the text in reverse video. On VT52 terminals, you might find it easier to use RESET to cancel the select range and then start over.

You can use a select range with line-mode commands by entering the line mode range specifier SELECT. However, line mode requires that the select range contain only whole lines.

SPECINS (Special Insert) Function

description

Pressing SPECINS (special insert) enables you to insert any character from the DEC Multinational Character Set into your text, using the character's decimal equivalent value. You can use SPECINS to enter ASCII control characters, such as CTRL/L, or letters with diacritical marks such as the umlaut (¨) or acute accent (´).

To use SPECINS, first press GOLD. Next, type the decimal equivalent number for the character you want to insert. Use the main keyboard digits to type this number; do not use the keypad number keys. EDT displays the number you typed at the bottom of the screen. You can use the DELETE key to edit the number. Now press GOLD again, this time followed by the SPECINS key. The EDT symbol for the character you inserted appears on the screen to the left of the cursor.

Each time you want to enter a special character, you must repeat the entire procedure. You cannot enter two characters with one SPECINS function, nor can you use the GOLD repeat feature to enter the same character several times in one location.

EDT-24 EDT Keypad Commands **SPECINS (Special Insert) Function**

SPECINS cannot be used if SET NOREPEAT is in effect for your editing session.

The maximum decimal character value for SPECINS is 255.

string specifier

description

The string specifier is generally used either to locate characters in a buffer or to replace the located characters. When a string specifier is used to locate a piece of text, it is referred to as the search string. All three editing modes use search strings. Line mode and nokeypad mode use substitute strings.

Whenever you enter a search string, EDT overwrites the contents of the search buffer. (Similarly, when you enter a substitute string, EDT overwrites the contents of the substitute buffer.)

The search and substitute buffers cannot be edited or entered. Their names never appear in the SHOW BUFFER list. You can use the nokeypad CLSS (clear search string) command to delete the contents of the search buffer.

EDT has a number of ways to perform searches. See the discussion of the SET SEARCH command for information about the EDT search parameters.

SUBS (Substitute) Function

description

Pressing SUBS (substitute) replaces the current search string with the contents of the PASTE buffer. In order to use SUBS, you must first put the string you want to replace in the search buffer and the new text in the PASTE buffer. All searches and substitutions are made in the current direction.

CUTSR=DELETE PASTEKS"" is the nokeypad definition for SUBS. This means that the select range — in most cases the current search string — is deleted from the current buffer and placed in a buffer named DELETE. The contents of the PASTE buffer are inserted in the text, and the cursor is placed on the last character of the inserted text if EDT's direction is forward. (If EDT's direction is backward, the cursor is positioned on the first character of the inserted text.) Finally, EDT moves to the next occurrence of the current search string.

Using SUBS involves four steps:

1. Put the search string in the search buffer — The easiest way to load the search buffer is with the FIND function. You can also use any line mode or nokeypad command that involves a search string. Remember, the search buffer cannot be entered or edited.
2. Put the replacement text in the PASTE buffer — There are two ways to load the PASTE buffer.
 - You can type the replacement text in your current buffer, make it a select range, and then use CUT to transfer it to the PASTE buffer.
 - Since you can enter the PASTE buffer and edit its contents, you can use the line-mode FIND command to move to the PASTE buffer and then insert the replacement text there. Use the FIND command to return to the buffer you are editing.
3. Locate the search string — If you reverse the order of steps 1 and 2, the cursor will already be at the search string. Otherwise, you must be sure that the cursor is positioned on the first character of the search string before you press SUBS. This is because SUBS performs the substitution first and then moves to the next occurrence of the search string. The order allows you to decide whether you want to perform the substitution on that instance of the search string or go on to the next one. (Use FNDNXT to skip the substitution on the current search string match and advance to the next occurrence.)
4. Press GOLD and then SUBS — If the cursor is not on the search string, EDT prints the message “No select range active”. If there is no other search string match in the remaining portion of your buffer, EDT prints the message “String was not found”.

SUBS is the only substitute function that can use a line terminator in the replacement text.

TAB Function

description

Pressing TAB (CTRL/I) moves text to the right. The number of column positions that the text moves depends on the cursor position; the value set by the SET TAB command, if one is in effect; and the indentation level count, if one is in effect. (SET NOTAB is the default.)

EDT-26 EDT Keypad Commands

TAB Function

EDT has preset tab stops every eight characters, regardless of how your terminal is set. If no SET TAB command has been entered, pressing TAB moves the cursor character, as well as all the characters on the current line to the right of the cursor, to the nearest preset tab position. Text is always moved to the right, regardless of EDT's current direction.

When a SET TAB value is in effect, TAB moves the entire line to the column designated by the SET TAB value only if the cursor is located in column 1. If the cursor is located anywhere else on the line, TAB moves the text to the nearest preset EDT tab stop.

If a tab indentation level count is in effect and the cursor is located in column 1 of the line, TAB moves the text to the indentation level position. The indentation level count is determined by three functions: (1) CTRL/A, which can be used to compute the indentation level count, (2) CTRL/D, which decrements the count, and (3) CTRL/E, which increments the count. Use the SHOW TAB command to find out the current SET TAB value and the indentation level count.

CTRL/T indents whole lines of text by the SET TAB value.

The TAB key and CTRL/I always have identical functions in EDT. When you redefine the TAB key, you redefine CTRL/I. To redefine the TAB key using the line-mode DEFINE KEY command, enter DEFINE KEY CONTROL I. When you want to find out the definition of the TAB key, enter SHOW KEY CONTROL I.

TOP Function

description

Pressing TOP moves the cursor to the first character position at the beginning of the buffer. TOP has no effect on EDT's current direction.

UND C (Undelete Character) Function

description

Pressing UND C (undelete character) inserts the current contents of the delete character buffer into your text to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is located on the inserted character if you used DEL C to delete the character. If you used DELETE to delete the character, the cursor is located to the right of the inserted character.

EDT Keypad Commands EDT-27

UND C (Undelete Character) Function

The keypad functions DEL C and DELETE both place the character they delete in the delete character buffer. Each time you use DEL C or DELETE, the contents of the delete character buffer are overwritten. The buffer contains only the most recently deleted character. When you use a repeat count with DEL C or DELETE, only the last character deleted is in the delete character buffer. If no character has been deleted during the current EDT session, UND C inserts nothing. Note that, if you use the DELETE key to delete characters in a command line or prompt line, these characters are *not* stored in the delete character buffer and will not affect the character inserted by UND C.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in your text and you delete it. When you undelete this character, EDT changes the <CR> character into a line terminator and inserts the line terminator in your text.

UND L (Undelete Line) Function

description

Pressing UND L (undelete line) inserts the current contents of the delete line buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to a new line below the current line if the buffer contents end with a line terminator. Otherwise, text just moves to the right. The cursor is located on the first character of the inserted text if you used DEL L or DEL EOL to delete the text. If you used CTRL/U to delete the text, the cursor is located to the right of the inserted text.

The delete line buffer is loaded by using DEL L, DEL EOL, or CTRL/U. Each time one of these three functions is used, the contents of the delete line buffer are overwritten. The current contents of the buffer are the most recently deleted line or line portion. When you use a repeat count with a delete line function, only the last line or line portion that was deleted is in the delete line buffer. If no line has been deleted in your EDT session, UND L inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer.

UND W (Undelete Word) Function

description

Pressing UND W (undelete word) inserts the current contents of the delete word buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is located on the first character of the inserted word or word portion if you used DEL W to make the deletion. If you used LINEFEED, the cursor is located to the right of the inserted word or word portion.

The delete word buffer is loaded by using DEL W or LINEFEED (CTRL/J, F13 — LK201). Each time you use DEL W or LINEFEED, the contents of the delete word buffer are overwritten. The current contents of the buffer are the most recently deleted word or word portion. When you use a repeat count with a delete word function, only the last word or word portion deleted is in the delete word buffer. If no word has been deleted in your EDT session, UND W inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer.

UP Arrow

description

Pressing the UP arrow key moves the cursor up one line toward the top of the buffer regardless of EDT's direction.

When you use the UP arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of the short line. If you continue to use the UP arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for the UP arrow and resets it the next time you use the function.

WORD Function

description

Pressing **WORD** moves the cursor to the beginning of the next word in the current direction (forward or backward, depending on whether **ADVANCE** or **BACKUP** is in effect).

An EDT word is any group of characters bounded by a space, horizontal tab, line feed <LF>, vertical tab <VT>, form feed <FF>, or carriage return <CR>. You can establish different word boundaries with the line-mode **SET ENTITY WORD** command. Use the line-mode **SHOW ENTITY WORD** command to find the current boundary markers for the word entity.

The **SET WORD [NO]DELIMITER** command affects how EDT interprets word boundaries. With **SET WORD DELIMITER** (the default) in effect, EDT considers all word boundaries, except the space, as words themselves.

EVE Commands

This section describes each EVE command. The command descriptions are given in alphabetical order. Most of the information in this section is available by using EVE online help, so that even when this manual is not at hand, you can find out about EVE commands. EVE online help includes topics for each EVE command, each EDT keypad or WPS keypad key, and other features.

If there are default key bindings for a command, the command description includes a list of the keys defined on VT300-, VT200-, and VT100-series terminals; control keys are defined the same on all three series of terminals.

In some cases, there is a GOLD key combination for a command. However, EVE does *not* have a default GOLD key. You set the GOLD key with the SET GOLD KEY, SET KEYPAD EDT, or SET KEYPAD WPS command. This also enables several GOLD key combinations, such as GOLD-↓ for BOTTOM. (See Table EVE-1.) Note that some GOLD key combinations require a VT300- or VT200-series terminal (for example, GOLD-HELP).

@

format

@ *init-filespec*

description

Executes the initialization file you specify. This lets you execute a series of commands at the same time, such as setting the left and right margins or defining several keys for particular kinds of editing. For more information about initialization files, see Section 6.5.9.

parameter

init-filespec

The initialization file you want to execute. The default file type is EVE. You can use logical names in the file specification, but cannot use wildcards. For example, you can use SYS\$LOGIN or another logical name to specify the device or directory for the initialization file. You can use several initialization files during an editing session, but execute only one at a time. If you do not specify a file, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

EVE-2 EVE Commands

@

example

The following command executes an initialization file called MYINIT.EVE in your top-level (or login) directory:

```
Command: @ sys$login:myinit  
Executing commands in initialization file: DISK$1:[GEOFF]MYINIT.EVE;1
```

ATTACH

format

ATTACH [*process-name*]

description

Suspends your editing session, without ending it, and attaches the terminal to another process or subprocess. The other process or subprocess must already exist; the ATTACH command does not create it. Using ATTACH and SPAWN commands, in EVE and at the DCL level or in other utilities such as MAIL, lets you keep an editing session active throughout your VMS session (or login)—effectively making EVE a “kept” editor. This makes it faster to resume editing, but uses more system resources. To find out the names of your processes and subprocesses, use the DCL command SHOW PROCESS/SUBPROCESS.

The ATTACH command is not supported if you invoke EVE using /DISPLAY=DECWINDOWS.

parameter

process-name

Optionally, the process or subprocess to which you want to attach the terminal. Process names are case sensitive, and must be from 1 to 15 alphanumeric characters. You cannot specify a process ID. If you do not specify a process or subprocess, EVE attaches the terminal to the parent process.

example

In the following example, the DCL command SPAWN creates a subprocess named SMITH_1, invoking EVE to edit a file called MEMO.TXT. While you are editing the MEMO.TXT buffer, the EVE command ATTACH returns control to process SMITH (the parent process). After you complete work at the DCL level, the DCL command ATTACH SMITH_1 resumes the editing session. Exiting from EVE terminates the subprocess.

```
$ SPAWN EDIT/TPU memo.txt
%DCL-S-SPAWNED, process SMITH_1 spawned
%DCL-S-ATTACHED, terminal now attached to process SMITH_1
.
. [ editing MEMO.TXT (subprocess SMITH_1) ]
.
Command: ATTACH
%DCL-S-RETURNED, control returned to parent process SMITH
$
.
. [ at DCL level (process SMITH) ]
.
$ ATTACH SMITH_1
.
. [ editing MEMO.TXT (subprocess SMITH_1) ]
.
```

BOTTOM

format

BOTTOM

VT300, VT200:

GOLD-↓

VT100:

GOLD-↓

description

Moves the cursor to the end of the current buffer unless it is already there. The bottom of the buffer is marked [End of file].

BUFFER

format

BUFFER *buffer-name*

EVE-4 EVE Commands

BUFFER

description

Puts a buffer you specify into the current EVE window. If the buffer exists, EVE returns the cursor to your last position in that buffer. If the buffer does not exist, EVE creates a new buffer and puts the cursor at the top of that buffer (upper left corner). To return to a buffer that you previously viewed, use the BUFFER command and specify the buffer name. Typically, a buffer name is the same as the file it contains—that is, the file specified when you invoked EVE or when you used the GET FILE, OPEN, or OPEN SELECTED command. For a list of the buffers you have created, use the SHOW BUFFERS command.

parameter

buffer-name

The buffer you want to edit or create. In returning to an existing buffer, you can abbreviate the buffer name; also, buffer names are not case sensitive. You cannot use wildcards (for example, an asterisk is treated as a character in the buffer name). If more than one name matches your request, EVE shows a list of the matching names so you can choose the one you want. If you do not specify a buffer, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command puts a buffer named MEMO.TXT into the current window, returning the cursor to your last position in that buffer, or creating a new buffer:

Command: BUFFER memo.txt

CAPITALIZE WORD

format

CAPITALIZE WORD

description

Capitalizes a single word, select range, or found range—making the first character uppercase (if it is a letter) and the other letters lowercase. With a select range or found range, CAPITALIZE WORD works on each word in the range, starting with the first character of the range. If the highlighted range ends in the middle of a word, the case change continues to the end of that word. A select range takes precedence over a found range. If there is no select range or found range, CAPITALIZE WORD works on the current word. If you are between words, it works on the next word on the line.

CENTER LINE

format

CENTER LINE

description

Centers the current line between the left and right margins of the buffer, by inserting spaces at the start of the line. If you are on a blank line, CENTER LINE inserts spaces to move the cursor to the center column between the left and right margins.

CHANGE DIRECTION

format

CHANGE DIRECTION

VT300, VT200:

F11

VT100:

PF3

description

Changes the direction of the current buffer from forward to reverse or conversely. The direction of the buffer is shown in the status line. It affects commands such as FIND, REPLACE, MOVE BY LINE, MOVE BY WORD, and MOVE BY PAGE. For buffers you create, the default direction is forward (right and down). Note that direction is a buffer-specific setting; you can have one buffer set to forward and another buffer set to reverse. For editing EVE command lines, the default direction is reverse. To change the direction when you are editing a command line, press a key defined as CHANGE DIRECTION. This direction remains in effect until you change it again—it does *not* revert to the previous direction after you finish typing a command. It is independent of the direction of your text buffers.

CHANGE MODE

format

CHANGE MODE

VT300, VT200:

F14
CTRL/A

VT100:

ENTER
CTRL/A

description

Changes the mode of the current buffer from insert to overstrike or conversely. The mode of the buffer is shown in the status line. It affects not only how text is entered, but also some EVE commands, such as DELETE, ERASE CHARACTER, and RESTORE CHARACTER. For buffers you create, the default mode is insert. Note that the mode is a buffer-specific setting; you can have one buffer set to insert and another buffer set to overstrike. For typing or editing command lines, the default mode matches your terminal setting (according to the DCL command SET TERMINAL). To change the mode when you are editing a command line, press CTRL/A or other key defined as CHANGE MODE. This mode remains in effect until you change it again—it does *not* revert to the previous mode after you finish typing a command. It is independent of the mode of your text buffers.

If you set the buffer to unmodifiable (for example, by using the command SET BUFFER READ_ONLY), then Unmodifiable appears in the status line, instead of Insert or Overstrike. To change the mode of an unmodifiable buffer, first use the command SET BUFFER MODIFIABLE.

COPY

format

COPY

description

Same as the STORE TEXT command—copies a select range or found range, without removing it, so you can insert the text elsewhere.

CUT

format

CUT

description

Same as the REMOVE command—removes a select range or found range, which you can then paste elsewhere.

DCL

format

DCL *dcl-command*

description

Executes the DCL command you specify, and puts the command and any output from it into the DCL buffer in a second EVE window. One window shows the buffer you are editing. The other window shows the DCL buffer. (If you are already using two or more windows, EVE uses the next or other window to show the DCL buffer. If you are using only one window, EVE splits the main window in two.) EVE creates a subprocess for executing the DCL command. When the command is completed, the cursor automatically returns to your last position in the buffer you are editing in the other window. The DCL window stays on the screen. You can edit the DCL buffer to move the output from the DCL command into another buffer. To delete the DCL window, typically you use the ONE WINDOW command.

parameter

dcl-command

The DCL command you want to execute, including any required parameters. If you do not specify a command, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command splits the EVE window (unless it is already split), and displays the DCL command DIRECTORY and its output (the directory listing) in the second window:

```
Command: DCL DIRECTORY *.txt
```

DEFINE KEY

format

DEFINE KEY [=key-name] eve-command

description

Defines a key to execute an EVE command or an EDT keypad or WPS keypad function you specify. You can type the key name (preceded by an equal sign) on the command line or let EVE prompt you to press the key you want to define. For information about key names and nondefinable keys, see Section 6.4.3. Generally, the DEFINE KEY command overrides any current definition of the specified key, whether EVE default, EDT keypad, WPS keypad, or your own. For example, if you define a key that is ordinarily defined by a keypad setting, such as EDT or WPS, your definition overrides the keypad definition. Use the UNDEFINE KEY command to restore the keypad definition of the key. Setting the EDT keypad or WPS keypad makes PF1 the GOLD key, overriding any definition you have given PF1—unless you set a different key as GOLD (with the SET GOLD KEY command).

The key definition remains in effect throughout the editing session or until you redefine or undefine the key. To save key definitions for future sessions, put the DEFINE KEY commands in your EVE initialization file or use the SAVE EXTENDED EVE command to create a section file. To show the definition of a key, use the SHOW KEY command. To cancel a key definition, use the UNDEFINE KEY command.

parameters

key-name

The key you want to define. You cannot abbreviate the key name. Note that the key name must be preceded by an equal sign to distinguish it from the command you are assigning to the key. For more information about EVE key names, see Section 6.4.3. If you do not type a key name on the command line, EVE prompts you to press the key you want to define. Pressing the RETURN key or CTRL/M at the prompt cancels the operation, because those keys cannot be redefined.

eve-command

The command you want to bind to the key, or the name of an EDT key or WPS key whose function you want to bind to the key. If you do not specify a command, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command defines F20 as the INCLUDE FILE command. Thereafter, when you press the key, EVE prompts you to type the name of the file to be included.

```
Command: DEFINE KEY include file  
Press the key that you want to define: F20
```

DELETE

format

DELETE

VT300, VT200:

<X

VT100:

DELETE

description

Erases the character left of the cursor, or replaces it with a space, depending on the mode of the buffer. In insert mode, the rest of the line moves left one character to close the space. In overstrike mode, the erased character is replaced by a space. At the start of a line, DELETE erases the carriage return for the previous line—regardless of the mode—causing the current line to move up. This is useful to erase blank lines to form paragraphs for FILL commands. To put back the character you erased, use the RESTORE CHARACTER command, which is also mode sensitive. If you enable pending delete and then select text, DELETE erases the selected text. To put back the erased text, use the RESTORE SELECTION command. For more information, see the description of the SET PENDING DELETE command.

DELETE BUFFER

format

DELETE BUFFER *buffer-name*

description

Deletes the buffer you specify by name. If the specified buffer is displayed in a window, EVE deletes the buffer, and then displays another buffer—usually the first buffer viewed in the editing session. If you specify a buffer that has been modified and is not empty, EVE asks you to confirm that you want to delete it. The following table shows the possible responses and the effect of each response. You need only type the first letter of the response (and press RETURN).

Response	Effects
DELETE_ONLY	Deletes the specified buffer.
WRITE_FIRST	Writes out the buffer to a file before deleting it. If there is no file specification for the buffer—that is, if you invoked EVE without specifying an input file or if you created the buffer with the BUFFER or NEW command—EVE prompts you for one, as with the WRITE FILE command.
QUIT	Cancels the operation—the buffer is <i>not</i> deleted. This is the default response: you can simply press RETURN or CTRL/Z.

You can also delete buffers by using REMOVE or CUT in the Buffer List buffer, without having to type the buffer name. See the description of the SHOW BUFFERS command.

parameter

buffer-name

The buffer you want to delete. The buffer name must match exactly—no wildcards or abbreviations. Buffer names are not case sensitive. Typically, a buffer name is the same as the file it contains (as specified when you invoked EVE or when you used the BUFFER, GET FILE, NEW, OPEN, or OPEN SELECTED command). If you do not specify a buffer, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command deletes a buffer named MEMO.TXT. In this case, because the buffer has been modified (and not yet written out), EVE prompts you to confirm the deletion.

```
Command: DELETE BUFFER memo.txt  
That's a modified buffer. Type Delete_only, Write_first, or Quit: D
```

DELETE WINDOW

format

DELETE WINDOW

description

Deletes the current window, if you are using more than one window in EVE. Deleting a window does not delete the buffer that was displayed in the window.

DO

format

DO

VT300, VT200:

DO
PF4

VT100:

PF4

description

Enters or terminates an EVE command that you type, as follows:

1. Press DO or PF4. The cursor moves to the command window (just below the status line), and the *Command:* prompt appears.
2. Type the EVE command you want to execute, such as CENTER LINE, GET FILE, or SET RIGHT MARGIN. You can abbreviate the command, usually using the first letters of the command. Also, you can use EVE keys to edit the command line.

EVE-12 EVE Commands

DO

3. Press RETURN or DO to terminate the command. EVE then executes the command or prompts you for more information.

To cancel the command, erase the command line (for example, by pressing CTRL/U).

Pressing DO twice repeats the last command you entered. If you press DO and then press RETURN without typing a command, no command is executed.

END OF LINE

format

END OF LINE

VT300, VT200:

CTRL/E
GOLD-→

VT100:

CTRL/E
GOLD-→

description

Moves the cursor to the end of the current line. If you are already at the end of the line, the cursor does not move.

ENLARGE WINDOW

format

ENLARGE WINDOW *integer*

description

Enlarges the current window by the number of lines you specify—if you are using more than one window in EVE. The lines are added to the bottom of the window, unless the window is the bottommost window.

parameter

integer

The number of screen lines you want to add to the current window. The maximum size of a window depends on the size and type of terminal you are using. The minimum size is one line of text, one line for the status line, and on DECwindows, one line for the horizontal scroll bar. If there

is not enough room on the screen to enlarge the window as specified, EVE enlarges it as much as possible. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following commands form two windows, and then enlarge the lower of the two windows by five lines:

```
Command: TWO WINDOWS  
Command: ENLARGE WINDOW 5
```

ERASE CHARACTER

format

ERASE CHARACTER

description

Erases the character that the cursor is on, or replaces it with a space, depending on the mode of the buffer. In insert mode, the rest of the line moves left one character to close up the space. In overstrike mode, the erased character is replaced by a space. If you are at the end of a line, ERASE CHARACTER erases only the carriage return for that line—regardless of the mode—causing the next line (if any) to move up. This is useful to erase blank lines to form paragraphs for FILL commands. To put back the character you erased, use the RESTORE CHARACTER command, which is also mode sensitive.

ERASE LINE

format

ERASE LINE

description

Erases the current line, starting with the character that the cursor is on. The next line (if any) moves up. If you are at the end of a line, ERASE LINE erases only the carriage return for that line; the next line (if any) moves up. This is useful to erase blank lines to form paragraphs for FILL commands. To reinsert the erased text, use the RESTORE LINE command.

ERASE PREVIOUS WORD

format

ERASE PREVIOUS WORD

description

Erases all of the previous word or all of the current word, depending on your cursor position. If you are between words or on the first character of a word, the previous word is erased (left of the cursor). In the middle of a word, all of that word is erased (same as the ERASE WORD command). If you are at the start of a line, ERASE PREVIOUS WORD erases only the carriage return for the previous line (if any), causing the current line to move up. This is useful to erase blank lines to form paragraphs for FILL commands. To reinsert the erased text, use the RESTORE WORD command.

Note that if you are editing an EVE command line, any keys defined as ERASE WORD work like ERASE PREVIOUS WORD. Thus, you can use CTRL/J for editing command lines much as at the DCL level.

ERASE START OF LINE

format

ERASE START OF LINE

VT300, VT200:

CTRL/U
CTRL/<X>

VT100:

CTRL/U
CTRL/DELETE

description

Erases the current line, starting with the character left of the cursor until the start of the line. If you are already at the start of a line, nothing is erased. To reinsert the erased text, use the RESTORE LINE command.

ERASE WORD

format

ERASE WORD

VT300, VT200:

F13
CTRL/J

VT100:

COMMA
CTRL/J
LINEFEED

description

Erases all of the current word or, if you are between words, erases all of the next word. Erasing a word also erases the trailing spaces and tabs. If you are at the end of a line, ERASE WORD erases only the carriage return for that line; the next line (if any) moves up. This is useful to erase blank lines to form paragraphs for FILL commands. To put back the erased text, use the RESTORE WORD command.

Note that if you are editing an EVE command line, any keys defined as ERASE WORD work like ERASE PREVIOUS WORD. Thus, you can use CTRL/J for editing command lines much as at the DCL level.

EXIT

format

EXIT

VT300, VT200:

F10
CTRL/Z

VT100:

CTRL/Z

description

Ends the editing session and, typically, produces a new file or a new version of an existing file. When you exit, EVE writes out (saves) the current buffer, unless you have made no edits or unless there are no changes since you previously wrote out the buffer during the session. If there is no file specification for the buffer—that is, if you invoked EVE without specifying an input file or if you created the buffer with the BUFFER or NEW command—EVE asks you for one. Simply pressing RETURN at the prompt discards the buffer and continues exiting. If you have modified other buffers (as in editing more than one file in the session), EVE asks if you want to write out those buffers. Respond YES or NO. If necessary, EVE prompts you for any output file specifications. If you have not modified any buffers, the EXIT and QUIT commands are the same—exiting does not produce a new file or new version of a file. The output file for a buffer is typically the same as its input file—that is, the file specified when you invoked EVE or when you used the GET FILE, OPEN, or OPEN SELECTED command. If you wrote out (saved) the buffer with the WRITE FILE or SAVE FILE AS command and specified an output file, that file specification is used for writing out the buffer on exiting.

EXTEND ALL

format

EXTEND ALL

description

Compiles all the VAXTPU procedures in the current buffer. (Same as the command EXTEND EVE * or EXTEND TPU *.) EXTEND commands do not execute procedures. To execute a compiled procedure, use the EVE command TPU followed by the name of the procedure. To save compiled procedures for future editing sessions, use the SAVE EXTENDED EVE command to create a section file.

EXTEND EVE

format

EXTEND EVE { *procedure-name* }
 *

description

Compiles one or more VAXTPU procedures to extend EVE (same as the EXTEND TPU command). You can specify the name of a procedure in the current buffer or use the asterisk wildcard (*) to specify all the procedures in the current buffer.

EXTEND commands do not execute procedures. To execute a compiled procedure, use the EVE command TPU followed by the name of the procedure. To save compiled procedures for future editing sessions, use the SAVE EXTENDED EVE command to create a section file.

parameters

procedure-name

The VAXTPU procedure you want to compile. The procedure must be in the current buffer. You can abbreviate the procedure name, but cannot use wildcards; the name is not case sensitive. If more than one name matches your request, EVE shows a list of the matching names so you can choose the one you want. If you do not specify a procedure, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

*

Asterisk wildcard symbol specifying that you want to compile all the procedures in the buffer. This is the same as using the EXTEND ALL command.

example

The following command compiles a procedure named USER_PROC:

```
Command: EXTEND EVE user_proc  
EVE extended by: USER_PROC
```

EXTEND THIS

format

EXTEND THIS

description

Compiles the VAXTPU procedure that the cursor is in. This is the same as using the EXTEND EVE command without having to type the procedure name. This is useful for compiling a procedure with a lengthy name or a name similar to other procedures, without having to type it exactly.

Steps:

1. Put the cursor anywhere in the procedure you want to compile (that is, anywhere between the PROCEDURE and ENDPROCEDURE statements).
2. Use the EXTEND THIS command.

EXTEND commands do not execute procedures. To execute a compiled procedure, use the EVE command TPU followed by the name of the procedure. To save compiled procedures for future editing sessions, use the SAVE EXTENDED EVE command to create a section file.

EXTEND TPU

format

EXTEND TPU { *procedure-name* }
 *

description

Same as the EXTEND EVE command—compiles one or more VAXTPU procedures to extend EVE.

FILL

format

FILL

description

Reformats (rewraps) a select range, found range, or the current paragraph, so that the maximum number of words fits on a line according to the margins of the buffer. Typically, you use FILL commands to rewrap text after making some change in the buffer, such as inserting new text or changing the margins.

Steps:

1. Optionally, use SELECT, FIND, or WILDCARD FIND to highlight the text you want to fill. (A select range takes precedence over a found range.) Or put the cursor anywhere in the paragraph you want to fill.
2. Use the FILL command. The highlighting, if any, is canceled. The cursor stays at the end of the range you fill or moves to the end of the paragraph you fill.

In EVE, a *paragraph* is bounded by any of the following:

- Blank line
- Bottom or top of the buffer
- Page break (form-feed character)
- DIGITAL Standard Runoff command (such as .BLANK)

FILL PARAGRAPH

format

FILL PARAGRAPH

description

Reformats (rewraps) the current paragraph so that the maximum number of words fits on a line according to the margins of the buffer.

FILL RANGE

format

FILL RANGE

description

Reformats (rewraps) a select range or found range, so that the maximum number of words fits on a line according to the margins of the buffer. Filling a range keeps blank lines and page breaks as paragraph boundaries, which is useful if you select several paragraphs or the entire buffer for reformatting.

FIND

format

FIND *search-string*

VT300, VT200:

FIND

VT100:

PF1

description

Searches the current buffer for the text string you specify (or for one already entered). If the string is found, EVE puts the cursor at the beginning of the string and highlights the found text. If the string is found only in the opposite direction, EVE asks if you want to change the direction of the search and go there. Press RETURN if you want to go there, or type NO and press RETURN to end the search. If the string is not found, the cursor does not move.

The found text is highlighted (video bold), with the cursor at the beginning of the string. If there is no select range, you can use a command such as COPY, FILL, REMOVE, or UPPERCASE WORD with the found range much the same as with a select range. If there is a select range, the operation works on the selected text, which may not include the found range. To cancel the highlighting, move the cursor off the found range or use the RESET command.

parameter

search-string

The text you want to find. Use all lowercase to find any occurrence of the string. Use mixed case or all uppercase to find an exact match. EVE is also sensitive to diacritical marks, such as accents, in the search string. If you do not specify a search string, EVE prompts you for one. Pressing the FIND key at the prompt without typing anything searches for the previous string, if any; pressing RETURN at the prompt without typing anything cancels the operation.

example

The following command searches for the word *digital*, finding any occurrence regardless of its case in the buffer:

Command: FIND digital

FIND NEXT

format

FIND NEXT

description

Searches the current buffer for another occurrence of a string already entered with the **FIND**, **FIND SELECTED**, **REPLACE**, or **WILDCARD FIND** command. If the string is found only in the opposite direction, **EVE** asks if you want to change the direction of the search and go there. Press **RETURN** if you want to go there, or type **NO** and press **RETURN** to end the search. If the string is not found, the cursor does not move.

If the string is found, **EVE** puts the cursor at the beginning of the string and highlights the found text (video bold). If there is no select range, you can use **COPY**, **FILL**, **REMOVE**, **UPPERCASE WORD**, or other commands that work on a range of text. (If there is a select range, the operation works on the selected text, which may not include the found range.)

FIND SELECTED

format

FIND SELECTED

description

Searches the current buffer for the text string you have selected, rather than for a typed string. This is particularly useful to find a lengthy mixed-case string (such as a book title or a person's name) without having to type it exactly.

Steps:

1. Select the text you want to find.

On **DEC**windows, the select range can be in **EVE** or in another **DEC**windows application running concurrently.

2. Use the **FIND SELECTED** command. The selection is canceled.

EVE searches the buffer first in the current direction and then in the opposite direction. If the string is found only in the opposite direction, **EVE** asks if you want to change the direction of the search and go there. Press **RETURN** if you want to go there, or type **NO** and press **RETURN** to end the search. If the string is not found, the cursor does not move.

EVE-22 EVE Commands

FIND SELECTED

If the string is found, EVE puts the cursor at the beginning of the string, highlights the found text (video bold), and cancels the select range. You can then use COPY, FILL, REMOVE, UPPERCASE WORD, or other commands that work on a range of text. To find another occurrence of the same string, use the FIND NEXT command or press FIND twice.

If there is a found range (highlighted found text), and no select range, FIND SELECTED is the same as FIND NEXT. If there are both a found range and a select range, FIND SELECTED uses the select range.

FORWARD

format

FORWARD

description

Sets the direction of the current buffer to forward (right and down). The direction of the buffer is shown in the status line. It affects commands like FIND and MOVE BY LINE and some EDT keypad and WPS keypad keys. For buffers you create, the default direction is forward. Note that direction is a buffer-specific setting; you can have one buffer set to forward and another buffer set to reverse. For editing EVE command lines, the default direction is reverse, independent of the direction of your text buffers.

GET FILE

format

GET FILE *filespec*

description

Puts the file you specify into the current EVE window, creating a new buffer if necessary (same as the OPEN command). This lets you edit another file in the same session. If the file exists, EVE copies it into a new buffer in the current window. If the file does not exist, EVE creates a new, empty buffer, using the file name and file type for the buffer name. If you specify a file you have already opened in the editing session—that is, a file for which there is already a buffer—EVE returns to your last location in the buffer for that file, if the buffer still exists.

parameter

filespec

The file you want to edit or create. You can use logical names and wildcards in the file specification. If more than one file matches your request, EVE shows a list of the matching files so you can choose the one you want. You can edit several files in an editing session, but can specify only one file at a time. If you do not specify a file, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command gets a file called MEMO.TXT, returning the cursor to your last position in the buffer or creating a new buffer:

Command: GET FILE memo.txt

GO TO

format

GO TO *marker-name*

description

Moves the cursor to the position you specify, as previously labeled with the MARK command. Using MARK and GO TO commands makes it easier to move through a large buffer or to move between buffers. To find out marker names, use the SHOW command.

parameter

marker-name

The marker you want to go to, as previously specified with the MARK command. You can abbreviate the marker name, but cannot use wildcards (for example, an asterisk is treated as a character in the marker name). Marker names are not case sensitive. If more than one name matches your request, EVE shows a list of the matching names so you can choose the one you want. If you do not specify a marker, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

EVE-24 EVE Commands

GO TO

example

The following commands mark the current position as INTRO SEC, and later move the cursor to that position:

```
Command: MARK intro sec
          .
          .
          .
Command: GO TO intro sec
Going to mark: intro sec
```

HELP

format

HELP [*topic-name*]

VT300, VT200:

HELP
GOLD-HELP

VT100:

PF2

description

Displays online help on EVE commands, keys, or other topics, including VAXTPU built-in procedures. By default, HELP (or PF2 on VT100-series terminals) is defined as HELP KEYPAD, which draws a keypad diagram, showing the mini keypad, keypad, or both, depending on which keys are defined. You can then press keys you want help on. To get a list of all key definitions, use the command HELP KEYS or, if the GOLD key is set, press GOLD-HELP. This lists all the currently defined keys, including control keys and GOLD key combinations, if any. You can then press keys you want help on.

parameter

topic-name

An EVE command or other topic on which you want help. You can abbreviate the topic name. (You cannot type the name of an EDT keypad or WPS keypad function.) If more than one name matches your request, EVE shows a list of the matching topics so you can choose the one you want. (For example, if you type HELP SET, EVE lists all the SET commands.) If you specify a question mark (?), or if you do not specify any topic, EVE displays the list of topics. Pressing RETURN at the prompt exits from HELP.

example

The following command displays the list of topics on which help is available. You can then type the name of a topic you want help on.

Command: HELP

INCLUDE FILE

format

INCLUDE FILE *filespec*

description

Copies the file you specify into the current buffer, inserting its contents before the current line. The text of the included file is inserted whether the mode of the buffer is insert or overstrike. The cursor remains on the current character after the text of the included file is inserted. Including a file does not change the name of the buffer or the output file associated with the buffer. If the file you are including contains tab characters, the tab stops of the current buffer apply. Including a file does not reformat (rewrap) the text. To reformat text according to the margins of the buffer, use FILL commands.

parameter

filespec

The file you want to include. You can use logical names and wildcards in the file specification. If more than one file matches your request, EVE shows a list of the matching files so you can choose the one you want. You can include several files in a buffer, but can include only one file at a time. If you do not specify a file, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command includes a file called MORE.DAT, copying it into the current buffer:

Command: INCLUDE FILE more.dat

INSERT HERE

INSERT HERE

format

INSERT HERE

VT300, VT200:

INSERT HERE

VT100:

KP9

description

Inserts at your current position what you previously copied or removed (same as the PASTE command). The text is inserted whether the mode of the buffer is insert or overstrike. Existing text is pushed to the right or down. Depending on the amount of text inserted and where you are on the line, your text may go past the right margin or even partly out of view. Use FILL commands, if necessary, to reformat (rewrap) your text.

Depending on your setting, the text is inserted either from the Insert Here buffer in EVE or from the DECwindows clipboard. The default setting is NOCLIPBOARD, which uses the Insert Here buffer. For more information, see the description of the SET CLIPBOARD command.

INSERT MODE

format

INSERT MODE

description

Sets the mode of the current buffer to insert, as opposed to overstrike. As you type, the new text is inserted at the current position, pushing the cursor and any existing text to the right or down. The mode of the buffer is shown in the status line. For editing text, the default mode is insert. Note that the mode is a buffer-specific setting; you can have one buffer set to insert and another buffer set to overstrike. For editing EVE command lines, the default mode matches your terminal setting (according to the DCL command SET TERMINAL), independent of the mode of your text buffers.

INSERT PAGE BREAK

format

INSERT PAGE BREAK

VT300, VT200:

CTRL/L

VT100:

CTRL/L

description

Inserts at your current position a “hard” page break—a form feed, appearing as a small F . EVE puts the form feed on a line by itself, as follows:

- If you are at the start of a line of text, EVE inserts a form feed and does a RETURN.
- At the start of a blank line, EVE inserts a form feed and moves the cursor to the start of the next line—without doing a RETURN.
- If you are *not* at the start of a line, EVE first does a RETURN to start a new line, inserts a form feed, and then does another RETURN.

To erase a page break, use the MOVE BY PAGE command to put the cursor on a page break, and then use the ERASE LINE command or a similar EDT keypad or WPS keypad key. To insert a “soft” page break for a 54-line page, use the PAGINATE command.

LEARN

format

LEARN

description

Learns a sequence of keystrokes and remembers them as a single key. The sequence can comprise commands, text, or both. (In some text editors and word processors, this is called a *macro*.)

EVE-28 EVE Commands

LEARN

Steps:

1. Use the LEARN command to begin recording your keystrokes.
2. EVE prompts you to enter the keystrokes you want learned. You can enter text, commands, or both, including pressing keys already defined.
3. To end or remember the learn sequence, press CTRL/R (defined as REMEMBER). Do *not* type the REMEMBER command.
4. EVE then prompts you to press the key to be defined. You can press any of the following keys. Do *not* press a key you have used in the sequence being remembered.
 - A function key such as PF4, KP7, or F20
 - A control key such as CTRL/N
 - A GOLD key combination, such as GOLD-KP7 or GOLD-A

To cancel the definition, press the RETURN key or CTRL/M, which cannot be redefined. For information about nondefinable keys, see Section 6.4.3.

LINE

format

LINE *integer* [*procedure-name*]

description

Moves the cursor to the start of a line you specify by number—either in the current buffer or within a specified VAXTPU procedure in the buffer. To find out the current line number and total number of lines in the buffer, use the WHAT LINE command.

parameters

integer

The number of the line you want to go to. If you specify a number greater than the total number of lines in the buffer, EVE moves the cursor to the end of the buffer. If you do not specify a line number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

procedure-name

Optionally, the VAXTPU procedure in which you want to go to the specified line. The procedure must be in the current buffer. You can abbreviate the procedure name, but cannot use wildcards; the name is not case sensitive. If more than one procedure matches your request, EVE

shows a list of the matching procedures so you can choose the one you want. Specifying a procedure is useful because some compiler messages refer to line numbers in a procedure.

example

The following command moves the cursor to the start of line 10 in the current buffer:

Command: `LINE 10`

LOWERCASE WORD**format**

LOWERCASE WORD

description

Makes letters lowercase in a single word, select range, or found range. With a select range or found range, **LOWERCASE WORD** changes the letters in the range, starting with the first letter in the range (even if it is not the first letter of the word). A select range takes precedence over a found range. If there is no select range or found range, **LOWERCASE WORD** works on the current word. If you are between words, it works on the next word on the line.

MARK**format**

MARK *marker-name*

description

Puts an invisible marker at the current position, and associates it with the name you specify. Later, using the **GO TO** command, you can return to the marked position. This makes it easier to move through a large buffer or between buffers. To find out marker names, use the **SHOW** command.

parameter***marker-name***

The name you want to mark the current position in the buffer. Marker names are not case sensitive and may contain embedded spaces. You cannot use wildcards (for example, an asterisk is treated as a character in the marker name). If you specify a marker name that is already used, the previous marker is canceled. If you do not specify a marker, **EVE** prompts

EVE-30 EVE Commands

MARK

you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following commands mark the current position as INTRO SEC, and later move the cursor to that position:

```
Command: MARK intro sec
Current position marked as: intro sec
      .
      .
      .
Command: GO TO intro sec
```

MOVE BY LINE

format

MOVE BY LINE

VT300, VT200:

F12

VT100:

MINUS

description

Moves the cursor a line at a time in the direction of the buffer—forward or reverse, as shown in the status line. In forward direction, the cursor moves to the end of the current line or, if you are already there, to the end of the next line. In reverse direction, it moves to the start of the current line or, if you are already there, to the start of the previous line. You can repeat the operation until you reach the bottom or top of the buffer.

MOVE BY PAGE

format

MOVE BY PAGE

description

Moves the cursor a page at a time in the direction of the buffer—forward or reverse, as shown in the status line. *Pages* are bounded by a form-feed character (usually appearing as a small F) or by the top or bottom of the buffer. For more information about page breaks, see the descriptions of the INSERT PAGE BREAK and PAGINATE commands.

MOVE BY WORD

format

MOVE BY WORD

description

Moves the cursor a word at a time in the direction of the buffer—forward or reverse, as shown in the status line. In forward direction, the cursor moves to the start of the next word, if any—that is, the first nonspace character in the word. In reverse direction, it moves to the start of the current word or, if you are already there, to the start of the previous word, if any. You can repeat the operation until you reach the bottom or top of the buffer.

MOVE DOWN

format

MOVE DOWN

VT300, VT200:

↓

VT100:

↓
KP2

EVE-32 EVE Commands
MOVE DOWN

description

Moves the cursor down a line at a time. If the cursor is free—which is the default setting—it moves down in the same column on the screen, regardless of whether text is there or not. If the cursor is bound, it moves down to the corresponding line position, as in EDT, WPS, and other editors. For example, from the end of a line longer than the next line, MOVE DOWN moves the cursor to the end of the next line. It does not move into the unused portion of the buffer. For information about setting the type of cursor motion, see the descriptions of the SET CURSOR BOUND and SET CURSOR FREE commands.

MOVE LEFT

format

MOVE LEFT

VT300, VT200:

←

VT100:

←

KP1

description

Moves the cursor left one character or column at a time. If the cursor is free—which is the default setting—you can move it anywhere in the buffer, whether characters are already there or not. For example, if the left margin is greater than 1, you can move left of the left margin. If the cursor is bound, then from the start of a line it moves to the end of the previous line, if there is one. When you edit an EVE command line, the cursor is always bound and does not move past the start of the line. For information about setting the type of cursor motion, see the descriptions of the SET CURSOR BOUND and SET CURSOR FREE commands.

MOVE RIGHT

format

MOVE RIGHT

VT300, VT200:

→

VT100:

→

KP3

description

Moves the cursor right one character or column at a time. If the cursor is free—which is the default setting—you can move it anywhere in the buffer, whether characters are already there or not. For example, you can move right of the right margin. If the cursor is bound, then from the end of a line it moves to the start of the next line, if there is one. When you edit an EVE command line, the cursor is always bound and does not move past the end of the line. For information about setting the type of cursor motion, see the descriptions of the SET CURSOR BOUND and SET CURSOR FREE commands.

MOVE UP

format

MOVE UP

VT300, VT200:

↑

VT100:

↑

KP5

description

Moves the cursor up a line at a time. If the cursor is free—which is the default setting—it moves up in the same column on the screen, regardless of whether text is there or not. If the cursor is bound, it moves up to the corresponding line position, as in EDT, WPS, and other editors. For example, from the end of a line longer than the previous line, MOVE UP moves the cursor to the end of the previous line. It does not move into the unused portion of the buffer. For information about setting the type of cursor motion, see the descriptions of the SET CURSOR BOUND and SET CURSOR FREE commands.

NEW

format

NEW

description

Creates a new buffer, putting it into the current EVE window. The cursor moves to the top of the new buffer. The new buffer is named MAIN. If a buffer named MAIN already exists, EVE asks you for the name of the new buffer to create. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

You cannot create a buffer with the same name as an existing buffer. For example, you cannot create a buffer named MESSAGES, because EVE has a system buffer with that name (for storing the messages you receive during your editing session). For a list of the buffers you have created, use the SHOW BUFFERS command. For a list of the buffers created by EVE, use the SHOW SYSTEM BUFFERS command.

example

The following command creates a new buffer. Since a buffer named MAIN already exists, EVE asks for you a buffer name. In this case, you call the new buffer TEST.

Command: NEW

Type a new buffer name or press RETURN to cancel: test

NEXT BUFFER

format

NEXT BUFFER

description

Puts your next buffer into the current EVE window and returns the cursor to your last position in that buffer—if the buffer still exists. This lets you toggle between two buffers or cycle through several buffers without having to type their names. (It does *not* create a new buffer or re-create a deleted buffer.) If you have only two buffers, repeating NEXT BUFFER toggles between them. If you have more than two buffers, the next buffer is determined by the order in which you created the buffers. For a list of your buffers, use the SHOW BUFFERS command.

example

In the following example, you first edit a file called ALPHA.TXT, and then edit a file called BETA.TXT. The NEXT BUFFER command then returns you to the ALPHA.TXT buffer.

```
Command: GET FILE alpha.txt
      .
      .
      .
Command: GET FILE beta.txt
      .
      .
      .
Command: NEXT BUFFER
```

NEXT SCREEN

format

NEXT SCREEN

VT300, VT200:

NEXT SCREEN

VT100:

KP0

description

Scrolls vertically to show the next screen's worth of text—roughly, the length of the current EVE window. If the cursor is free—which is the default setting— it moves down in the same column on the screen, regardless of whether text is there or not. If the cursor is bound, it moves down by the corresponding line positions, depending on the shape of your text. For information about setting the type of cursor motion, see the descriptions of the SET CURSOR BOUND and SET CURSOR FREE commands.

NEXT WINDOW

format

NEXT WINDOW

VT300, VT200:

GOLD-NEXT SCREEN

description

Puts the cursor at your last position in the next window, if you are using two or more windows in EVE (same as the OTHER WINDOW command). For example, if you split the EVE main window into three windows, the NEXT WINDOW command does the following:

- From the top window, the cursor returns to your last position in the middle window.
- From the middle window, the cursor returns to your last position in the bottom window.
- From the bottom window, the cursor returns to your last position in the top window.

If you are using only two windows, the NEXT WINDOW, OTHER WINDOW, and PREVIOUS WINDOW commands are the same.

ONE WINDOW

format

ONE WINDOW

description

Restores a single, large window when the EVE main window is split into two or more windows.

OPEN

format

OPEN *filespec*

description

Same as the GET FILE command—puts the file you specify into the current EVE window, creating a new buffer if necessary. This lets you edit another file in the same session.

OPEN SELECTED

format

OPEN SELECTED

description

Opens the file whose name you have selected or found—same as using the GET FILE or OPEN command, without having to type the file name. This is particularly useful to open a file that has a long name, or a name similar to other files, without having to type it exactly.

Steps:

1. Use SELECT, FIND, or WILDCARD FIND to highlight the name of the file you want to edit or create.

A select range takes precedence over a found range. On DECwindows, the select range can be in EVE or in another DECwindows application running concurrently.

2. Use the OPEN SELECTED command.

If the file exists, EVE copies it into a new buffer in the current window. If the file does not exist, EVE creates a new, empty buffer, using the file name and file type for the buffer name. If you specify a file you have already opened in the editing session—that is, a file for which there is already a buffer—EVE returns to your last location in the buffer for that file, if the buffer still exists.

OTHER WINDOW

format

OTHER WINDOW

description

Same as the NEXT WINDOW command—puts the cursor at your last position in the next window, if you are using two or more windows in EVE.

OVERSTRIKE MODE

format

OVERSTRIKE MODE

description

Sets the mode of the current buffer to overstrike, as opposed to insert. Each character you type replaces the character at the current position. (In some editors, this is called *typeover mode* or *replace mode*.) The mode of the buffer is shown in the status line. For editing text, the default mode is insert. Note that the mode is a buffer-specific setting; you can have one buffer set to insert and another buffer set to overstrike. For editing command lines, the default mode matches your terminal setting (according to the DCL command SET TERMINAL), independent of the mode of your text buffers.

PAGINATE

format

PAGINATE

description

Inserts a “soft” page break for a 54-line page, erasing any existing soft breaks within the 54 lines. A soft page break is a form feed followed by the null character—appearing as a small F_N .

Steps:

1. Use the PAGINATE command. EVE then moves back to the previous page break or to the top of the buffer, and checks ahead for page breaks within the next 54 lines.
2. If soft breaks are found within the 54 lines, EVE deletes them, moves down to insert a soft break for a 54-line page, and then puts the cursor on the next line. The page break always appears on a line by itself.

You can then repeat the PAGINATE command to continue paginating the buffer.

If a “hard” page break is found, EVE stops on the line after that page break, in case you want to erase it. A hard page break is a form feed only, typically inserted with the INSERT PAGE BREAK command (CTRL/L key).

PASTE

format

PASTE

description

Same as the INSERT HERE command—inserts the text you have copied or removed.

Note that the PASTE command is not the same as the WPS keypad Paste key. The PASTE (or INSERT HERE) command uses either the Insert Here buffer in EVE or the DECwindows clipboard, depending on your setting. The WPS Paste key uses either the Insert Here buffer or a WPS-style alternate paste buffer—but does *not* use the clipboard.

PREVIOUS SCREEN

format

PREVIOUS SCREEN

VT300, VT200:

PREV SCREEN

VT100:

PERIOD

description

Scrolls vertically to show the previous screen's worth of text—roughly, the length of the current EVE window. If the cursor is free—which is the default setting—it moves up in the same column on the screen, regardless of whether text is there or not. If the cursor is bound, it moves up by the corresponding line positions, depending on the shape of your text. For information about setting the type of cursor motion, see the descriptions of the SET CURSOR BOUND and SET CURSOR FREE commands.

PREVIOUS WINDOW

format

PREVIOUS WINDOW

VT300, VT200:

GOLD-PREV SCREEN

description

Puts the cursor at your last position in the previous window, if you are using two or more windows in EVE. For example, if you split the EVE main window into three windows, the PREVIOUS WINDOW command does the following:

- From the bottom window, the cursor returns to your last position in the middle window.
- From the middle window, the cursor returns to your last position in the top window.
- From the top window, the cursor returns to your last position in the bottom window.

If you are using only two windows, the PREVIOUS WINDOW, NEXT WINDOW, and OTHER WINDOW commands are the same.

QUIT

format

QUIT

description

Ends the editing session without writing out a new file or new version of an existing file. Quitting discards the edits made during the session, except those you have already saved by using SAVE FILE or WRITE FILE commands. If you have modified any buffers that you created, EVE asks you to confirm that you want to quit (to prevent accidentally discarding your edits). If you want to quit, simply press RETURN at the prompt. If you do *not* want to quit, type NO and press RETURN.

example

The following command ends the editing session without saving your edits:

Command: QUIT

Buffer modifications will not be saved, continue quitting?

QUOTE

format

QUOTE

VT300, VT200:

CTRL/V

VT100:

CTRL/V

EVE-42 EVE Commands

QUOTE

description

Enters a control code or other character you specify by a key press. You can quote a character either as part of a command string or to enter the character as text in the buffer. Some control codes appear as a backwards question mark. QUOTE is sensitive to the mode of the buffer (shown in the status line). In insert mode, the quoted character is inserted at the current position. In overstrike mode, the quoted character replaces the current character. You can quote a control code or other character when you enter a string for the FIND or REPLACE commands.

example

The following example inserts an escape character in the buffer:

Command: QUOTE

Press the key to be added: CTRL/J

RECALL

format

RECALL

VT300, VT200:

CTRL/B

VT100:

CTRL/B

description

Recalls a previous EVE command, which you can edit (if necessary) and execute again.

Do not type the command RECALL. If you type RECALL, that command itself is recalled. Instead, use CTRL/B or a key you have defined as RECALL.

REFRESH

format

REFRESH

VT300, VT200:

CTRL/W

VT100:

CTRL/W

description

Refreshes (repaints) the screen, typically to remove extraneous characters that are the result of a system broadcast.

REMEMBER

format

REMEMBER

VT300, VT200:

CTRL/R

VT100:

CTRL/R

description

Ends (“remembers”) a learn sequence and prompts you to press the key to be defined for the sequence. (See the description of the LEARN command.)

Do not type the REMEMBER command. If you type REMEMBER, that command itself is remembered as part of the learn sequence. Instead, use CTRL/R or a key you have defined as REMEMBER.

REMOVE

format

REMOVE

VT300, VT200:

REMOVE

VT100:

KP8

description

Removes a select range or found range, which you can insert elsewhere (same as the CUT command). In the Buffer List buffer, deletes the buffer whose name the cursor is on.

Steps:

1. Use SELECT, FIND, or WILDCARD FIND to highlight the text you want to remove. (A select range takes precedence over a found range.)
2. Use the REMOVE command.
3. To insert the removed text elsewhere, use the INSERT HERE or PASTE command.

The removed text is stored either in the Insert Here buffer in EVE or in the DECwindows clipboard, depending on your setting, and replaces in that storage area whatever you previously removed or copied. The default setting is NOCLIPBOARD, which uses the Insert Here buffer. For more information, see the description of the SET CLIPBOARD command.

When you are in the Buffer List buffer, REMOVE deletes a buffer without your having to type the buffer name, as follows:

1. Use the SHOW BUFFERS command to list the buffers you have created.
2. Put the cursor on the name of the buffer you want to delete.
3. Use REMOVE to delete that buffer.

For more information about deleting buffers, see the description of the DELETE BUFFER command.

REPEAT

format

REPEAT *integer*

description

Repeats the next command or keystroke as often as you specify, without your having to retype it. For example, you can repeat an arrow key or a cursor-movement command, or you can repeat a typing key (such as the dash) or an editing operation (such as an ERASE command). To cancel a pending repeat count, use the RESET command.

NOTE: Do not use CTRL/C to cancel a REPEAT operation. Pressing CTRL/C may cancel the operation, but CTRL/C is not recorded in the journal file, which may make it impossible to recover your work if your editing session is interrupted by a system failure. If you use CTRL/C to cancel an operation, you should exit immediately, saving your edits, and then restart the editing session.

parameter

integer

The number of times you want the next operation repeated. Must be greater than 1. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

In the following example, you repeat the ERASE WORD command five times—that is, you erase the current word and the next four:

```
Command: REPEAT 5  
Command: ERASE WORD
```

REPLACE

format

```
REPLACE { "old-string" ["new-string"] }  
         { old-string [new-string] }
```

description

Replaces one text string with another—that is, EVE searches for the *old string* you specify and replaces it with the *new string* you specify. EVE searches for the old string first in the current direction and then, if necessary, in the opposite direction. If the old string is found in the opposite direction, EVE asks if you want to change the direction of the search and go there. If you want to go there, press RETURN. If you do not want to go there, type NO and press RETURN.

If the old string is found, EVE puts the cursor at the beginning of the string, highlights the found text, and asks you for one of the following choices. You need only type the first letter of the response (and press RETURN).

Response	Effects
YES	Replaces this occurrence of the old string and searches for the next occurrence. This is default choice: you can simply press RETURN.
NO	Skips this occurrence and searches for the next occurrence.
ALL	Replaces all the occurrences, starting with this one, without moving the cursor to each successively found occurrence.
LAST	Replaces this occurrence and stops here.
QUIT	Skips this occurrence and stops here. (You can also press CTRL/Z.)

With YES or ALL, if the search covers the buffer more than once, EVE asks if you want to continue (so you can avoid replacing a string again when the old and new strings are similar).

The REPLACE command is case sensitive. If the old string is all lowercase, EVE searches for any occurrence, regardless of its case

in the buffer (much like the FIND command). If the new string is *also* all lowercase, EVE tries to match the case appropriately for each replacement, as follows:

- A capitalized version of the old string (first letter uppercase, others lowercase) is replaced by a capitalized version of the new string.
- An all-uppercase version of the old string is replaced by an all-uppercase version of the new string.
- Otherwise, the old string is replaced by an all-lowercase version of the new string.

parameters

old-string

The text you want to remove. If the string is more than one word, put it in quotation marks or let EVE prompt you for the string. Use all lowercase to search for any occurrence; use mixed case or all uppercase to search for an exact match. If you do not specify an old string, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

new-string

The text you want to replace the old string. If the string is more than one word, put it in quotation marks or let EVE prompt you for the string. If you do not specify a new string—that is, if you simply press RETURN at the prompt—REPLACE deletes the old string without substituting any text.

example

In the following example, you replace all occurrences of the word *butter* with the word *margarine*. Because the old string is lowercase, EVE finds any occurrence of *butter* regardless of its case in the buffer. Because the new string is also lowercase, EVE matches the case appropriately in the replacement.

```
Command: REPLACE butter margarine
Replace? Type yes, no, all, last, or quit: A
Replaced 8 occurrences.
```

RESET

format

RESET

VT300, VT200:

GOLD-SELECT

description

Cancels any of the following and sets the direction of the buffer to forward:

- Highlighting of a select range or found range
- A press of the GOLD key or GOLD-number combination for a repeat count (with the EDT keypad or WPS keypad)
- An incomplete or recalled command line, or a Choices buffer display when you type an ambiguous command
- Display from the SHOW, SHOW DEFAULTS BUFFER, SHOW SUMMARY, or SHOW WILDCARDS command, thus returning you to the buffer you were working in

RESTORE

format

RESTORE

VT300, VT200:

GOLD-INSERT HERE

description

Reinserts at your current position what you last erased with most ERASE commands or similar EDT keypad or WPS keypad keys—same as using RESTORE LINE, RESTORE SENTENCE, or RESTORE WORD, depending on what you last erased. The restored text is inserted whether the mode of the buffer is insert or overstrike. Existing text is pushed to the right or down. Depending on the amount of text restored and where you are on the line, your text may go past the right margin or even partly out of view. Use FILL commands, if necessary, to reformat (rewrap) your text.

RESTORE CHARACTER

format

RESTORE CHARACTER

description

Puts back at your current position what you last erased with DELETE, ERASE CHARACTER, or similar EDT keypad or WPS keypad keys. RESTORE CHARACTER is sensitive to the mode of the buffer (shown in the status line). In insert mode, the restored character is inserted at the current position. In overstrike mode, the restored character replaces the current character.

RESTORE LINE

format

RESTORE LINE

description

Reinserts at your current position what you last erased with ERASE LINE, ERASE START OF LINE, or similar EDT keypad or WPS keypad keys. The restored text is inserted whether the mode of the buffer is insert or overstrike. Existing text is pushed to the right or down. Depending on the amount of text restored and where you are on the line, your text may go past the right margin or even partly out of view. Use FILL commands, if necessary, to reformat (rewrap) your text.

RESTORE SELECTION

format

RESTORE SELECTION

description

Reinserts at your current position what you last erased with a pending delete operation. This is useful if you inadvertently erased a selection, and it also lets you use the pending delete feature as another way to cut and paste text. The restored text is inserted whether the mode of the buffer is insert or overstrike. Existing text is pushed to the right or down. Depending on the amount of text restored and where you are on the line, your text may go past the right margin or even partly out of view. Use FILL commands, if necessary, to reformat (rewrap) your text.

RESTORE SENTENCE

format

RESTORE SENTENCE

description

Reinserts at your current position what you last erased with the WPS Delete Beginning Sentence key (GOLD-F13 or GOLD-CTRL/J). The restored text is inserted whether the mode of the buffer is insert or overstrike. Existing text is pushed to the right or down. Depending on the amount of text restored and where you are on the line, your text may go past the right margin or even partly out of view. Use FILL commands, if necessary, to reformat (rewrap) your text.

RESTORE WORD

format

RESTORE WORD

VT300, VT200:

GOLD-F13 (except with the WPS keypad)

description

Reinserts at your current position what you last erased with ERASE PREVIOUS WORD, ERASE WORD, or similar EDT keypad or WPS keypad keys. The restored text is inserted whether the mode of the buffer is insert or overstrike. Existing text is pushed forward. Depending on the length of the restored text and where you are on the line, your text may go past the right margin or even partly out of view. Use FILL commands, if necessary, to reformat (rewrap) your text.

RETURN

format

RETURN

VT300, VT200:

RETURN
CTRL/M
ENTER

VT100:

RETURN
CTRL/M

description

Inserts a carriage return at your current position to start a new line of text, or terminates an EVE command or a response to a prompt. In terminating a command or response to a prompt, you can have the cursor anywhere on the command line. Generally, if an EVE command prompts you for required information, such as a file name, search string, or other parameter, simply pressing RETURN at the prompt without typing anything cancels the operation. In some cases, pressing RETURN indicates a default choice.

EVE does not let you define the RETURN key or CTRL/M. You can redefine ENTER.

REVERSE

format

REVERSE

description

Sets the direction of the current buffer to reverse (left and up). The direction of the buffer is shown in the status line. It affects commands like FIND and MOVE BY LINE and some EDT keypad and WPS keypad keys. For buffers you create, the default direction is forward. Note that direction is a buffer-specific setting; you can have one buffer set to forward and another buffer set to reverse. For editing EVE command lines, the default direction is reverse, independent of the direction of your text buffers.

SAVE EXTENDED EVE

format

SAVE EXTENDED EVE *section-filespec*

description

Creates a section file you specify, saving your current key definitions and other extensions for future editing sessions (same as the SAVE EXTENDED TPU command).

Steps:

1. Compile any VAXTPU procedures you have written to extend EVE. You can use EXTEND commands during the editing session, or you can put the procedures in a command file executed when you invoke the editor (with the /COMMAND qualifier).
2. Define any keys you want by using DEFINE KEY, LEARN, SET GOLD KEY, and SET KEYPAD commands. You can do the commands during the editing session, or you can put the commands in an EVE initialization file.
3. To create a section file, use the SAVE EXTENDED EVE command and specify the name of the section file. The default file type is TPU\$SECTION. For example, the following command creates a section file called MYEVE.TPU\$SECTION in your current (default) directory:

```
Command: SAVE EXTENDED EVE myeve
```

4. To use a section file, invoke the editor using the /SECTION qualifier, or define the logical name TPU\$SECTION (particularly if there is a section file you want to use for all or most sessions). For example, the following command invokes the editor using a section file called MYSEC.TPU\$SECTION in your top-level (or login) directory:

```
$ EDIT/TPU/SECTION=sys$login:mysec
```

In specifying the section file, include the device (disk) and directory. Otherwise, VAXTPU assumes the section file is in SYS\$SHARE.

A section file is in binary form, so it is executed quickly. You use one section file at a time. Section files are cumulative, saving the current key definitions and extensions done during the editing session and adding them to those already saved in the section file you are using. Effectively, the section file is your own, customized version of EVE. However, a section file usually does not save margins, tabs, and other settings. Therefore, you may want to use an EVE initialization file to save your editing preferences. (See Section 6.5.9.) The default section file is SYS\$SHARE:EVE\$SECTION.TPU\$SECTION.

parameter

section-filespec

The section file you want to create. The default file type is TPU\$SECTION. You can use logical names in the file specification, but cannot use wildcards. For example, you can use SYS\$LOGIN or other logical names to specify the device or directory where you want the section file created. By default, the section file is created in your current (default) directory. If you do not specify a file, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

SAVE EXTENDED TPU

format

SAVE EXTENDED TPU *section-filespec*

description

Same as the SAVE EXTENDED EVE command—creates a section file you specify, saving your current key definitions and other extensions for future editing sessions.

SAVE FILE

format

SAVE FILE

description

Saves (writes out) the current buffer, without ending the editing session. Similar to the WRITE FILE command, except that you do not specify an output file on the command line. Instead, SAVE FILE uses the output file specification associated with the buffer. Typically, this is the same as the file specified when you invoked EVE or when you used the GET FILE, OPEN, or OPEN SELECTED command. If there is no output file associated with the buffer—for example, if you invoked EVE without specifying a file, or if you created the buffer with the BUFFER or NEW command, or if you are saving an EVE system buffer—then EVE prompts you to enter an output file name. In such a case, specifying an output file does not change the buffer name, but does associate that file with the buffer for later SAVE FILE or WRITE FILE commands or for exiting (except with system buffers). To check the output file specification of the buffer, use the SHOW command.

EVE-54 EVE Commands

SAVE FILE

example

The following commands open a file called MEMO.TXT and then save your edits in a new version of that file:

```
Command: OPEN memo.txt
          .
          .
          .
Command: SAVE FILE
45 lines written to file DISK$1:[GEOFF]MEMO.TXT;2
```

SAVE FILE AS

format

SAVE FILE AS *output-filespec*

description

Saves (writes out) the current buffer to the file you specify, without ending the editing session. Similar to the SAVE FILE or WRITE FILE command, except that SAVE FILE AS requires an output file specification. This lets you save your edits in a file with a different name from the input file. Specifying an output file does not change the buffer name, but does associate that file with the buffer for later SAVE FILE or WRITE FILE commands or for exiting (except with system buffers). To check the output file specification of the buffer, use the SHOW command.

parameter

output-filespec

The output file you want to create for saving the contents of the current buffer. If you do not specify a file, EVE prompts you for one. Pressing RETURN or DO at the prompt without specifying a file, writes the buffer to the output file associated with that buffer, if there is one (same as with the SAVE FILE or WRITE FILE command).

example

The following commands open a file called ROUGH.DAT and then save your edits as a file called FINAL.TXT:

```
Command: OPEN rough.dat
          .
          .
          .
Command: SAVE FILE AS final.txt
38 lines written to DISK$1:[GEOFF]FINAL.TXT;1
```

SELECT

format

SELECT

VT300, VT200:

SELECT

VT100:

KP7

description

Selects text for an editing operation such as COPY, FILL, REMOVE, OPEN SELECTED, or UPPERCASE WORD. In the Buffer List buffer, lets you view the buffer whose name the cursor is on.

Steps:

1. Put the cursor where you want to begin the selection.
2. Use the SELECT command to begin selecting text.
3. Move the cursor to select text. Whatever text the cursor crosses is highlighted in reverse video. Blank lines are not highlighted. If you move the cursor forward, the select range begins with the current character. If you move the cursor back (reverse direction), the select range begins with the character left of the cursor. If you move the cursor by using FIND, FIND NEXT, or WILDCARD FIND in forward direction, the select range ends at the start of the found string—that is, the found text is *not* part of the select range.

You can then use a command such as COPY, FILL, REMOVE, or UPPERCASE WORD with the select range.

On DECwindows, you can select text by using MB1 as follows. The cursor moves where you are pointing.

EVE-56 **EVE Commands**
SELECT

Mouse Action	Selection
1 Click	Cancels a selection, if any.
2 Clicks	Selects all of the word that the pointer is on.
3 Clicks	Selects all of the line that the pointer is on.
4 Clicks	Selects all of the paragraph that the pointer is on.
5 Clicks	Selects all of the buffer (same as the SELECT ALL command).
Drag (press-move-release)	Selects a block of text, starting with the character the pointer is on when you press MB1, and ending with the character the pointer is on when you release MB1.

To cancel the selection, do any of the following:

- Use the RESET command.
- Repeat the SELECT command (for example, by pressing the SELECT key again).
- Click MB1 once.
- If the selection was done by clicking or dragging the mouse, you can simply move the cursor out of the select range (for example, by pressing the down arrow key).

When you are in the Buffer List buffer, SELECT is specially redefined to view a buffer, as follows:

1. Use the SHOW BUFFERS command to get a list of the buffers you have created, or use the SHOW SYSTEM BUFFERS command to get a list of the buffers EVE creates.
2. Put the cursor on a buffer name in the list, and use SELECT or, on DECwindows, click MB1 twice.

EVE then puts that buffer into the current window. Effectively, this is the same as using the BUFFER command without having to type the buffer name.

SELECT ALL

format

SELECT ALL

description

Selects all of the current buffer—regardless of your position—so you can perform an editing operation, such as COPY, FILL, or REMOVE.

Using SELECT ALL or clicking MB1 five times temporarily disables pending delete, to prevent accidentally erasing all of the buffer.

SET BUFFER

format

SET BUFFER { *MODIFIABLE*
READ_ONLY
UNMODIFIABLE
WRITE }

description

Sets the editing status of the current buffer—whether you can modify the buffer or whether EVE saves (writes out) the buffer on exiting. The modification status is indicated in the status line by Insert or Overstrike (if the buffer is modifiable) or by Unmodifiable. The read/write status is shown by Read-only or Write in the status line. Typically you set a buffer to read-only, unmodifiable, or both to prevent inadvertently changing text you want to keep intact, such as reference data or a previous draft. If you create a “scratchpad” buffer as a temporary work area, you may want to set it to read-only and modifiable. This lets you edit the buffer, but EVE does not write out (save) that buffer on exiting.

parameters

MODIFIABLE

Default setting. The buffer can be modified (edited). Also restores the previous mode of the buffer (insert or overstrike).

READ_ONLY

The buffer is write-locked and unmodifiable. Text-editing functions do not work in the buffer, and on exiting, EVE does not write out (save) the buffer. However, you can write out the buffer by using the WRITE FILE, SAVE FILE, or SAVE FILE AS command.

EVE-58 EVE Commands

SET BUFFER

UNMODIFIABLE

The buffer cannot be modified. For example, you cannot insert or erase text in the buffer. In the status line, Unmodifiable replaces the Insert or Overstrike indicator.

WRITE

Default setting. The buffer is write-enabled (opposite of READ_ONLY). On exiting, if the buffer has been modified, EVE writes it out or asks if you want to write it out.

You can specify only one keyword per command. If you do not specify a keyword, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command sets the current buffer to read-only and modifiable.

```
Command: SET BUFFER READ_ONLY  
Command: SET BUFFER MODIFIABLE
```

SET CLIPBOARD

format

SET CLIPBOARD

description

Enables the DECwindows clipboard for copying, cutting, and pasting text, instead of the Insert Here buffer in EVE. Using the clipboard lets you transfer text between EVE and other DECwindows applications.

You can enable the clipboard only if you invoke EVE using /DISPLAY=DECWINDOWS. Otherwise, the command is invalid. The default setting is NOCLIPBOARD, which uses the Insert Here buffer.

SET CURSOR BOUND

format

SET CURSOR BOUND

description

Enables bound cursor motion, similar to that in EDT and WPS. A bound cursor cannot move into unused portions of the buffer. For example, if you are at the end of a line and press the right arrow key, the cursor moves to the start of the next line. If you want bound cursor motion for all or most editing sessions, put the SET CURSOR BOUND command in your EVE initialization file. (See Section 6.5.9.)

The SET KEYPAD WPS command automatically sets the cursor to bound.

SET CURSOR FREE

format

SET CURSOR FREE

description

Enables free cursor motion, which lets you move anywhere in the buffer and insert text whether characters are already there or not. (Default setting.) If you move up and down, the cursor stays in the same column on the screen. You can move left of the left margin (if the left margin is greater than 1), right of the right margin, or past the [End of file] marker (if the buffer is shorter than the current window). For example, if you are at the end of a line and press the right arrow key, the cursor moves past the end of the line and you can put text there. By contrast, a bound cursor moves to the start of the next line.

SET FIND NOWHITESPACE

format

SET FIND NOWHITESPACE

description

Enables FIND and WILDCARD FIND commands to match spaces and tabs exactly as in the search string, rather than as “white space,” and to search for multiword strings that are entirely on one line. (Default setting.)

example

In the following example, you search for *Mark Twain* with exactly one space between the words and entirely on one line:

```
Command: SET FIND NOWHITESPACE  
Command: FIND Mark Twain
```

SET FIND WHITESPACE

format

SET FIND WHITESPACE

description

Enables FIND and WILDCARD FIND commands to treat spaces, tabs, and up to one line break as “white space.” This lets you search for a string of two or more words, regardless of how they are separated. The default setting is NOWHITESPACE—that is, EVE matches spaces and tabs in the search string exactly, and search strings do not span a line break. If you want white-space find for all or most editing sessions, put the SET FIND WHITESPACE command in your EVE initialization file. (See Section 6.5.9.)

example

In the following example, you search for *Mark Twain* whether there is one or more spaces or tabs between the words or if *Mark* is at the end of one line and *Twain* at the start of the next line:

```
Command: SET FIND WHITESPACE  
Command: FIND Mark Twain
```

SET GOLD KEY

format

SET GOLD KEY [*key-name*]

description

Defines a key as the GOLD key for use with other keys, and enables several GOLD key combinations. You can type the key name on the command line or let EVE prompt you to press the key you want to set as GOLD. The GOLD key increases the possible key bindings. For example, you can define F20 to execute one command and define the GOLD-F20 combination to execute another command. To execute one function, you press F20 alone; to execute the other function, you press GOLD and then press F20. You can also define combinations of GOLD and a typing key, such as GOLD-C. Setting the GOLD key—by itself or by setting the EDT keypad or WPS keypad—automatically defines some GOLD combinations for the arrow keys and the mini keypad, unless you have defined the keys otherwise. Table EVE-1 lists the default GOLD combinations. Note that some GOLD combinations require a VT300- or VT200-series terminal (for example, GOLD-HELP).

Table EVE-1: EVE Default GOLD Key Combinations

Key	Definition
GOLD-F13	RESTORE WORD or WPS Delete Beginning Sentence
GOLD-HELP	HELP KEYS (list)
GOLD-FIND	WILDCARD FIND
GOLD-INSERT HERE	RESTORE
GOLD-REMOVE	STORE TEXT
GOLD-SELECT	RESET
GOLD-PREV SCREEN	PREVIOUS WINDOW
GOLD-NEXT SCREEN	NEXT WINDOW
GOLD-↑	TOP
GOLD-←	START OF LINE
GOLD-↓	BOTTOM
GOLD-→	END OF LINE

SET GOLD KEY overrides any current definition of the key you specify, whether the key is defined by EVE, the EDT keypad, the WPS keypad, or a definition of your own. You can have only one key set as GOLD at a time. Setting the EDT keypad or WPS keypad makes PF1 the GOLD key, overriding any current definition of PF1. However, if you set a different key as the GOLD key, then the EDT keypad and WPS keypad use your GOLD

EVE-62 EVE Commands

SET GOLD KEY

key. In such a case, using the SET NOGOLD KEY command cancels your GOLD key and restores PF1 as the GOLD key for the EDT keypad or WPS keypad.

parameter

key-name

The key you want to set as GOLD. You cannot abbreviate the key name. For more information about EVE key names, see Section 6.4.3. If you do not specify a key name, EVE prompts you to press the key you want to define. Pressing the RETURN key or CTRL/M at the prompt cancels the operation, because those keys cannot be redefined.

example

The following commands set PF1 as the GOLD key, and then define the combination of GOLD and the letter C as the CENTER LINE command. Typing a C or c by itself still inserts that letter. In specifying a GOLD key combination, use a dash, slash, or underscore as a delimiter in the key name.

```
Command: SET GOLD KEY pf1
Command: DEFINE KEY= gold-c center line
```

SET KEYPAD EDT

format

SET KEYPAD EDT

description

Enables the EDT-style keypad, defining the numeric keypad keys and other keys. To save the EDT keypad for future sessions, put the SET KEYPAD EDT command in your EVE initialization file, or use the SAVE EXTENDED EVE command to create a section file.

Setting the EDT keypad does not completely emulate EDT. The following is a list of the important differences between the EDT keypad in EVE and real EDT. For information on converting from EDT to EVE, see Section 6.8.

- The EDT keypad makes PF1 the GOLD key, overriding any current definition of PF1. However, if you set a different key as GOLD (with the SET GOLD KEY command), your GOLD key is used. You can have only one key set as the GOLD key at a time.
- If you define keys that EDT ordinarily defines, such as KP8, GOLD-KP8, or CTRL/U, your definitions override the EDT definitions.

EVE Commands EVE-63

SET KEYPAD EDT

- In addition to EDT keys, setting the EDT keypad defines the same GOLD combinations as the SET GOLD KEY command. For example, GOLD-FIND is defined as WILDCARD FIND and GOLD-↓ is defined as BOTTOM. (See Table EVE-1.)
- PF2 is defined as HELP KEYPAD, which draws a diagram of the current keypad, and GOLD-PF2 is defined as HELP KEYS, which lists all the current key definitions.
- GOLD-KP7 is defined as DO, for typing EVE commands. EVE does *not* support EDT line-mode commands or nokeypad commands.
- GOLD-KP8 is defined as FILL, to reformat a select range, found range, or the current paragraph. If you want the key to fill only a select range or found range, as in real EDT, redefine the key as FILL RANGE.
- On VT300- and VT200-series terminals, NEXT SCREEN and PREV SCREEN on the mini keypad are defined slightly differently from EVE. In EVE, the keys scroll the length of the current window. With the EDT keypad, the keys scroll 75% of the window size.
- CTRL/H, CTRL/J, and CTRL/U are defined to emulate EDT. Their standard EVE definitions are slightly different. Also, the EDT keypad defines CTRL/K as LEARN.
- CTRL/Z is defined as EXIT, to end the editing session. In real EDT, CTRL/Z exits to line mode. To emulate this in EVE, you can redefine CTRL/Z as follows:

```
Command: DEFINE KEY= CTRL/Z DO
```
- Some other control keys are defined differently from real EDT: CTRL/A is defined as CHANGE MODE (to switch between insert mode and overstrike). CTRL/E is defined as END OF LINE (which is slightly different from the EDT keypad EOL key). CTRL/R is defined as REMEMBER (to end a learn sequence).
- CTRL/C may cancel an operation, but its use is *not* recommended, because CTRL/C is not recorded in the journal file, which may make it impossible to recover your work if your editing session is interrupted by a system failure. If you use CTRL/C to cancel an operation, you should exit immediately, saving your edits, and then restart the editing session.
- The EDT keypad defines ENTER as RETURN, to terminate a command or start a new line. You can redefine the ENTER key, but cannot redefine the RETURN key or CTRL/M.

EVE-64 **EVE Commands**
SET KEYPAD EDT

- Some EDT keypad definitions use the corresponding EVE commands, which may have slightly different names but are usually functionally similar to EDT. The KP1 key is defined as MOVE BY WORD, which uses slightly different word boundaries. In EVE, a *word* includes the trailing spaces or tabs until the next word separator (typically, a printing character).
- EVE key names are usually the same as at the DCL level and therefore are different from EDT key names. For more information about EVE key names, see Section 6.4.3. For examples of equivalent EDT and EVE key names, see Section 6.8.
- In EVE, using the SELECT command and then the REMOVE (or CUT) command—without moving the cursor—does not clear the paste buffer. Instead, it selects and removes the current character.
- The following EDT keypad keys use either the Insert Here buffer in EVE or the DECwindows clipboard, depending on your setting:

INSERT HERE or GOLD-KP6
REMOVE (KP6)
Append (KP9)
EDT Replace (GOLD-KP9)
Subs (GOLD-ENTER)

The default setting is NOCLIPBOARD, which uses the Insert Here buffer. See the description of the SET CLIPBOARD command.

- By default EVE uses a free cursor, which you can move anywhere in the buffer regardless of whether text is already there. To enable an EDT-style bound cursor, use the SET CURSOR BOUND command. The type of cursor motion affects the following EDT keypad keys:

EDT Next Screen (NEXT SCREEN)
EDT Previous Screen (PREV SCREEN)
Sect (KP8)

The EDT Char key (KP3) uses bound cursor motion even if the cursor is set to free.

- Some key functions are sensitive to the mode of the buffer (insert or overstrike):

Del C (COMMA)
SpecIns (GOLD-KP3)

Also, remember that some EVE commands are mode sensitive, such as RESTORE CHARACTER, which the EDT keypad binds to GOLD-COMMA.

- To set distances for scrolling to begin automatically, use the SET SCROLL MARGINS command, in place of the EDT command SET CURSOR. Note that in EVE, scroll margins are measured from the top and the bottom respectively. For example, with a 24-line terminal screen (21-line main window), the command SET SCROLL MARGINS 5 6 is equivalent to the EDT command SET CURSOR 5:15. The default settings are 0 0 (scrolling begins when you move past the top or bottom of the window).
- Searches follow EVE rules for case sensitivity and direction. Because EVE does not define the RETURN and ENTER keys differently, as EDT does, search strings cannot contain a carriage return. However, you can use the SET FIND WHITESPACE command to enable searching across line breaks or use the WILDCARD FIND command to search for text at the start or end of a line.
- Exiting from EVE creates a new file only if you have made changes to the buffer (and not yet written it out). Quitting discards your edits, but if you have made changes to the buffer, EVE asks you to confirm that you want to quit. Also, if the buffer or buffers have not been modified (or already written out and not modified since then), EXIT and QUIT are the same—no new file is produced.
- The EVE commands SHIFT LEFT and SHIFT RIGHT moves the horizontal position of the window relative to the buffer; whereas the EDT nokeypad commands SHL and SHR move the buffer relative to the window. Thus, in EVE, the command SHIFT RIGHT 8 is equivalent to SHL in EDT—column 9 of your text appears in the leftmost column of the screen.
- EDT features *not* implemented in EVE:

GOLD key equivalents for control keys. For example, GOLD-U and GOLD-Z are not defined, although CTRL/U and CTRL/Z are defined.

Keys for tab adjustments. To change tab stops, use the SET TABS AT or SET TABS EVERY command. You can also define a key for the WPS keypad Ruler key (WPS GOLD-R) and use the ruler to add or delete tab stops.

For information about customizing EVE to emulate EDT more closely, see Section 6.8.

SET KEYPAD NOEDT

format

SET KEYPAD NOEDT

description

Disables (undefines) the EDT keypad, restoring the default keypad for the type of terminal you are using:

- On a VT300- or VT200-series terminal, the effect is the same as using the SET KEYPAD NUMERIC command.
- On a VT100-series terminal, it is the same as the SET KEYPAD VT100 command.

Keys defined with DEFINE KEY, LEARN, or SET GOLD KEY commands remain defined. However, remember that any learn sequences that use EDT keypad keys will not work properly, because the keys are now undefined or defined differently. SET KEYPAD NOEDT cancels the current GOLD key if it was set by enabling the EDT keypad, but does not cancel the GOLD key if you set it with the SET GOLD KEY command.

SET KEYPAD NOWPS

format

SET KEYPAD NOWPS

description

Disables (undefines) the WPS keypad, restoring the default keypad for the type of terminal you are using:

- On a VT300- or VT200-series terminal, the effect is the same as using the SET KEYPAD NUMERIC command.
- On a VT100-series terminal, it is the same as the SET KEYPAD VT100 command.

Keys defined with DEFINE KEY, LEARN, or SET GOLD KEY commands remain defined. However, remember that any learn sequences that use WPS keypad keys will not work properly, because the keys are now undefined or defined differently. SET KEYPAD NOWPS cancels the current GOLD key if it was set by enabling the WPS keypad, but does not cancel the GOLD key if you set it with the SET GOLD KEY command.

Disabling the WPS keypad does not restore free cursor motion. To restore free cursor motion, use the SET CURSOR FREE command.

SET KEYPAD NUMERIC

format

SET KEYPAD NUMERIC

description

Sets the numeric keypad to the default state, canceling the current keypad setting (EDT, VT100, or WPS).

This is the default setting except on VT100-series terminals. The command is *not* valid on VT100-series terminals.

Keys defined with DEFINE KEY, LEARN, or SET GOLD KEY commands remain defined. However, remember that any learn sequences that use EDT keypad or WPS keypad keys will not work properly, because the keys are now undefined or defined differently. SET KEYPAD NUMERIC cancels the current GOLD key if it was set by the EDT keypad or WPS keypad, but does not cancel the GOLD key if you set it with the SET GOLD KEY command. Figure EVE-1 shows the EVE default keypad for VT300- and VT200-series terminals.

SET KEYPAD VT100

format

SET KEYPAD VT100

description

Enables the VT100-style keypad, canceling the current keypad setting (EDT, NUMERIC, or WPS).

This is the default setting if you are using a VT100-series terminal.

Keys defined with DEFINE KEY, LEARN, or SET GOLD KEY commands remain defined. However, remember that any learn sequences that use WPS keypad keys will not work properly, because the keys are now undefined or defined differently. SET KEYPAD VT100 cancels the current GOLD key if it was set by the EDT keypad or WPS keypad, but does not cancel the GOLD key if you set it with the SET GOLD KEY command. Figure EVE-2 shows the EVE default keys for VT100-series terminals.

Figure EVE-1: EVE Default Keys for VT300- and VT200-Series Terminals

Result of SET KEYPAD NOEDT or SET KEYPAD NOWPS Commands

	Exit
F9	F10

- ☒ DELETE
- Tab TAB
- Return RETURN
- Enter RETURN
- PF4 DO

Change Direction	Move By Line	Erase Word Res Wer	Change Mode
F11	F12	F13	F14

- CTRL/A CHANGE MODE
- B RECALL
- E END OF LINE
- H START OF LINE
- I TAB
- J ERASE WORD
- L INSERT PAGE BREAK
- M RETURN
- R REMEMBER
- U ERASE START OF LINE
- V QUOTE
- W REFRESH
- Z EXIT

Help	Do
Keypad	
Keys	

Find	Insert Here	Remove
Wild Find	Restors	Stors Text
Select	Prev Screen	Next Screen
Reset	Prev Window	Next Window
	Top	
Sta of LI	Bottom	End of LI

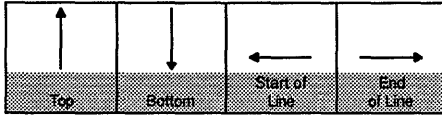
GOLD key functions are shown in gray shading.

Sample Function or Keypad Key

Help	← Key Label
Keypad	← Default Function
Keys	← GOLD Function

Figure EVE-2: EVE Default Keys for VT100-Series Terminals

Default on VT100 Terminal
 Available on VT200 Terminal With SET KEYPAD VT100 Command



Delete DELETE
 Tab TAB
 Return RETURN
 Backspace START of LINE
 Linefeed ERASE WORD

CTRL/A CHANGE MODE
 B RECALL
 E END of LINE
 H START of LINE
 I TAB
 J ERASE WORD
 L INSERT PAGE BREAK
 M RETURN
 R REMEMBER
 U ERASE START OF LINE
 V QUOTE
 W REFRESH
 Z EXIT

Find	Help Keypad	Change Direction	Do
Select	Remove	Insert Here	Move By Line
	↑		Erase Word
←	↓	→	Change Mode
Next Screen	Prev Screen		

GOLD key functions are shown in gray shading.

ZK-6301-GE

SET KEYPAD WPS

format

SET KEYPAD WPS

description

Enables the WPS-style keypad, defining the numeric keypad and other keys, and setting the cursor to bound. To save the WPS keypad for future sessions, put the SET KEYPAD WPS command in your EVE initialization file, or use the SAVE EXTENDED EVE command to create a section file.

SET KEYPAD WPS provides most WPS keypad keys for "GOLD-key editing." It does not fully implement or emulate WPS. The following is a list of differences between the WPS keypad in EVE and real WPS:

- The WPS keypad makes PF1 the GOLD key, overriding any current definition of PF1. However, if you set a different key as GOLD (with the SET GOLD KEY command), your GOLD key is used. You can have only one key set as the GOLD key at a time.

EVE-70 EVE Commands

SET KEYPAD WPS

- In addition to WPS keys, setting the WPS keypad defines the same GOLD combinations as the SET GOLD KEY command—except that GOLD-F13 is defined as Delete Beginning Sentence. (See Table EVE-1.)
- You can use GOLD-number combinations for repeat counts. For example, to repeat the next keystroke or command five times, you can press GOLD-5. However, you cannot repeat the WPS Paste key this way, because WPS Paste interprets GOLD-1 through GOLD-9 as specifying WPS-style alternate paste buffers.
- If you define keys that WPS ordinarily defines, such as KP5 or GOLD-R, your definitions override the WPS definitions.
- GOLD-[is defined as DO for typing EVE commands, and defines both GOLD-> and CTRL/K as LEARN. To end a learn sequence, press a key defined as REMEMBER (CTRL/R), or press the WPS Halt key (GOLD-·).
- CTRL/J and F13 are both defined as Delete Previous Word. GOLD-F13 and GOLD-CTRL/J are both defined as Delete Beginning Sentence.
- Both GOLD-PF3 and GOLD-PF4 are defined as RESTORE, which reinserts what you last erased with the WPS Delete Word key (PF3), WPS Delete Beginning Sentence key, ERASE LINE command, and so on, but does not put back the last character erased or deleted. Therefore, you may want to define GOLD-PF3 as RESTORE WORD, and define GOLD-PF4 as RESTORE CHARACTER.
- The RESTORE SENTENCE command reinserts what you last erased with the Delete Beginning Sentence key (GOLD-F13 or GOLD-CTRL/J). Setting the WPS keypad does *not* define a key for RESTORE SENTENCE. Therefore, you may want to define a key for RESTORE SENTENCE.
- With the WPS keypad, pressing either SELECT on the mini keypad or PERIOD on the keypad also sets the direction of the buffer to forward. However, typing the SELECT command or using the mouse to select text does not change the direction.
- When you are in the Buffer List buffer, you can press the WPS Cut key (MINUS or REMOVE) to delete the buffer whose name the cursor is on (same as REMOVE). For more information, see the description of the SHOW BUFFERS command.
- WPS keypad keys do *not* use the DECwindows clipboard. For example, the WPS Copy, Cut, and Paste keys use EVE's Insert Here buffer or a WPS-style, alternate paste buffer which you specify by number (GOLD-1 through GOLD-9). EVE commands and EDT keypad keys use either the Insert Here buffer or the clipboard, depending on your setting. (See the description of the SET CLIPBOARD command.)

- When you use the WPS Ruler key (GOLD-R), only one ruler can be active at a time. Rulers cannot be embedded in a document. Setting margins or paragraph indent does *not* automatically rewrap or reformat text. To reformat text, use FILL commands.
- Scrolling with WPS keypad keys is halted when you press any key—not just the WPS Halt key (GOLD-·). Pressing a key to stop scrolling executes whatever function is assigned to that key.
- Setting the WPS keypad automatically sets the cursor to bound, which does not move into the unused portion of the buffer. To enable a free cursor, use the SET CURSOR FREE command, which is otherwise the EVE default setting. The type of cursor motion affects the following WPS keypad keys:

Advance (KP0)
 Backup (KP1)
 Scroll Advance (GOLD-KP0)
 Scroll Backup (GOLD-KP1)

- The WPS keypad defines the following keys for pagination:

Key	Definition
PF2	MOVE BY PAGE. Puts the cursor on the next or previous page break, depending on the direction of the buffer.
GOLD-PF2	PAGINATE. Inserts a “soft” break for a 54-line page—a form feed and a null character, appearing as a small F_N .
GOLD-N	INSERT PAGE BREAK. Inserts a “hard” page break—a form feed appearing as a small F . Same as CTRL/L.
GOLD-P	WPS Page Marker. Inserts a “soft” page break.

- Searches follow EVE rules for case sensitivity and direction. For more information, see the description of the FIND command.
- In EVE, *paragraphs* are bounded by a blank line, the top or bottom of the buffer, a page break, or a DIGITAL Standard Runoff command (such as .BLANK). *Sentences* are bounded by a period, question mark, or exclamation point. Periods in Runoff commands or in decimal numbers are treated as sentence boundaries.
- Paragraph indent done with the SET PARAGRAPH INDENT command is relative to the left margin of the buffer; done with the WPS Ruler key (GOLD-R), it is independent of the margin.
- GOLD-C is defined as CENTER LINE, which uses spaces to center the current line between the left and right margins. It does not leave a centering mark.

EVE-72 EVE Commands

SET KEYPAD WPS

- Exiting does not delete the old version of the input file. Also, if the buffer or buffers have not been modified (or already written out and not modified since then), EXIT and QUIT are the same—no new file is produced. EVE defines both F10 and CTRL/Z as EXIT. The WPS keypad defines GOLD-F as EXIT, and defines GOLD-K as QUIT.
- WPS features *not* implemented in EVE:
 - Abbreviation or library documents
 - Control commands for printing
 - Editor math
 - Footnotes, paragraph numbering, or table of contents
 - Hyphenation and nonbreaking spaces
 - Output files other than ASCII
 - Superscripts, subscripts, or composite characters
 - VIEW mode
 - TDE (two-dimension editor)
 - UDP (user-defined WPS procedures)
 - Word-wrap returns (In EVE, all lines end in a “hard” return.)

SET LEFT MARGIN

format

SET LEFT MARGIN *integer*

description

Sets the left margin of the current buffer to the column you specify. This does not change existing text, but only affects new text or text you reformat with CENTER LINE or FILL commands. When you press RETURN or use FILL commands, or when EVE wraps text automatically, lines start at the left margin. The left margin is a buffer-specific setting; you can have a different left margin for each buffer. To find out the current margins of the buffer, use the SHOW command. If you want a particular left margin for all or most editing sessions, put the SET LEFT MARGIN command in your initialization file. When you invoke EVE using that initialization file, the setting then applies to the main (or first) buffer and to an EVE system buffer named \$DEFAULTS\$, so that each buffer you create has the same left margin. (See Section 6.5.9.)

parameter

integer

The column at which you want the left margin. Must be less than the right margin. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation. The default left margin is 1 (leftmost column).

example

The following command sets the left margin to 5. To reformat existing text according to the new margin, use FILL commands.

Command: SET LEFT MARGIN 5

SET NOCLIPBOARD

format

SET NOCLIPBOARD

description

Disables the DECwindows clipboard for copying, cutting, and pasting text, and enables the Insert Here buffer in EVE. (Default setting.)

SET NOGOLD KEY

format

SET NOGOLD KEY

description

Cancels (undefines) the current GOLD key, so you can define that key by itself. (Default setting.) You can have only one key set as GOLD at a time. If you set the GOLD key other than PF1 and set the EDT keypad or WPS keypad, your GOLD key is used. You can then define PF1 like any other key, or use SET NOGOLD KEY to restore PF1 as the GOLD key for the EDT keypad or WPS keypad, canceling your GOLD key. SET NOGOLD KEY does not cancel or undefine GOLD key combinations, but they cannot be executed unless another key is set as GOLD, either by using the SET GOLD KEY command or by setting the EDT keypad or WPS keypad.

example

In the following example, you set F20 as the GOLD key, and then enable the EDT keypad, which ordinarily uses PF1 as the GOLD key. The SET NOGOLD KEY command then cancels F20, making PF1 the GOLD key:

Command: SET GOLD KEY F20

Command: SET KEYPAD EDT

Command: SET NOGOLD KEY

GOLD key restored to PF1 in the EDT keypad.

SET NOPENDING DELETE

format

SET NOPENDING DELETE

description

Disables the deletion of selected text when you use DELETE or type new text. (Default setting.) If you select text in the buffer, typing new text adds characters to the select range, and using DELETE erases only the character left of the cursor. In other words, if pending delete is disabled, DELETE works the same whether there is a select range or not.

SET NOWRAP

format

SET NOWRAP

description

Disables automatic wrapping in the current buffer, so that as you type at the end of a line, your text may go past the right margin. This is useful for editing very long lines, such as wide multicolumn tables or lengthy program statements that are progressively indented. Depending on the width of the EVE window, your text may go out of view.

SET PARAGRAPH INDENT

format

SET PARAGRAPH INDENT *[[+|-]]integer*

description

Sets the number of spaces to be added or subtracted at the start of paragraphs in the current buffer—relative to the left margin. This does not change existing text, but only affects paragraphs you create or reformat with FILL commands. In EVE, a *paragraph* is bounded by any of the following:

- Blank line
- Top or bottom of the buffer
- Page break (form-feed character)
- DIGITAL Standard Runoff command (such as .BLANK)

Paragraph indent is a buffer-specific setting; you can have a different setting for each buffer. If you want a particular paragraph indent for all or most editing sessions, put the SET PARAGRAPH INDENT command in your initialization file. When you invoke EVE using that initialization file, the setting then applies to the main (or first) buffer and to an EVE system buffer named \$DEFAULTS\$, so that each buffer you create has the same paragraph indent. (See Section 6.5.9.)

If the paragraph indent is other than 0 (the default setting), you cannot use FILL or FILL RANGE commands to reformat a range that does not begin at the beginning of a paragraph.

parameter

integer

The number of spaces to be added or subtracted to the start of paragraphs, relative to the left margin of the buffer. If the value is positive (or unsigned), the sum of the left margin and paragraph indent must be less than the right margin. A negative value produces a "hanging" paragraph—its first line starts left of the left margin. The algebraic sum of a negative indent and the left margin must be at least 1. If you do not specify a paragraph indent, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation. The default setting is 0 (no indent).

example

The following examples show how to set paragraph indent, including a negative indent for a "hanging" paragraph:

Command: SET PARAGRAPH INDENT 4

Sets the paragraph indent at 4 columns from the left margin. Thus, if your left margin is 5, the first line of a new paragraph starts at column 9 and the remaining lines in column 5. To reformat existing text use the FILL or FILL PARAGRAPH command.

Command: SET LEFT MARGIN 4

Command: SET PARAGRAPH INDENT -3

·
·
·

Command: SET PARAGRAPH INDENT 0

Command: SET LEFT MARGIN 1

Sets the left margin at column 4 and the paragraph indent at three columns to the left of the left margin. Thus, the first line of a new paragraph starts in column 1, and the rest of the lines in column 4 (called a "hanging" paragraph). This is useful to format lists, for example, to have a bullet or counter three spaces left of the left margin. In restoring your previous settings, note the order of the commands (paragraph indent 0, left margin 1).

SET PENDING DELETE

format

SET PENDING DELETE

description

Enables deletion of a select range when you use **DELETE** or type new text. This is useful for quickly erasing or replacing a block of text.

Steps:

1. Use the **SET PENDING DELETE** command.
2. Select text you want to erase. (See the description of the **SELECT** command.)
3. Use the **DELETE** command or type new text. The selected text is erased, or it is replaced by the new text.

To reinsert what you erased, use the **RESTORE SELECTION** command. If you want pending delete enabled for all or most editing sessions, put the **SET PENDING DELETE** command in your EVE initialization file. (See Section 6.5.9.) The default setting is **NOPENDING DELETE**. If you have selected text, using **DELETE** erases the character left of the cursor and typing new text inserts the new characters.

If you select the entire buffer (with the **SELECT ALL** command or by clicking MB1 five times), pending delete is disabled, to prevent accidentally erasing all of the buffer.

SET RIGHT MARGIN

format

SET RIGHT MARGIN *integer*

description

Sets the right margin of the current buffer to the column you specify. This does not change existing text, but only affects new text or text you reformat with **CENTER LINE** or **FILL** commands. When EVE wraps text automatically, or when you use **FILL** commands, no characters will go beyond the right margin. The right margin is a buffer-specific setting; you can have a different right margin for each buffer. To find out the current margins of the buffer, use the **SHOW** command. If you want a particular right margin for all or most editing sessions, put the **SET RIGHT MARGIN** command in your initialization file. When you invoke

EVE using that initialization file, the setting then applies to the main (or first) buffer and to an EVE system buffer named \$DEFAULTS\$, so that each buffer you create has the same right margin. (See Section 6.5.9.)

parameters

integer

The column at which you want the right margin. Must be greater than the left margin (or greater than the sum of the left margin and the paragraph indent). If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation. The default right margin is one column less than the screen width. Typically, the width is 80 columns; the default right margin is then 79.

example

The following command sets the right margin to 65. To reformat existing text according to the new margin, use FILL commands.

Command: SET RIGHT MARGIN 65

SET SCROLL MARGINS

format

SET SCROLL MARGINS *integer1*[%] *integer2*[%]

description

Sets the top and bottom distances at which scrolling begins automatically as you move the cursor up and down. You specify these distances as numbers of lines or as percentages of the window size. Scroll margins apply to all windows in EVE. Also, EVE converts numbers of lines into percentages, and uses the percentages when you split the main window into two or more windows. If you want particular scroll margins for all or most editing sessions, put the SET SCROLL MARGINS command in your EVE initialization file. (See Section 6.5.9.)

parameters

integer1

The number of lines down from the top of a window at which you want scrolling to begin. Cannot overlap the bottom scroll margin (*integer2*). The default setting is 0—that is, scrolling starts when you move past the top of the window. If you do not specify a value, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything keeps the current value.

EVE-78 EVE Commands

SET SCROLL MARGINS

integer2

The number of lines up from the bottom of a window at which you want scrolling to begin. Cannot overlap the top scroll margin (*integer1*). The default setting is 0—that is, scrolling starts when you move past the bottom of the window. If you do not specify a value, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything keeps the current value.

%

Percent sign, specifying that scroll margins are percentages of the window height, rounded to the nearest line. This is useful if you frequently split the EVE main window into two or more windows.

example

The following command sets the scroll margins at two lines from the top and three lines from the bottom of the window:

Command: SET SCROLL MARGINS 2 3

SET TABS

format

SET TABS { AT *integer1* [*integer2* ...]
EVERY *integer*
INSERT
INVISIBLE
MOVEMENT
SPACES
VISIBLE }

description

Sets tab stops for the buffer (AT or EVERY), tab modes (INSERT, MOVEMENT, or SPACES), or the appearance of tab characters during editing (INVISIBLE or VISIBLE). Tab stops are buffer-specific settings; you can have different tab stops for each buffer. Changing the tab stops affects any tab characters already in the buffer. To find out the current tab stops of the buffer, use the SHOW command. If you want particular tab stops for all or most editing sessions, put the SET TABS AT or SET TABS EVERY command in your initialization file. When you invoke EVE using that initialization file, the setting then applies to the main (or first) buffer and to an EVE system buffer named \$DEFAULTS\$, so that each buffer you create has the same tab stops. (See Section 6.5.9.)

parameters

AT integer1 [integer2...]

The column or columns at which you want a tab stop in the current buffer. The new tab stops are applied to any tab characters already in the buffer. Enter the numbers in ascending order, separated by spaces. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

EVERY integer

An equal interval for all tab stops in the current buffer. The new tab stops are applied to any tab characters already in the buffer. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation. The default setting is EVERY 8 (that is, tab stops are set at columns 9, 17, 25, 33, and so on.)

INSERT

Default setting. Makes TAB insert a tab character, pushing the cursor and any existing text on the line to the next tab stop. The setting applies to all buffers.

MOVEMENT

Makes TAB move the cursor to the next tab stop, without inserting anything. The cursor stays on the current line and can move into the unused portion of the buffer, even if the cursor is set to bound. The setting applies to all buffers. This is useful for moving through tab-aligned text, such as tables and multicolumn lists.

SPACES

Makes TAB insert the appropriate number of spaces, instead of a tab character, pushing the cursor and any existing text on the line to the next tab stop. The setting applies in all buffers, but does not affect existing tabs—for example, does *not* convert tab characters to spaces. This is useful for editing text to be printed or displayed on different devices, because the spacing will be the same regardless of the tab stops set for the printer or display device.

VISIBLE

Makes tab characters visible—appearing as a small H_T (horizontal tab). The setting applies to all buffers. Visible tabs are an editing convenience only; the setting does *not* affect how tab characters appear when the text is printed.

INVISIBLE

Default setting. Makes tab characters invisible—appearing as white space. The setting applies to all buffers.

EVE-80 EVE Commands

SET TABS

example

The following example sets tab stops every 10 columns in the current buffer (11, 21, 31, and so on), changing any tab characters already in the buffer, and sets the tab mode to move the cursor to the next tab stop without inserting anything. Thus, pressing TAB moves the cursor 10 columns or less, to the next tab stop.

```
Command: SET TABS EVERY 10
Command: SET TABS MOVEMENT
```

SET WIDTH

format

SET WIDTH *integer*

description

Sets the width of the EVE screen layout to the number of columns you specify. This does not affect how many characters you can put on a line (which is determined by the right margin), but only how many characters are visible. The setting applies to all windows in EVE. When you end the editing session, your terminal setting is restored. If you want a particular width for all or most editing sessions, put the SET WIDTH command in your EVE initialization file. The width determines the default right margin, unless you specify otherwise by using the SET RIGHT MARGIN command. Thus, if you use a width of 120 columns, the default right margin is 119. To find out the current width, use the SHOW command. Also, the horizontal length of the status line indicates the width of the window.

SET WIDTH makes the right margin of the \$DEFAULTS\$ buffer one column less than the width. Buffers you create thereafter will have the same right margin as \$DEFAULTS\$. For example, the command SET WIDTH 132 makes the default right margin 131; the command SET WIDTH 80 makes the default right margin 79. This does not affect the right margin of other, existing buffers, but only buffers you create after a SET WIDTH command.

parameter

integer

The number of columns you want for the width of the display. If you specify a value greater than 80, EVE sets the terminal to 132-character mode, which uses a smaller video font. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation. The default width is

the same as your terminal setting (according to the DCL command SET TERMINAL)—typically, 80 columns.

Do not use a width greater than 80 on VT100-series terminals without the advanced video option (AVO).

example

The following commands set the width of the display to 132 columns and later restore it to 80 columns (which is typically the default setting):

```
Command: SET WIDTH 132
```

```
      .  
      .  
      .
```

```
Command: SET WIDTH 80
```

SET WILDCARD ULTRIX

format

SET WILDCARD ULTRIX

description

Enables ULTRIX-style wildcards for the WILDCARD FIND command. ULTRIX-style wildcards (sometimes called *regular expressions* or *meta-characters*) include the period (.) to match any single character on a line, the dollar sign (\$) to match end-of-line, and the circumflex (^) to match beginning-of-line. For a list of the available wildcards, use the SHOW WILDCARDS command. If you want ULTRIX-style wildcards for all or most editing sessions, put the SET WILDCARD ULTRIX command in your EVE initialization file. (See Section 6.5.9.) The default setting is VMS.

SET WILDCARD VMS

format

SET WILDCARD VMS

description

Enables VMS-style wildcards for the WILDCARD FIND command. (Default setting.) VMS-style wildcards (sometimes called *meta-characters*) include the percent sign (%) to match any single character on a line, the asterisk (*) to match any amount of text on a line, and the backslash and right angle bracket (\>) to match end-of-line. For a list of the available wildcards, use the SHOW WILDCARDS command.

SET WRAP

format

SET WRAP

description

Enables automatic wrapping in the current buffer, so that as you type at the end of a line, EVE starts a new line when your cursor goes past the right margin, without your having to press RETURN or use FILL commands. (Default setting.) The SET WRAP command does not by itself rewrap or reformat existing text. To reformat text, use FILL commands. To disable wrapping, use the SET NOWRAP command, so that lines can go past the right margin. This is useful for editing very long lines, such as multicolumn tables or program statements that are progressively indented. Note that wrapping is a buffer-specific setting; you can disable and enable wrapping for the current buffer, without affecting other buffers.

SHIFT LEFT

format

SHIFT LEFT *integer*

description

Shifts the current EVE window to the left by the number of columns you specify. You can use SHIFT LEFT commands only if you have used SHIFT RIGHT. Using SHIFT LEFT and SHIFT RIGHT commands lets you view the undisplayed portion of very wide text, such as lines 100 characters long, without having to change the width of the window or use 132-column mode. This does not shift text within the buffer, but shifts the window's horizontal position relative to the buffer. SHIFT LEFT and SHIFT RIGHT commands affect only the current window, if you are using two or more windows in EVE. To find out the current width and any shift, use the SHOW command.

parameter

integer

The number of columns you want to shift the window to the left. You cannot shift the window left past column 1. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

In the following example, you shift the current window five columns to the right, and then another five columns to the right. (Thus, column 11 of the buffer appears in the leftmost column of the screen.) You then shift the window back 10 columns to the left (so that column 1 of the buffer is in the leftmost column of the screen). Note that for each shift right, EVE tells you the cumulative shift.

```
Command: SHIFT RIGHT 5
Window now shifted right 5 columns.
Command: SHIFT RIGHT 5
Window now shifted right 10 columns.
      .
      .
      .
Command: SHIFT LEFT 10
Window now shifted right 0 columns.
```

SHIFT RIGHT

format

SHIFT RIGHT *integer*

description

Shifts the current EVE window to the right by the number of columns you specify. Using **SHIFT RIGHT** and **SHIFT LEFT** commands lets you view the undisplayed portion of very wide text, such as lines 100 characters long, without having to change the width of the window or use 132-column mode. This does not shift text within the buffer, but shifts the window's horizontal position relative to the buffer. **SHIFT RIGHT** and **SHIFT LEFT** commands affect only the current window, if you are using two or more windows in EVE. To find out the current width and any shift, use the **SHOW** command.

parameter

integer

The number of columns you want to shift the window to the right. If you do not specify a number, EVE prompts you for one. Pressing **RETURN** or **DO** at the prompt without typing anything cancels the operation.

EVE-84 EVE Commands

SHIFT RIGHT

example

In the following example, you shift the current window five columns to the right, and then another five columns to the right. (Thus, column 11 of the buffer appears in the leftmost column of the screen.) You then shift the window back 10 columns to the left (so that column 1 of the buffer is in the leftmost column of the screen). Note that for each shift right, EVE tells you the cumulative shift.

```
Command: SHIFT RIGHT 5
Window now shifted right 5 columns.
Command: SHIFT RIGHT 5
Window now shifted right 10 columns.
      .
      .
      .
Command: SHIFT LEFT 10
Window now shifted right 0 columns.
```

SHOW

format

SHOW

description

Shows the following information about the current buffer:

- Name of the buffer
- Input file for the buffer, if any
- Output file for the buffer, if any (typically the same as the input file)
- Whether the buffer has been modified
- Total number of lines in the buffer
- Margins, tab stops, and other buffer settings
- Window width and any shift
- Names of markers in the buffer, if any
- List of nondefault keymaps for the buffer, if any

Steps:

1. Use the **SHOW** command. The output appears in an EVE system buffer named Show in the current window.
2. If you created other buffers, EVE first shows information about the current buffer. To show information about your other buffers, if any, press DO. To return to the buffer you were editing, press any other key.

If you created only one buffer, press any key to return to that buffer.

SHOW BUFFERS

format

SHOW BUFFERS

description

Lists the buffers you have created and puts the cursor in the list so you can view or delete a buffer without having to type the buffer name. The list also tells you the total number of lines in each buffer, whether the buffer has been modified, and other information.

Steps:

1. Use the **SHOW BUFFERS** command. The output (the list of buffers you have created) appears in an EVE system buffer named Buffer List in the current window.
2. Put the cursor on the name of a buffer in the list. (To scroll through the list, you can press the up and down arrow keys or other cursor-movement keys.)
3. To view that buffer, use **SELECT**. (On DECwindows, you can click MB1 twice.) EVE then puts that buffer into the current window.

To delete that buffer, use **REMOVE** or **CUT**.

This is effectively the same as using the **BUFFER** or **DELETE BUFFER** commands respectively, without having to type the buffer name. For more information about deleting buffers, see the description of the **DELETE BUFFER** command.

SHOW DEFAULTS BUFFER

format

SHOW DEFAULTS BUFFER

description

Shows information about the \$DEFAULTS\$ buffer—margins, tab stops, direction, mode, maximum lines, and so on. The \$DEFAULTS\$ buffer is an EVE system buffer whose settings are used when you create new buffers. If you use an initialization file when you invoke EVE, commands in the initialization file for buffer settings apply to the \$DEFAULTS\$ buffer as well as to the main (or first) buffer, so that each buffer you create has the same setting (effectively, setting your own, private defaults). If your initialization file does not have commands for buffer settings, the EVE default settings are used. For more information about EVE default settings, see Section 6.5.9.

SHOW KEY

format

SHOW KEY [*key-name*]

description

Shows the definition of a key, telling you the command or keypad function bound to the key, if any. You can type the name of the key on the command line or let EVE prompt you to press the key.

parameter

key-name

The key you want to know about. You cannot abbreviate the key name. For more information about EVE key names, see Section 6.4.3. If you do not specify a key name, EVE prompts you to press the key you want to know about.

example

The following command shows the definition of GOLD-KP8 when you set the EDT keypad. In specifying control keys or GOLD key combinations, use a slash, dash, or underscore as a delimiter in the key name.

```
Command: SHOW KEY gold-kp8
GOLD/KP8 is defined as 'fill' in the EDT keypad.
```

SHOW SUMMARY

format

SHOW SUMMARY

description

Shows statistics and other information about EVE, such as the following:

- Version number of the software
- Current journal file specification, if any
- Current section file specification
- Total number of buffers (system- and user-created)
- Modules used in the current section file
- Other information about the EVE configuration

This information is useful for VAXTPU programming or in case you have to submit a software performance report (SPR).

SHOW SYSTEM BUFFERS

format

SHOW SYSTEM BUFFERS

description

Lists the buffers created by EVE and puts the cursor in the list so you can view a buffer without having to type its name. This makes it easy to view the Messages buffer (for example, to check compiler messages), the Insert Here buffer (to check what text you have removed or copied), or the \$RESTORE\$ buffer (to check what you last erased).

Steps:

1. Use the SHOW SYSTEM BUFFERS command. The output (the list of buffers EVE creates) appears in an EVE system buffer named Buffer List in the current window.
2. Put the cursor on the name of a buffer in the list, such as Messages. (To scroll through the list, you can press the up and down arrow keys or other cursor-movement keys.)
3. To view that buffer, use SELECT. (On DECwindows, you can click MB1 twice.) EVE then puts that buffer into the current window.

EVE-88 **EVE Commands**
SHOW SYSTEM BUFFERS

As a rule, do not delete system buffers or change their read/write status, because they may be required for some EVE commands to work. Some system buffers cannot be deleted and are marked permanent in the list.

SHOW WILDCARDS

format

SHOW WILDCARDS

description

Lists the wildcards available for the WILDCARD FIND command—either VMS or ULTRIX, depending on your setting.

Steps:

1. Use the SHOW WILDCARDS command. The output (the list of wildcards) appears in an EVE system buffer named Show in the current window.
2. To scroll through the list, you can press the up and down arrow keys or other cursor-movement keys.
3. To return to the buffer you were editing, press DO or use the RESET command.

SHRINK WINDOW

format

SHRINK WINDOW *integer*

description

Shrinks the current window by the number of lines you specify—if you are using more than one window in EVE. The lines are subtracted from the bottom unless the window is the bottommost window.

parameter

integer

The number of screen lines you want to subtract from the current window. The minimum size of a window is one line of text, one line for the status line, and on DECwindows, one line for the horizontal scroll bar. If the number you specify would shrink the window beyond these limits, EVE shrinks the window as much as possible. The maximum size depends on the size and type of terminal you are using. If you do not specify a number, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following commands form two windows, and then shrink the lower of the two windows by five lines:

```
Command: TWO WINDOWS  
Command: SHRINK WINDOW 5
```

SPAWN

format

SPAWN [*command-string*]

description

Spawns a subprocess, suspending, but not ending, your editing session. This lets you return to the DCL level or run another utility, such as MAIL, without having to end your editing session. Using SPAWN and ATTACH commands, in EVE and at the DCL level or in other utilities such as MAIL, lets you keep an editing session active throughout your VMS session (or login)—effectively making EVE a “kept” editor. This makes it faster to resume editing, but uses more system resources.

The SPAWN command is not supported if you invoke EVE using /DISPLAY=DECWINDOWS.

parameter

command-string

Optionally, the DCL command you want executed in the subprocess, such as a utility you want to invoke. Exiting from that utility ends the subprocess and resumes your editing session. If you do not specify a command string, EVE spawns a subprocess for DCL. To resume your editing session, use the LOGOUT command.

example

The following command spawns a subprocess for MAIL. Exiting from MAIL resumes your editing session.

```
Command: SPAWN mail  
MAIL>  
.  
.  
.  
MAIL> EXIT
```

SPELL

format

SPELL

description

Runs DECspell (if it is installed on your system) to check the currently selected text or the entire buffer.

The SPELL command is not supported if you invoke EVE using /DISPLAY=DECWINDOWS.

Steps:

1. Optionally, select the text you want to check. (See the description of the SELECT command.)

If you select less than a full line, EVE extends the select range to include the start and end of the line. If you do *not* select any text, SPELL checks the entire buffer.

2. Use the SPELL command.

EVE spawns a subprocess to run DECspell, and writes out the current buffer or select range to a temporary file in a system directory called SYS\$SCRATCH.

3. Use DECspell commands to correct your text.

After you make the corrections, exiting from DECspell resumes your editing session. EVE then replaces the buffer or range with the new version of the temporary file, containing any corrections, and deletes the temporary file.

NOTE: Do *not* use CTRL/Y while using SPELL. This deletes lines in the temporary output file, and therefore destroys the select range or current buffer.

SPLIT WINDOW

format

SPLIT WINDOW [*integer*]

description

Splits the current EVE window into two or more smaller windows. This lets you view different buffers at the same time or different parts of the same buffer. The cursor then appears in the bottommost new window. Each window has its own status line and displays the buffer you are currently editing. To put a different buffer into the window, use one of the following commands:

BUFFER
GET FILE or **OPEN**
NEW
NEXT BUFFER (if you have created more than one buffer)
OPEN SELECTED
SHOW BUFFERS (to choose a buffer from the list)

parameter

integer

Optionally, the number of windows you want to form. The default is 2 (effectively the same as using the TWO WINDOWS command). The maximum number of windows in EVE depends on the size and type of terminal you are using.

example

The following command splits the current window into three smaller windows, putting the cursor in the bottommost of the three windows:

Command: SPLIT WINDOW 3

START OF LINE

format

START OF LINE

VT300, VT200:

CTRL/H
GOLD-←

EVE-92 EVE Commands
START OF LINE

VT100:

CTRL/H
GOLD-←
BACKSPACE

description

Moves the cursor to the start of the current line. If you are already at the start of the line, the cursor does not move.

STORE TEXT

format

STORE TEXT

VT300, VT200:

GOLD-REMOVE

description

Copies a select range or found range, without removing it, so you can insert the text elsewhere (same as the COPY command).

Steps:

1. Use **SELECT**, **FIND**, or **WILDCARD FIND** to highlight the text you want to copy. (A select range takes precedence over a found range.)
2. Use the **STORE TEXT** command. The highlighting is canceled. A message tells you that the copying has been completed.
3. To paste the copied text elsewhere, use the **INSERT HERE** or **PASTE** command.

The copied text is stored either in the Insert Here buffer in EVE or in the DECwindows clipboard, depending on your setting, and replaces in that storage area whatever you previously copied or removed. The default setting is **NOCLIPBOARD**, which uses the Insert Here buffer. For more information, see the description of the **SET CLIPBOARD** command.

TAB

format

TAB

VT300, VT200:

TAB
CTRL/I

VT100:

TAB
CTRL/I

description

Inserts a tab at the current position, according to the current tab stops of the buffer and the tab modes. Note that redefining either the TAB key or CTRL/I affects the other as well.

TOP

format

TOP

VT300, VT200:

GOLD-↑

VT100:

GOLD-↑

description

Moves the cursor to the top of the current buffer—upper left corner—unless it is already there.

TPU

format

TPU *procedure-name*

description

Executes the VAXTPU procedure or statement you specify. You can execute a VAXTPU built-in procedure or a procedure you have compiled.

parameter

procedure-name

The VAXTPU procedure or statement you want to execute, including any required parameters or arguments. You cannot abbreviate the procedure name or statement, and cannot use wildcards. If you do not specify a procedure or statement, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command executes the COPY_TEXT built-in procedure to insert the current date:

```
Command: TPU COPY_TEXT (FAO ('!11%D', 0))
```

TWO WINDOWS

format

TWO WINDOWS

description

Splits the current EVE window into two smaller windows. (Same as the SPLIT WINDOW command, except that it does not take a parameter.) This lets you view different buffers at the same time or different parts of the same buffer. The cursor then appears in the new, lower window. Each window has its own status line and displays the buffer you are currently editing. To put a different buffer into the window, use one of the following commands:

- BUFFER
- GET FILE or OPEN
- NEW
- NEXT BUFFER (if you have created more than one buffer)
- OPEN SELECTED
- SHOW BUFFERS (to choose a buffer from the list)

You can repeat the TWO WINDOWS command to continue splitting windows. The maximum number of windows in EVE depends on the size and type of terminal you are using.

UNDEFINE KEY

format

UNDEFINE KEY *key-name*

description

Cancels the current definition of a key, if it was done with the DEFINE KEY or LEARN command. It does *not* cancel definitions done by a SET KEYPAD command or with the SET GOLD KEY command. You can type the key name on the command line or let EVE prompt you to press the key to be undefined. If the key you specify was previously defined by a SET KEYPAD command, its previous definition is restored, if the keypad setting is still in effect. You cannot undefine a key defined as DO unless there is another key defined as DO, and you cannot undefine or redefine the RETURN key or CTRL/M.

parameter

key-name

The key you want to undefine. You cannot abbreviate the key name. For more information about EVE key names, see Section 6.4.3. If you do not specify a key name, EVE prompts you to press the key to be undefined. Pressing the RETURN key or CTRL/M at the prompt cancels the operation, because those keys cannot be undefined.

example

In the following example, you set the EDT keypad, then redefine KP9 as CENTER LINE, overriding its EDT definition. The UNDEFINE KEY command then cancels that definition, restoring its EDT keypad definition (Append).

```
Command: SET KEYPAD EDT
Command: DEFINE KEY= kp9 center line
Command: UNDEFINE KEY kp9
```

UPPERCASE WORD

format

UPPERCASE WORD

description

Makes letters uppercase in a single word, select range, or found range. With a select range or found range, UPPERCASE WORD changes the letters in the range, starting with the first letter in the range (even if it is not the first letter of the word). A select range takes precedence over a found range. If there is no select range or found range, UPPERCASE WORD works on the current word. If you are between words, it works on the next word on the line.

WHAT LINE

format

WHAT LINE

description

Shows the current line number, total number of lines in the buffer, and percentage of that position in the buffer. This is useful if you want to know whether to insert a page break or simply to find out how many lines are in the buffer. To go to a particular line by number, use the LINE command. For example, the command LINE 10 puts the cursor at the start of line 10 in the current buffer.

example

The following example shows the output from the WHAT LINE command:

```
Command: WHAT LINE  
You are on line 35 of 45 (78%).
```

WILDCARD FIND

format

WILDCARD FIND *search-pattern*

VT300, VT200:

GOLD-FIND

description

Searches for a pattern of text using wildcards, literal text, or both. You can use VMS or ULTRIX wildcards, depending on your setting. The default setting is VMS. For a list of the available wildcards, use the SHOW WILDCARDS command.

If the string is found only in the opposite direction, EVE asks if you want to change the direction of the search and go there. Press RETURN if you want to go there, or type NO and press RETURN to end the search. If the string is not found, the cursor does not move.

EVE highlights the found text (video bold), with the cursor at the beginning of the string. If there is no select range, you can use COPY, FILL, REMOVE, UPPERCASE WORD, or other commands that work on a range of text. (If there is a select range, the operation works on the selected text, which may not include the found range.) To cancel the highlighting, move the cursor off the found range or use the RESET command. To find another occurrence of the same string, use the FIND NEXT command or press FIND *twice*.

parameter

search-string

The pattern of text you want to find, using wildcards, literal text, or both. WILDCARD FIND follows the same rules as the FIND command for case sensitivity and white space—unless specified otherwise by a wildcard. If you do not specify a string, EVE prompts you for one. Pressing RETURN or DO at the prompt without typing anything cancels the operation.

example

The following command, using VMS wildcards, finds a string like *bet* or *But* at the end of a line. The equivalent using ULTRIX wildcards is *b.t\$*.

Command: WILDCARD FIND b*t\>

WRITE FILE

format

WRITE FILE *output-filespec*

description

Writes out (saves) the current buffer, without ending the editing session. The buffer is written out to the file you specify or to the output file associated with the buffer. (Similar to the SAVE FILE or SAVE FILE AS command.) If you do not specify a file on the command line, EVE writes out the buffer using the output file associated with the buffer. Typically, this is the same as the file specified when you invoked EVE or when you used the GET FILE, OPEN, or OPEN SELECTED command. If there is no file associated with the buffer—for example, if you invoked EVE without specifying a file, or if you created the buffer with the BUFFER or NEW command, or if you are writing out an EVE system buffer—then EVE prompts you to type a file name. Specifying an output file does not change the buffer name, but does associate that file with the buffer for later SAVE FILE or WRITE FILE commands or for exiting (except with system buffers). To check the output file specification of the buffer, use the SHOW command.

parameter

output-filespec

Optionally, the output file you want to create for saving the contents of the current buffer. If you do not specify a file, EVE uses the file associated with the buffer. If there is no file associated with the buffer, EVE prompts you to type a file name. In such a case, simply pressing RETURN or DO at the prompt without specifying a file cancels the operation.

example

In the following example, you are editing a file called ROUGH.DAT and write out (saves) your edits as a file called FINAL.DAT, rather than as a new version of ROUGH.DAT:

```
Command: GET FILE rough.dat
.
.
.
Command: WRITE FILE final.txt
38 lines written to DISK$1:[GEOFF]FINAL.TXT;1
```

Mail Utility

You can use the VMS Mail Utility (MAIL) to send messages to other users on your system or on any other computer that is connected to your system by means of DECnet-VAX. You can also read, file, forward, delete, print, and reply to messages that other users send to you.

format

MAIL [*filespec*] [*recipient-name*]

parameters

filespec

Specifies the name of the file to be mailed.

recipient-name

Specifies the name of a user (or users) or a distribution list to which the file is mailed.

When you specify a list of users, separate each name by a comma.

When you specify a distribution list, precede the name of the list with an at sign (@) and enclose both the at sign and the name in quotation marks, as the following example shows:

```
$ MAIL JOKES.DAT "@LIST"
```

usage summary

To use MAIL interactively, enter the following command in response to the DCL prompt:

```
$ MAIL
```

The Mail Utility responds with the following prompt:

```
MAIL>
```

Once MAIL has been invoked, you can enter any of the MAIL commands. To exit from MAIL, enter the EXIT command at the MAIL> prompt.

```
MAIL> EXIT
```

You can also exit from MAIL by pressing CTRL/Z or using the QUIT command.

MAIL-2 MAIL /EDIT

MAIL Qualifiers

You can supply the /EDIT, /PERSONAL_NAME, /SELF, and /SUBJECT qualifiers when invoking MAIL.

/EDIT

Sets the default to /EDIT for the SEND and REPLY commands and allows you to edit your mail messages.

format

MAIL/EDIT [(keyword[=option], ...)]

qualifier values

keyword

Allowed keywords are FORWARD, REPLY, and SEND.

option

The EXTRACT option can be used with the REPLY keyword.

example

```
$ MAIL/EDIT
MAIL> SEND
To: EARTH::MAX
Subj: Experiment
```

```
.
.
.
```

```
[EOB]
*exit
MAIL>
```

This example shows how to use the /EDIT qualifier with the MAIL command enabling you to create and edit a new message. Press CTRL/Z to return to the line-editing prompt (*). Type EXIT to send the message.

/PERSONAL_NAME=name

/PERSONAL_NAME=name

Specifies the personal name to be used when sending a message. This qualifier does not override the default personal name specified by the SET PERSONAL_NAME command; the personal name is only changed for the current message.

format

MAIL/PERSONAL_NAME =*name file-name recipient-name*
MAIL/NOPERSONAL_NAME

parameter values***name***

Personal name to be used. Use quotes around the personal name to include more than one word or to print in lowercase letters.

file-name

Name of file to be sent.

recipient-name

Names of users to whom the message is sent.

example

```
$ MAIL/PERSONAL_NAME ="Joe M." test.dat smith
```

This example shows the user's personal name defined as *Joe M.* in the current message containing the file TEST.DAT sent to user SMITH.

/SELF

Sends a copy of the message containing the file specification on the command line back to you as well as to other users.

format

MAIL/SELF *filespec recipient-name*

parameter values***filespec***

Name of file to be sent.

recipient-name

Names of users to whom the message is sent.

MAIL-4 MAIL
/SELF

example

\$ MAIL/SELF experiments.dat smith,jones

This example shows how to use the /SELF qualifier to send a copy of the message containing the file named EXPERIMENTS.DAT back to you and to users SMITH and JONES.

/SUBJECT

Specifies the subject of the message for the heading. If the text consists of more than one word, enclose the text in quotation marks.

format

MAIL/SUBJECT="text" filespec recipient-name

parameter values

filespec

Name of file to be sent.

recipient-name

Names of users to whom the message is sent.

example

\$ MAIL/SUBJECT="Life in the Big City" newfile.txt JOHNSON

This example shows how to use the /SUBJECT qualifier to send a file named NEWFILE.TXT with a subject heading of "Life in the Big City." Use quotation marks around the subject heading to include more than one word or to print in lowercase letters.

MAIL Commands

To enter MAIL commands, first invoke MAIL at the DCL prompt (\$) and then enter the MAIL commands at the MAIL> prompt. These commands can be abbreviated to unique, shorter forms (usually as short as one letter). Note that D is the short form of DELETE (not DIRECTORY) and R is the short form of REPLY (not READ).

MAIL provides commands that enable you to do the following:

- Read and organize mail messages.
- Exchange mail messages with other users.
- Remove mail messages.
- Tailor the Mail Utility.
- Exit from MAIL or transfer control to another process while still in MAIL.
- Make hardcopies of mail messages.

BACK

Displays the message preceding the current or last-read message when the last command issued was READ. When the last command issued was DIRECTORY, the BACK command displays the preceding screen of the directory listing.

format

BACK

qualifier

/EDIT

Indicates that the default editor is invoked. You can use the editor to easily peruse the previous message. When you are done, enter the QUIT command. You will see the MAIL> prompt. If you decide to edit the message and want to keep a copy of the newly edited message, enter the appropriate command to exit from your editor (use the EXIT command with the EDT editor) and supply a file name.

COPY

Copies a message to another folder without deleting it from the current folder. If the specified folder does not exist, it is created.

If you want to copy a message to a sequential file (outside of MAIL) instead of to a mail file, use the EXTRACT command.

If you decide (after entering the COPY command, pressing RETURN, and supplying a folder name at the prompt, but before pressing RETURN again) that you do not want to copy the message, press CTRL/C. CTRL/C aborts the operation and keeps you within MAIL.

format

COPY *foldername* [*filename*]

parameters

foldername

Indicates the name of the folder to which the message is to be copied. If the specified folder does not exist (and you have not entered the qualifier /NOCONFIRM), you are asked whether you want to create it. If you respond with Y, the new folder is created. A folder name can be 1 to 39 characters in length. Valid characters for folder names are A through Z, a through z, dollar sign (\$), underscore (_), and 0 through 9.

filename

Indicates the name of the mail file to which the message is to be copied. If the specified mail file does not exist, it is created. If a file name is omitted, the message is copied to the specified folder in the current file.

qualifiers

/ALL

Indicates that all of the currently selected messages are to be copied to another message folder. You select a folder by entering the SELECT command followed by the name of the folder. (See the SELECT command for more information.) If the /ALL qualifier is omitted, only the current message is copied.

/CONFIRM

/NOCONFIRM

Determines whether you will be queried about creating a new folder or file. The default is /CONFIRM.

example

MAIL> DIRECTORY

			MAIL
#	From	Date	Subject
1	MARK	29-NOV-1988	Upcoming Meetings
2	GRIM	3-DEC-1988	Horror Stories
3	KATE	7-DEC-1988	Getting a Court for Fridays

MAIL> 2

MAIL> COPY

_Folder: TALES

_File: **RET**

MAIL> SELECT TALES

%MAIL-I-SELECTED, 1 message selected

MAIL> DIRECTORY

			TALES
#	From	Date	Subject
1	GRIM	3-DEC-1988	Horror Stories

This example shows how to put a copy of a mail message (from a user named GRIM) into another folder (TALES) and how to move to that folder to see the copy of the mail message.

DELETE

Deletes either the message you are currently reading or the message you just read and moves it to the WASTEBASKET folder. When you enter the EXIT or PURGE commands, your WASTEBASKET folder empties automatically.

To recover a message accidentally deleted (while it is still in the WASTEBASKET folder), select the WASTEBASKET folder, read the desired message, and move it to another folder.

Usually you delete only one mail message at a time, but you may also delete several mail messages using one DELETE command. You may specify a range or a list of messages to be deleted.

format

DELETE [*message-number*]

parameter

message-number

Deletes the message specified by its number, or deletes a range or list of messages.

MAIL-8 MAIL DELETE

qualifier

/ALL

Deletes all the currently selected messages. You select a folder by entering the SELECT command followed by the name of the folder. (See the SELECT command for more information.)

example

```
MAIL> DELETE 1,3,5-7,9:11
MAIL>
```

This example shows how to delete mail messages 1, 3, 5, 6, 7, 9, 10, and 11. The hyphen and colon are used to designate a range of numbers.

DIRECTORY

Displays a list of the messages in the current mail file, including message number, sender's name, date, and subject.

You create a new set of selected messages every time you use the following qualifiers:

- /BEFORE**
- /CC_SUBSTRING**
- /NEW**
- /SINCE**
- /MARKED**
- /FROM_SUBSTRING**
- /TO_SUBSTRING**
- /SUBJECT_SUBSTRING**

format

DIRECTORY [*foldername*]

parameter

foldername

Specifies the name of the folder. If you omit this parameter and you have already specified a folder, messages from that folder are displayed. If you have not selected a folder, messages from the NEWMAIL folder are displayed. If the NEWMAIL folder does not exist, messages from the MAIL folder are displayed.

qualifiers

/BEFORE=date

Displays a listing of all the mail messages received before the specified date. If no date is specified, a listing of all the mail messages received before the current day ("today") is displayed.

/CC_SUBSTRING=text

Selects messages containing "text" in the CC field of the message.

/EDIT

Invokes the editor using the output of the DIRECTORY command as input to the editor. Enables you to find messages easily by scrolling through the folders or searching text.

/FROM_SUBSTRING=text

Selects messages containing "text" in the FROM field of the message.

/FOLDER

Displays a listing of all the folders contained in the current mail file.

/FULL

Displays the number of records in the message and whether you have replied to the message. External message identification numbers (for messages larger than 3 blocks) are also displayed.

/MARKED

/NOMARKED

Selects messages that have been marked. The /NOMARKED qualifier selects messages that are not marked.

/NEW

Displays a listing of any new (unread) mail messages. When there are no unread messages, MAIL displays the message "No new messages."

/REPLIED

/NOREPLIED

Selects messages that have been replied to via the REPLY command. The /NOREPLIED qualifier selects messages that have not been replied to.

/SINCE=date

Displays a listing of all the mail messages received on or after the specified date. If no date is specified, a listing of all the mail messages received on the current day ("today") is displayed.

/START=start-point

Indicates the first message number you want to display. For example, to display all the messages beginning with number three, enter the command line DIRECTORY/START=3. Use the /START qualifier with the /FOLDER qualifier to indicate the first folder name you want to display.

MAIL-10 MAIL DIRECTORY

For example, to display all the folder names alphabetically following PLEAT, enter the command line DIRECTORY/START=PLEAT/FOLDER.

/SUBJECT_SUBSTRING=text

Selects messages containing "text" in the SUBJECT field of the message.

/TO_SUBSTRING=text

Selects messages containing "text" in the TO field of the message.

example

```
MAIL> DIRECTORY/SUBJECT_SUBSTRING= POUND
```

```
MAIL
# From Date Subject
1 BILL 13-APR-1988 The Pound
```

This example shows how to use the /SUBJECT_SUBSTRING qualifier with the DIRECTORY command to find messages that contain the substring POUND.

```
MAIL> DIRECTORY/FOLDER
```

```
Listing of folders in DISK$:[BACON]MAIL.MAI;1
```

```
Press CTRL/C to cancel listing
```

```
MAIL NEW_HIRES
PROJECTS SALES_LEADS
```

This example shows how to display a listing of all the folders in the current mail file.

EXIT

Allows you to exit from MAIL. You can also exit from MAIL by pressing CTRL/Z.

format

```
EXIT
```

EXTRACT

Places a copy of the current message into a sequential file. If you want to copy a mail message to a folder in an indexed sequential mail file, use either the COPY, FILE, or MOVE command.

format

```
EXTRACT filespec
```


parameter

filespec

Specifies the name of the output file to which the message is copied. The default file type is TXT. By default, the device and directory matches your current default device and directory.

qualifiers

/ALL

Copies all the currently selected messages to the specified file. Each message is separated by a form feed.

/APPEND

Adds the selected message to the end of the specified file. If the file does not exist, it is created. When you do not specify /APPEND, MAIL creates a new sequential file.

/MAIL

Specifies that the output file be a sequential mail file with a default file type of MAI and a protection code of (S:RW,O:RW,G,W). By default, the protection codes of the device and directory match those of your mail file directory. Like /APPEND, /MAIL adds the selected message to the end of the specified file.

/NOHEADER

Removes the header information (To: CC: From: Subject:) from the mail message.

example

```
MAIL> EXTRACT/ALL/NOHEADER
  _File: OUTER.DAT
%MAIL-I-CREATED, DISK$MEGAWORK:[CROWN]OUTER.DAT;1 created
MAIL>
```

This example shows how to place a copy of all the messages in the currently selected folder into a sequential file called OUTER.DAT. The /NOHEADER qualifier prevents the header information from being copied.

FILE

Moves the current message to the specified folder. You can use the FILE command and the MOVE command interchangeably because they work the same way. (Note, however, that the FILE command deletes the message from the original folder, unlike the COPY command, which leaves a copy.)

If (after entering the FILE command, pressing RETURN, and supplying a folder name at the prompt, but before pressing RETURN again) you decide that you do not want to file the message, press CTRL/C. CTRL/C aborts the operation and keeps you within MAIL.

format

FILE *foldername* [*filename*]

parameters

foldername

Indicates the name of the folder to which the current message is to be moved. If the specified folder does not exist, you are asked whether you want to create it. If you respond with Y, the new folder is created.

A folder name can be 1 to 39 characters in length. Valid characters for folder names are A through Z, a through z, dollar sign (\$), underscore (_), and 0 through 9.

filename

Indicates the name of the mail file to which the current message is to be moved. If the file name is omitted, the message is moved to the specified folder in the current file.

qualifiers

/ALL

Moves all the messages in the current folder to the specified folder.

/CONFIRM

/NOCONFIRM

Determines whether you are queried about creating a new folder or file. The default is /CONFIRM.

example

```
MAIL> 2
MAIL> FILE ①
  _Folder: WINNERS ②
  _FILE:  ③
Folder WINNERS does not exist.
Do you want to create it (Y/N, default is N)? y
%MAIL-I-NEWFOLDER, folder WINNERS created
MAIL> SELECT WINNERS ④
%MAIL-I-SELECTED, 1 message selected
MAIL> DIRECTORY ⑤

#   From      Date           Subject
1   BURK      18-APR-1988   Early American Art

                                WINNERS
MAIL>
```

- ① Enter the FILE command to move the current message to a new folder.
- ② Specify a name for the new folder.
- ③ Press RETURN to retain the default file.
- ④ To move to the new folder, enter the SELECT command followed by the name of the new folder (WINNERS).
- ⑤ Enter the DIRECTORY command to see the transferred message in the newly created folder (WINNERS).

This example shows how to FILE a message in a new folder named WINNERS.

FORWARD

Sends a copy of the message you are currently reading (or have just read) to one or more users. MAIL prompts you for the name of the user or users to whom you want to forward the message.

If you change your mind about forwarding a message after you have already entered the FORWARD command, press CTRL/C to abort the message. The MAIL> prompt is displayed.

**MAIL-14 MAIL
FORWARD**

format

FORWARD

qualifiers

/CC_PROMPT

/NOCC_PROMPT

Prompts for CC: line in the mail header. Overrides the SET CC_PROMPT command.

/EDIT

Determines whether the default editor is invoked to edit the message you are forwarding.

/NOHEADER

Enables you to forward a message without the original header information supplied from the user that sent it. The default is /HEADER.

/PERSONAL_NAME=name

/NOPERSONAL_NAME

Specifies the personal name to be used when forwarding the message. Overrides the default personal name specified with the SET PERSONAL_NAME command for this message only. The /NOPERSONAL_NAME qualifier sends a message with a null personal name field.

/SELF

/NOSELF

Specifies that a copy of the forwarded message is to be sent to you. Overrides the SET COPY_SELF command.

/SUBJECT="subject-text"

Prompts for the subject of the mail message to be sent.

example

MAIL> 3

From: PRESTON
To: MARLEY
Subj: Snakes

Beasts, under the earth, crawling...

MAIL> FORWARD/NOHEADER

To: SOUND::BURTON
Subj: Snakes Again

.
.
.

```
MAIL> READ
From:      MARLEY
To:        SOUND::BURTON
Subj:      Snakes Again
```

Beasts, under the earth, crawling...

This example shows how to forward a message to a user (SOUND::BURTON) without the original header information (From: PRESTON, To: MARLEY, Subj: Snakes).

HELP

Enables you to obtain information about the Mail Utility.

To obtain information about all of the MAIL commands, enter the following command:

```
MAIL> HELP
```

To obtain information about individual commands or topics, enter HELP followed by the command or topic name.

format

```
HELP [topic]
```

parameter

topic

Indicates a topic about which you want information. To display the list of available topics, enter the HELP command at the MAIL> prompt.

MARK

The MARK command sets a flag in the message header setting the current or message-identification message as marked. Marked messages are displayed with an asterisk (*) in the left-hand column of the directory listing. A marked message can serve as a reminder of important information. The /ALL qualifier sets all currently selected messages as unmarked.

The UNMARK command clears a flag in the message header setting the current or message-id message as unmarked (asterisk is deleted).

MAIL-16 MAIL MARK

format

MARK [/ALL] [message-number]
UNMARK [/ALL] [message-number]

parameter

message-number

Indicates the message number to be marked or unmarked.

qualifier

/ALL

Sets all currently selected messages as marked.

example

```
MAIL> DIR MISC
#           From           Date           Subject
1          MARS::SMITH     13-AUG-1988   Training Information
2          JUPITER::COLLINS 22-AUG-1988   Ideas
3          JUPITER::PETERS  24-AUG-1988   Meeting
MAIL> MARK 2,3
MAIL> DIR
#           From           Date           Subject
1          MARS::SMITH     13-AUG-1988   Training Information
* 2        JUPITER::COLLINS 22-AUG-1988   Ideas
* 3        JUPITER::PETERS  24-AUG-1988   Meeting
```

In this example, messages 2 and 3 in folder MISC are marked with an asterisk.

MOVE

The MOVE command is synonymous with the FILE command.

NEXT

Skips to the next message and displays it. This command is useful if, while reading through your messages, you encounter a long message that you would like to skip over.

format

NEXT

qualifier

/EDIT

Invokes your default editor. You can use your editor to peruse the next message. When you are done, enter the QUIT command. You see the MAIL> prompt. If you decide to edit the message and want to keep a copy of the newly edited message, enter the appropriate command to exit from your editor (enter the EXIT command with the EDT editor) and supply a file name.

PRINT

Adds a copy of the message you are currently reading to the print queue. The files created by the PRINT command are not actually released to the print queue until you exit from MAIL, so that multiple messages are concatenated into one print job (unless the /NOW or /PRINT qualifier is specified).

format

PRINT

qualifiers

/AFTER=time

Requests that the job not be printed until a specific time of day. You can specify either absolute or delta time.

/ALL

Indicates that all the currently selected messages be printed.

/BURST=keyword

/NOBURST

Controls whether a burst page is printed preceding a message. The /BURST qualifier can take either of two keywords: ALL or ONE. The ALL keyword indicates that each file in the job is to be preceded by a burst page and flag page. The ONE keyword indicates that a burst page applies only to the first copy of the first file in the job.

/CANCEL

Cancels all messages that have been queued for printing during this session.

/COPIES=n

Indicates the number of copies of the print job to be printed.

**MAIL-18 MAIL
PRINT**

***/FEED
/NOFEED***

Controls whether the PRINT command automatically inserts form feeds when it nears the end of a page. /FEED is the default.

***/FLAG=keyword
/NOFLAG***

Controls whether a flag page is printed preceding a message. The /FLAG qualifier can take either of two keywords: ALL or ONE. The ALL keyword indicates that each file in the job is preceded by a flag page. The ONE keyword indicates that a flag page applies only to the first copy of the first file in the job.

/FORM=form-name

Specifies the name or number of the form that you want for the print job. Enter the DCL command SHOW QUEUE/FORM to list the available forms.

***/HOLD
/NOHOLD***

Controls whether the message is available for print immediately. The print job is not released for actual printing until you use the DCL command SET QUEUE/ENTRY/RELEASE to release it.

/NAME=job-name

Defines the name string to identify the job.

/NOTIFY

Indicates that you are to be notified by a broadcast message when the file or files have been printed. /NONOTIFY is the default.

/NOW

Sends all messages that have been queued for printing with the PRINT command during this session to the printer. Allows the job to print without exiting mail. This qualifier is synonymous with the /PRINT qualifier.

/PARAMETERS=(parameter[, ...])

Specifies from one to eight optional parameters to be passed to the job.

/PRINT

The /PRINT qualifier is synonymous with the /NOW qualifier.

/QUEUE=queue-name

The name of the queue to which the message is to be sent. If the /QUEUE qualifier is not specified, the message is queued to the SYS\$PRINT printer. If you enter the PRINT command more than once and specify a different queue name, any previously queued messages are released to that print queue.

/SPACE
/NOSPACE

Controls whether output is double-spaced.

/TRAILER=keyword
/NOTRAILER

Controls whether a trailer page is printed at the end of the message. The **/TRAILER** qualifier can take either of two keywords: **ALL** or **ONE**. The **ALL** keyword indicates that each file in the job is preceded by a trailer page. The **ONE** keyword indicates that a trailer page applies only to the last copy of the last file in the job.

example

```
MAIL> 5
MAIL> PRINT/QUEUE=LMNO
MAIL> EXIT
  Job MAIL (queue LMNO_PRINT, entry 333) started on LMNO_PRINT
$
```

This example shows how to add message number 5 to queue LMNO_PRINT.

READ

Displays your messages.

The **READ** command can be entered with or without parameters. Pressing the **RETURN** key is the same as entering the **READ** command without parameters. If you press **RETURN** immediately after **MAIL** is invoked, **MAIL** displays the first unread message in your **NEWMAIL** folder. If all messages have been read or you have no new messages, **MAIL** displays the first message in the **MAIL** folder. Each time you enter the **READ** command without parameters or press **RETURN**, **MAIL** displays the next message.

If a new message arrives while you are in **MAIL**, you can enter **READ/NEW** to read the message, and then return to the previous **MAIL** activity.

You create a new set of selected messages every time you use the following qualifiers:

/BEFORE
/CC_SUBSTRING
/NEW
/SINCE
/MARKED

MAIL-20 MAIL
READ

/FROM_SUBSTRING
/TO_SUBSTRING
/SUBJECT_SUBSTRING

format

READ [*foldername*] [*message-number*]

parameters

foldername

Specifies the name of the folder. If you omit this parameter and you have already specified a folder, messages from that folder are displayed. If you have not selected a folder, messages from the NEWMAIL folder are displayed. If the NEWMAIL folder does not exist, messages from the MAIL folder are displayed.

message-number

Indicates the number of the message to be read. The message number represents the position of a message in a folder. If you specify a number greater than the number of messages in the folder, MAIL displays the last message in the folder. Therefore, to read the latest message in a folder, specify a large message number or enter the LAST command.

qualifiers

/BEFORE=date

Displays mail messages received before the specified date. If no date is specified, all the mail messages received before the current day ("today") are displayed.

/CC_SUBSTRING=text

Selects messages containing "text" in the CC field of the message.

/EDIT

Invokes the default editor. You can use the editor to easily peruse the next message. When you are done, enter the QUIT command and return to the MAIL> prompt. If you decide to edit the message and want to keep a copy of the newly edited message, enter the appropriate command to exit from your editor (use the EXIT command with the EDT editor) and supply a file name.

/FROM_SUBSTRING=text

Selects messages containing "text" in the FROM field of the message.

/MARKED

/NOMARKED

Selects messages that have been marked. The /NOMARKED qualifier selects messages that are not marked.

/NEW

Displays new mail messages received while you are in MAIL. If there are no new messages, the message "No new messages" is displayed.

/REPLIED

/NOREPLIED

Selects messages that have been replied to via the REPLY command. The /NOREPLIED qualifier selects messages that have not been replied to.

/SINCE=date

Displays mail messages received on or after the specified date. If no date is specified, all the mail messages received after the current day ("today") are displayed.

/SUBJECT_SUBSTRING=text

Selects messages containing "text" in the SUBJECT field of the message.

/TO_SUBSTRING=text

Selects messages containing "text" in the TO field of the message.

example

```
MAIL> READ/SUBJECT_SUBSTRING= MEETINGS
```

```
MAIL
```

```
# From      Date           Subject
1  BILL     16-APR-1988   Future Meetings
```

This example shows how to use the /SUBJECT_SUBSTRING qualifier with the READ command to find messages that contain the substring MEETINGS.

REPLY

The REPLY command is synonymous with the ANSWER command.

SEARCH

Searches the currently selected folder for the message containing the first occurrence of the specified text string.

format

```
SEARCH search-string
```

MAIL-22 MAIL SEARCH

parameter

[search-string]

Indicates the text string that MAIL searches for in the currently selected messages. The search starts from the beginning of the messages in the current folder. If **search-string** is not specified, a search is made for the previously specified string, starting after the message you are currently reading (or have just read).

example

```
MAIL> SEARCH "under the"
```

```
From:    BURT  
To:      ANTON  
Subj:    Coal Mines
```

```
They commute under the earth...
```

```
MAIL>
```

This example shows how to search for the string *under the*.

SELECT

Establishes a set of messages that you can manipulate. You can copy or move selected messages from one folder to another; or you can read and delete, or search and extract, a set of messages. After you select a set of messages, you can use the following commands to affect them:

```
COPY  
DELETE  
DIRECTORY  
EXTRACT  
FILE  
MOVE  
READ  
SEARCH
```

You can also use the SELECT command to move from one folder to another. If you select a folder that does not exist, MAIL displays the following message:

```
%MAIL-E-NOTEXIST, folder "foldername" does not exist
```

format

```
SELECT [foldername]
```

parameter

foldername

Specifies the name of the folder. If you omit this parameter and have already specified a folder, messages from that folder are selected. If you have not specified a folder, messages from the NEWMAIL folder are selected. If the NEWMAIL folder does not exist, messages from the MAIL folder are selected.

qualifiers

/BEFORE=date

Indicates that messages dated before the specified date be selected. If no date is specified, all the messages received before the current day ("today") are selected.

/CC_SUBSTRING=text

Selects messages containing "text" in the CC field of the message.

/FROM_SUBSTRING=text

Selects messages containing "text" in the FROM field of the message.

/MARKED

/NOMARKED

Selects messages that have been marked. The /NOMARKED qualifier selects messages that are not marked.

/NEW

Indicates that new (unread) messages be selected. When a mail file other than your default mail file is open, MAIL closes the file and opens your default mail file.

/REPLIED

/NOREPLIED

Selects messages that have been replied to via the REPLY command. The /NOREPLIED qualifier selects messages that have not been replied to.

/SINCE=date

Indicates that messages dated after the specified date be selected. If no date is specified, all the messages received on the current day ("today") are selected.

/SUBJECT_SUBSTRING=text

Selects messages containing "text" in the SUBJECT field of the message.

/TO_SUBSTRING=text

Selects messages containing "text" in the TO field of the message.

MAIL-24 MAIL SELECT

example

```
MAIL> DIRECTORY/FOLDERS ①
Listing of folders in DISK$APEX:[HARRIS]MAIL.MAI;1
Press CTRL/C to cancel listing
```

```
MAIL          NEWMAIL
WASTEBASKET   JUNK
COURSES
```

```
MAIL> SELECT WASTEBASKET ②
%MAIL-I-SELECTED, 3 messages selected
MAIL> DIRECTORY ③
```

```
                                WASTEBASKET
#   From      Date              Subject
1   MORRIS    19-APR-1988      Venus Fly Traps
2   MORRIS    21-APR-1988      The Aloe
3   BURT      22-APR-1988      Scales
```

- ① Enter the DIRECTORY/FOLDERS command to display all currently existing folders.
- ② Enter the SELECT command to move to the WASTEBASKET folder.
- ③ Enter the DIRECTORY command to display the contents of the WASTEBASKET folder.

This example shows how to use the SELECT command to move from the MAIL folder to the WASTEBASKET folder.

SEND

Sends a message to one or more other users. You can use the SEND command and the MAIL command interchangeably because they work the same way.

MAIL prompts you first for the name of the user or users to receive the message. You reply with the user names or with the file names of distribution lists, in the following format:

```
[[nodename::]username,...] [,] [@listname[,...]]
```

If you have entered the SET CC_PROMPT command, you can specify names of users to receive carbon copies of the message at the CC: prompt.

Next, MAIL prompts you for the subject of the mail. To avoid the Subj: prompt, specify the /SUBJECT qualifier with the SEND command.

You can include a file specification with the SEND command. If you do, the text in that file is sent to the specified users. If you do not specify a file, MAIL prompts you for the text of your message.

Enter the message you want to send, then press CTRL/Z. Note that, once you have typed a line and pressed RETURN, there is no way to edit it. Using the /EDIT qualifier enables you to edit the entire message before you send it. The /LAST qualifier enables you to send the last message. The /LAST qualifier, used with the /EDIT qualifier, enables you to edit the last message you sent. If you decide not to send a message you are typing but want to stay within the Mail Utility, press CTRL/C to abort the message. You then receive the MAIL> prompt. CTRL/Y exits you from MAIL.

format

SEND [*filespec*]

parameter

filespec

Indicates the name of the file to be sent.

qualifiers

/CC_PROMPT

/NOCC_PROMPT

Prompts for CC: line in the mail header. Overrides the SET CC_PROMPT command.

/EDIT

/NOEDIT

Determines whether the default editor is invoked to edit the message you are sending. The /NOEDIT qualifier overrides the SEND/EDIT default if you entered the DCL command MAIL/EDIT.

If you are interrupted while editing a mail message, a journal file is created containing your edits. To recover your edits, enter the following command line. (You may substitute another word of your choice for the word OOPS.)

```
$ EDIT/RECOVER/JOURNAL=SYS$SCRATCH:MAIL.JOU SYS$SCRATCH:OOPS.TMP
```

The editor is invoked displaying the text of the message you were editing. After you exit from the editor, you can mail the file (in this case, OOPS.TMP) by using the MAIL command SEND, as follows:

```
MAIL> SEND OOPS.TMP
```

```
To: MCNALLY
```

```
Subject: Vacationing in Venice
```

**MAIL-26 MAIL
SEND**

/LAST

Specifies that the last message be used as the text for the message you are currently sending. You can use */LAST* with the */EDIT* qualifier to edit the message before sending it. Press CTRL/C and enter the following command:

```
MAIL> SEND/LAST/EDIT
```

This command invokes the editor and allows you to edit the last message. Send the revised message by entering the *EXIT* command.

/PERSONAL_NAME=name

/NOPERSONAL_NAME

Specifies the personal name to be used when sending this message. The */NOPERSONAL_NAME* qualifier sends a message with a null personal name field.

/SELF

/NOSELF

Determines whether *MAIL* sends a copy of the message you are sending back to you. The */NOSELF* qualifier overrides the *SET COPY_SELF SEND* command.

/SUBJECT="subject-text"

Specifies the subject of the mail message to be sent.

example

```
MAIL> SEND
To: FLIGHT::WRIGHT
Subj: Meeting
Enter your message below. Press CTRL/Z when complete, CTRL/C to quit:
We will have our meeting on Monday, August 31st, as scheduled.
Please make sure you are prompt.
```

.
.
.

CTRL/C

```
% MAIL_E_SENDA BORT, no message sent
MAIL> SEND/LAST/EDIT
To: FLIGHT::WRIGHT
Subj: Meeting date correction
We will have our meeting on Friday, September 4th, as scheduled.
Please make sure you are prompt.
```

.
.
.

[EOB]
CTRL/Z
*EXIT
MAIL>

This example shows how to edit the last message before sending it to user WRIGHT on node FLIGHT. To make a change in text, enter CTRL/C and invoke the editor by entering the SEND/LAST/EDIT command. Edit the message you were in the process of entering, and send it by entering the EXIT command.

SET/SHOW CC_PROMPT

Sets the default for determining whether the carbon copy (CC:) prompt appears when sending a message.

format

```
SET CC_PROMPT
SET NOCC_PROMPT
SHOW CC_PROMPT
```

example

```
MAIL> SET CC_PROMPT
MAIL> SEND
To: Smith
CC: Jones
Subject:
```

This example shows how to set the carbon copy prompt. A copy of the message is sent to JONES.

SET/SHOW EDITOR

Invokes a text editor. Use the MAIL command SEND/EDIT to edit the message. The SHOW EDITOR command displays the name of the editor.

format

```
SET EDITOR editor-name
SHOW EDITOR
```

parameter

editor-name

Indicates the name of the editor. You can use any callable editor available on your system.

MAIL-28 MAIL
SET/SHOW EDITOR

qualifiers

None.

example

```
MAIL> SHOW EDITOR
Your editor is EDT
```

```
MAIL> SET EDITOR TPU
MAIL> SHOW EDITOR
Your editor is TPU
```

```
MAIL> SEND/EDIT
To: WHITE::STAFFORD
Subject: Manufacturing Office
```

This example shows how to change the editor from the default EDT editor to the TPU editor. Enter the MAIL command SEND/EDIT to edit the text of a message. Send the message by pressing CTRL/Z.

SET/SHOW COPY_SELF

Sets the default for determining whether the SEND, REPLY, or FORWARD commands return to the sender a copy of the message being sent.

By specifying NOSEND, NOREPLY, or NOFORWARD with the SET COPY_SELF command, you can clear any default copying you have established with the SET COPY_SELF command.

The SHOW COPY_SELF command displays which command (SEND, REPLY, or FORWARD) automatically sends a copy of the message to you.

format

```
SET COPY_SELF command [command]
SHOW COPY_SELF
```

parameter

command

The **command** parameter can be any one of the following: SEND, NOSEND, REPLY, NOREPLY, FORWARD, or NOFORWARD. You can use NOSEND, NOREPLY, and NOFORWARD to reverse previous settings of SEND, REPLY, or FORWARD.

example

```
MAIL> SET COPY_SELF SEND  
MAIL> SHOW COPY_SELF  
Automatic copy to yourself on SEND
```

This example shows how to use the SET COPY_SELF command to enable copies of mail messages you send to be returned back to you. The SHOW COPY_SELF command indicates that you have enabled automatic copying when you enter a SEND command.

SET/SHOW FORWARD

Sets a forwarding address for your mail. After you enter the SET FORWARD command, the address you specify will receive mail messages.

The default you establish with the SET FORWARD command remains in effect until you enter the SET NOFORWARD command.

The SHOW FORWARD command displays the name of the specified forwarding address.

If you have SYSNAM privilege, you can set and show forwarding addresses for other users.

format

```
SET FORWARD address  
SET NOFORWARD  
SHOW FORWARD
```

parameter

address

Indicates the address (NODE::NAME) to which your mail is forwarded.

qualifiers

/ALL

The /ALL qualifier lists forwarding information or displays a message if the specified user does not have forwarding enabled.

/USER=*username*

Indicates the name of another user for whom you are setting or showing a forwarding address. You can use the /USER qualifier only if you have SYSNAM privilege. With the SHOW FORWARD command, there are two ways to show a user's forwarding address: You can specify the user name, or you can use the wildcard characters (* or %) to search for names with a particular string in common.

MAIL-30 MAIL
SET/SHOW FORWARD

example

```
MAIL> SET FORWARD NEXUS::LARS
MAIL> SHOW FORWARD
Your mail is being forwarded to NEXUS::LARS
MAIL>
```

This example shows how a user named LARS establishes a forwarding address on node NEXUS with the SET FORWARD command, and displays the forwarding address with the SHOW FORWARD command.

SET/SHOW PERSONAL_NAME

Enables you to append a field to the end of the From: field of mail messages you send. You can fill this field with your full name or any other information.

The SET NOPERSONAL_NAME command clears any name you previously specified with the SET PERSONAL_NAME command.

The SHOW PERSONAL_NAME command displays a user's personal name.

format

```
SET PERSONAL_NAME "text-string"
SET NOPERSONAL_NAME
SHOW PERSONAL_NAME
```

parameter

"text-string"

Specifies the string for the From: field of mail messages you send. You must enclose the string in quotation marks; otherwise, MAIL converts it to uppercase letters. You must begin the string with an alphabetic character and avoid two consecutive embedded spaces within the string. The length of the text string should not exceed 127 characters.

qualifiers

/ALL

The /ALL qualifier lists personal name information or displays a message if the specified user has not entered a personal name.

/USER=name

Used with the SHOW PERSONAL_NAME command to allow a user with SYSNAM privilege to list personal names set by other users. There are two ways to show a user's personal name. The user name can be specified, or you can use the wildcard characters (* or %) to search for names with a particular string in common.

example

MAIL> SET PERSONAL_NAME "Catherine the Great"

.
. .
. .

MAIL> SEND

.
. .
. .

New mail on node FLAXEN from ALPHA::BELLINI "Catherine the Great"

.
. .
. .

From: ALPHA::BELLINI "Catherine the Great" 19-APR-1988 15:34
To: FLAXEN::STARCK

This example shows how a user named BELLINI sets her personal name to Catherine the Great.

Sort/Merge Utility

The VMS Sort/Merge Utility sorts records or merges input files. To sort one or more input files, specify the SORT command. These files are sorted according to the fields you select and one reordered output file is generated. To merge up to 10 input files that have previously been sorted according to the same key fields, specify the MERGE command. One output file is generated.

format

SORT *input-files output-file*

parameters

input-files

Specifies the files to be sorted. Up to 10 input files can be sorted to create one output file. Multiple input file specifications must be separated by commas. The default file type is DAT.

output-file

Specifies the file to be created. Only one file can be specified. If you omit a file type in the file specification, SORT defaults to the file type of the first input file.

format

MERGE *input-files output-file*

parameters

input-files

Specifies the files to be merged. Up to 10 files, presorted by the same keys, can be specified. Multiple file specifications must be separated by commas. The default file type is DAT.

output-file

Specifies the merged file to be created. Only one output file can be specified. If you omit a file type in the file specification, MERGE defaults to the file type of the first input file.

usage summary

Both the SORT and MERGE commands invoke the VMS Sort/Merge Utility. After completing an operation, the Sort/Merge Utility exits and returns the user to DCL command level. You can specify where you want the results of a SORT/MERGE operation with the output file parameter.

SORT-2 SORT/MERGE /CHECK_SEQUENCE

SORT/MERGE Qualifiers

This section describes the qualifiers to the **SORT** and **MERGE** commands. These qualifiers enable you to define your key fields, to describe the data in those fields, and to specify various **SORT** options.

/CHECK_SEQUENCE

Verifies the sequence of the records in **MERGE** input files. By default, the sequence of records is checked.

format

/CHECK_SEQUENCE
/NOCHECK_SEQUENCE

example

```
$ MERGE/KEY=(SIZE:4,POSITION:3)/NOCHECK_SEQUENCE PRICE1.DAT -  
_ $ PRICE2.DAT PRICE.LIS
```

The **/NOCHECK_SEQUENCE** qualifier specifies that sequence of the input files, **PRICE1.DAT** and **PRICE2.DAT**, need not be checked because the records in those files were sorted on the same key and the sequence of records is correct.

/COLLATING_SEQUENCE

Selects one of three predefined collating orders for character key fields, or specifies the name of a National Character Set (NCS) collating sequence to be used in comparing character keys. **SORT** arranges characters in ASCII sequence by default; the **EBCDIC** and **MULTINATIONAL** sequences can also be used.

format

/COLLATING_SEQUENCE=type
/COLLATING_SEQUENCE=cs-name

parameters

ASCII

Arranges characters according to ASCII sequence. ASCII is the default sequence and need not be specified.

EBCDIC

Arranges characters according to EBCDIC sequence. The characters remain in ASCII representation; only the order is changed.

MULTINATIONAL

Arranges characters according to MULTINATIONAL sequence, which collates the international character set. When you use the MULTINATIONAL sequence, characters are ordered according to the following rules:

- All diacritical forms of a character are given the collating value of the character (A', A", A' collate as A).
- Lowercase characters are given the collating value of their uppercase equivalents (a collates as A, a" collates as A").
- If two strings compare as equal, tie-breaking is performed. The strings are compared to detect differences due to diacritical marks, ignored characters, or characters that collate as equal although they are actually different. If the strings still compare as equal, another comparison is done based on the numeric codes of the characters. In this final comparison, lowercase characters are ordered before uppercase.

Care should be taken when sorting or merging files for further processing using the MULTINATIONAL sequence. Sequence checking procedures in most programming languages compare numeric characters. Because MULTINATIONAL is based on actual graphic characters and not on the codes representing those characters, normal sequence checking does not work.

Note that, some languages do not support MULTINATIONAL comparisons and instead can use the LIB\$COMPARE_MULTI routine.

CS-NAME

Arranges character keys according to the named sequence, which must be a collating sequence defined in a VAX NCS library.

example

```
$ SORT/COLLATING_SEQUENCE=MULTINATIONAL -  
_ $ NAMES.DAT,NOM.DAT LIST.LIS
```

This SORT command arranges the input files NAMES.DAT and NOM.DAT according to the MULTINATIONAL collating sequence to create the output file LIST.LIS.

SORT-4 SORT/MERGE /DUPLICATES

/DUPLICATES

Eliminates all but one of multiple records with duplicate keys. By default, SORT retains multiple records with duplicate keys.

format

/DUPLICATES
/NODUPLICATES

example

```
$ SORT/KEY=(POSITION:3,SIZE:5,DECIMAL)/NODUPLICATES -  
_ $ ACCT1,ACCT2 ACCT.LIS
```

This SORT command arranges the two input files according to the key supplied and eliminates all but one of multiple records with equal keys.

/KEY

Describes key fields, including the position, size, sorting order, and data type. By default, SORT reorders a file by sorting entire records with character data in ascending order. Any other type of key field must be specified. When specifying multiple keys, use a separate /KEY qualifier for each key.

format

/KEY=(field [...])

qualifier values

POSITION:n

Specifies the position of the first byte in the key field. A value of 1 to 32,767 may be specified. The first byte in a record is considered position 1. Both the position and the size of the key field must be specified. If a decimal sign is stored in a separate byte in the key field, that byte is counted when you determine position.

SIZE:n

Specifies the length of the key field. Both the position and size of the key field must be specified. The total composite size of all keys and the original input record length must be less than 32,767 bytes. If the decimal sign is stored in a separate byte in the key field, that byte is not counted toward the size of the data.

The data type of the key determines what values are acceptable when specifying size:

- 1 to 32,767 characters for character data

SORT/MERGE SORT-5 /KEY

- 1, 2, 4, 8, or 16 bytes for binary data
- 1 to 31 digits for decimal data
- No value is necessary for floating point data

ASCENDING	Orders the sorting operation in ascending alphabetical or numerical order. ASCENDING is the default order.
DESCENDING	Orders the sorting operation in descending alphabetical or numerical order.
CHARACTER	Specifies character data in the key field. CHARACTER is the default data type.
BINARY	Specifies binary data in the key field.

SIGNED—Specifies signed binary or decimal data in key field. SIGNED is the default for binary and decimal data.

UNSIGNED—Specifies unsigned binary or decimal data in the key field.

F_FLOATING	Specifies F_FLOATING format data in the key field.
D_FLOATING	Specifies D_FLOATING format data in the key field.
G_FLOATING	Specifies G_FLOATING format data in the key field.
H_FLOATING	Specifies H_FLOATING format data in the key field.
DECIMAL	Specifies decimal data in the key field.

TRAILING_SIGN—Specifies trailing sign decimal data in the key field. TRAILING_SIGN is the default for decimal data.

LEADING_SIGN—Specifies leading sign decimal data in the key field.

OVERPUNCHED_SIGN—Specifies overpunched decimal data in the key field. OVERPUNCHED_SIGN is the default for decimal data.

SEPARATE_SIGN—Specifies separate sign decimal data in the key field.

ZONED	Specifies zoned decimal data in the key field.
PACKED_DECIMAL	Specifies packed decimal data in the key field.
NUMBER:n	Specifies the order of priority of each key if you do not list multiple keys in the order of their priority. A value of 1 to 255 may be specified.

example

```
$ SORT/KEY=(POS:16,SIZ:3)/KEY=(POS:1,SIZ:11) -  
_$/KEY=(POS:40,SIZ:2,DESC) YRENDAVG.DAT YRAVGSRT.LIS
```

This SORT command identifies three key fields. The input file, YRENDAVG, is first sorted by the key beginning in position 16, then by the key beginning in position 1, and finally by the key beginning in position 40. The third key used sorts in descending order.

SORT-6 SORT/MERGE /PROCESS

/PROCESS

Defines the internal sorting process. The **/PROCESS** qualifier allows you to choose one of four processes: record, tag, address, or index. By default, SORT uses a record sorting process. Use only with the SORT command.

format

/PROCESS=type

qualifier values

RECORD

Keeps records intact while sorting and produces an output file consisting of complete records. Record is the default sorting process.

TAG

Sorts only the keys and then rereads the input file to produce an output file consisting of complete records.

ADDRESS

Sorts only the keys and produces an output file that is an index of record addresses in binary format. The index must be submitted to a program for further processing.

INDEX

Sorts only the keys and produces an output file that is an index of keys and of record addresses in binary form. The index must be submitted to a program for further processing.

example

```
$ SORT/KEY=(POS:40,SIZ:2,DESC)/PROCESS=TAG YRENDAVG.DAT-  
_ $ DESCYRAVG.LIS
```

This SORT operation uses a tag sorting process to create the output file, DESCYRAVG.LIS.

/SPECIFICATION

Identifies a SORT or MERGE specification file.

format

/SPECIFICATION=filespec

qualifier value

filespec

Specifies the SORT/MERGE specification file. The default file type is SRT.

example

```
$ SORT/SPECIFICATION=ACCTS.SRT SALES1.DAT,SALES2.DAT MAILING.LIS
```

This SORT command arranges the input files according to the instructions detailed in the specification file, ACCTS.SRT.

/STABLE

Directs records with equal keys to the output file in their input file order. The default condition is /NOSTABLE.

format

/STABLE

/NOSTABLE

example

```
$ SORT/KEY=(POS:1,SI:5,DECIMAL)/STABLE PRICESA.DAT,PRICESB.DAT -  
_ $ PRICESC.DAT SUMMARY.LIS
```

In this SORT operation, records with equal keys from PRICESA.DAT will be listed first, followed by those from PRICESB.DAT, followed by those from PRICESC.DAT.

/STATISTICS

Displays a statistical summary that can be used for optimization.

format

/STATISTICS

SORT-8 SORT/MERGE /STATISTICS

example

```
$ SORT /STATISTICS PRICE1.DAT,PRICE2.DAT PRICE.LIS
```

This SORT command results in the following statistical display:

VMS Sort/Merge Statistics

Records read:	793	Input record length:	80
Records sorted:	793	Internal length:	80
Records output:	793	Output record length:	80
Working set extent:	100	Sort tree size:	412
Virtual memory:	433	Number of initial runs:	2
Direct I/O:	22	Maximum merge order:	2
Buffered I/O:	9	Number of merge passes:	1
Page faults:	3418	Work file allocation:	114
Elapsed time: 00:00:05.98		Elapsed CPU:	00:00:03.63

In the sample statistics display, the sort data structure size is limited by the small working set extent. By doubling the working set extent you can almost double the sort data structure size, enabling all the records to fit in memory without using work files.

/WORK_FILES

Used for optimization.

format

/WORK_FILES=n

qualifier value

n

Specifies the number of work files requested; 0 to 10 files may be specified.

example

```
$ ASSIGN DRA5: SORTWORK0  
$ ASSIGN DB0: SORTWORK1  
$ ASSIGN DB1: SORTWORK2  
$ SORT/KEY=(POS:1,SI:80)/WORK_FILES=3 -  
_ $ STATS1,STATS2,STATS3,STATS4 SUMMARY.LIS
```

Since the input files in this sort operation are large files, specifying three work files improves the efficiency of the sort operation.

Input File Qualifiers

This section describes the qualifier to the input file. This qualifier should be specified immediately after the input file specification.

/FORMAT

Input File Qualifier

Defines input file characteristics; allows you to specify or override record or file size.

format

input filespec/FORMAT=(type:n,[...])

qualifier values

RECORD_SIZE:n

Specifies the file's longest record length (LRL) in bytes.

The maximum longest record length that can be specified depends on the file organization:

Sequential files	32,767
Relative files	16,383
Indexed-sequential files	16,362

These totals include control bytes for variable records with fixed-length control (VFC) format.

FILE_SIZE:n

Specifies input file size in blocks. The maximum file size accepted is 4,294,967,295 blocks.

**SORT-10 SORT/MERGE
/ALLOCATION**

Output File Qualifiers

This section describes the qualifiers to the output file. These qualifiers should be specified immediately after the output file specification.

/ALLOCATION

Output File Qualifier

Used for optimization.

format

output filespec/ALLOCATION=n

qualifier value

n

Specifies the number of blocks to be allocated. A value of 1 to 4,294,967,295 is allowed.

/BUCKET_SIZE

Output File Qualifier

Used for optimization.

format

output filespec/BUCKET_SIZE=n

qualifier value

n

Specifies the bucket size. A value of 1 to 32 is allowed.

/CONTIGUOUS

Output File Qualifier

Used for optimization.

format

output filespec/CONTIGUOUS

/FORMAT

Output File Qualifier

Specifies the output file record format if it differs from the input file format.

format

*output filespec*FORMAT=(*type:n ...*)

qualifier values

FIXED:n

Specifies fixed-length records in the output file.

VARIABLE:n

Specifies variable-length records in the output file.

CONTROLLED:n

Specifies variable with fixed-length control (VFC) records in the output file.

n

Optionally indicates the maximum record size (in bytes) of the output records. The maximum record size allowed depends on the file organization.

Sequential files	32,767
Relative files	16,383
Indexed-sequential files	16,362

These totals include control bytes. If you do not specify the maximum record size, the default is a length large enough to hold the longest output record.

SIZE:n

Specifies the size, in bytes, of the fixed portion of VFC (CONTROLLED) records, up to a maximum of 255 bytes. If you do not specify SIZE, the default is the size of the fixed portion of the first input file. If you specify this size as 0, RMS defaults the value to 2 bytes.

BLOCK_SIZE:n

Specifies the output file's block size, in bytes, if you have directed the file to magnetic tape. You can also accept the default. If the input file is a tape file, the block size of the output file defaults to that of the input file. Otherwise, the output file block size defaults to the size used when the tape was mounted.

SORT-12 SORT/MERGE
/FORMAT

Acceptable values for block size n range from 20 to 65,532. To ensure correct data interchange with other Digital systems, however, specify a block size of not more than 512 bytes. For compatibility with most non-Digital systems, the block size should not exceed 2048 bytes.

/INDEXED_SEQUENTIAL

Output File Qualifier

Defines file organization.

format

output filespec/INDEXED_SEQUENTIAL

/OVERLAY

Output File Qualifier

Specifies that the output file is to be overlaid on, or written to, an existing empty file.

format

output filespec/OVERLAY

/RELATIVE

Output File Qualifier

Defines output file organization as relative.

format

output filespec/RELATIVE

/SEQUENTIAL

Output File Qualifier

Defines output file organization as sequential.

format

output filespec/SEQUENTIAL

Appendix A

TFF Facility

A.1 Using the Terminal Fallback Facility

The VMS Terminal Fallback Facility (TFF) provides table-driven character conversion for terminals. Because every computer terminal can display only one set of characters at a time and each keyboard has a limited number of keys, software developed with one character set or terminal can be impossible to use with another character set or terminal. To help you bridge the gap between incompatible character sets and incompatible terminals, TFF converts characters transparently to software applications (unnoticed by the application software unless explicit inquiries are made). TFF can convert one character to many for characters sent to a terminal, and one to one for characters entered from a terminal.

TFF provides terminal fallback. When an application program sends a character that a terminal cannot display, TFF replaces that character with the closest possible visual character that the terminal can display. This is called **fallback**.

Finally, TFF can perform character compose emulation on input from a terminal. You can create characters that have no associated face on the keyboard by combining two existing characters. This process is known as **composing**. Although TFF offers compose sequence tables, you can also control which keys are **auto-compose** keys.

One of the applications of TFF is to allow users with National Replacement Character (NRC) set terminals to use software developed with the DEC Multinational Character Set (MCS). MCS is essentially the ASCII character set plus 128 characters currently used by owners of NRC terminals around the world.

TFF supersedes the function formerly provided by the (VMS Version 4.x) DCL command SET TERMINAL/FALLBACK. Your system manager uses Terminal Fallback Utility (TFU) commands to set up the TFF environment; you can use TFU commands to set your TFF terminal parameters.

A.1.1 The Purpose of Terminal Fallback

Terminals have physical limits. Every computer terminal can display one set of characters and each keyboard has a limited number of keys. Characters are arranged into character sets, where each character has a cardinal number (1, 2, 3, and so forth). Computers then use a character's cardinal number to tell the terminal to display that character. For example, to display the character A, a computer sends the binary value 64 to the terminal. In the same way, when you press the key labeled A, the terminal sends the binary value 64 to the computer.

One common character set is the ASCII character set, designed by the United States primarily for the English language. The ASCII character set, however, does not include many characters used in languages other than English. For example, the ASCII character set does not include accented characters.

Because of the limitations of the ASCII character set, many countries replace some symbols in ASCII with local characters, mostly accented, to produce their own variant of ASCII. A country-specific variant of ASCII is called a National Replacement Character set (NRC).

NRCs do not, however, solve the needs of all countries. Few countries are able to get all the characters they want into ASCII, because ASCII consists of a fixed set of symbols. Also, different countries replace the same ASCII symbol with different local characters. This leaves application software highly dependent on a country's NRC. Software designed using one country's NRC cannot be used in other countries.

In an effort to solve these problems, Digital Equipment Corporation designed a Multinational Character Set (MCS). MCS contains twice as many characters as ASCII, and covers the needs of most European languages. The VT200-series, VT300-series, and VAX workstation terminals, for example, use MCS.

To use an NRC terminal with an MCS-specific application, characters must be converted going to and from the terminal to MCS. The Terminal Fallback Facility provides this conversion transparently to the application through a library of character tables.

Because most NRC terminals cannot display all the MCS characters, TFF replaces those characters with fallback characters. For example, if a terminal cannot display the Japanese yen sign ¥, TFF sends a Y.

A.1.2 The Purpose of Compose Characters

Sometimes, you cannot use software developed with one character set on a terminal that does not include all of the required characters. Because each terminal keyboard has a limited number of keys, you must use compose sequences to create characters that have no associated face on the keyboard. TFF provides two compose sequence tables, LATIN_1 and ISO_COMPOSE. The default compose sequence table is LATIN_1. You should use this table with Digital applications. However, if an application uses a character set other than MCS, you need a

matching compose sequence table. For example, to use the ISO table ISO_VT100MCS, you need the compose sequence table ISO_COMPOSE. Use Terminal Fallback Utility commands to choose TFF tables. After you choose the tables, TFF handles the conversion process.

A.1.2.1 Composing Characters with TFF

To compose a character in the TFF environment, you press CTRL/K, and then enter the two existing keyboard characters that make up the compose sequence. For example, to compose the copyright sign, ©, press CTRL/K followed by CO; to compose ñ, press CTRL/K followed by n~ (lower case n followed by tilde). You can create some characters from more than one compose sequence. Some compose sequences are order or case sensitive, or both. To abort a compose sequence, press the DELETE key. If you press CTRL/K before completing the compose sequence, TFF restarts the compose sequence. If you press any other control key before completing the compose sequence, the compose sequence fails and the control character is sent to the application.

A.1.2.2 Compose Sequences—Digital LK201 Keyboard and the LATIN_1 Table

Table A-1 contains the sequences defined for the standard LK201 keyboard used with VT200-series terminals, DECmate, VAXmate, and Digital workstations.

TFF converts characters that do not have an accurate visual representation to the closest possible fallback representation. If no such fallback exists, TFF replaces the character with an underscore.

Table A-1: LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
"	"	quotation mark	" space	
#	#	number sign	+ +	
'	'	apostrophe	' space	
@	@	commercial at	a a	
[[opening bracket	((
\	\	backslash	// or /	
]]	closing bracket))	
^	^	circumflex accent	^ space	
'	'	single quote	' space	
{	{	opening brace	(-	
		vertical line	/ ^	

(continued on next page)

Table A-1 (Cont.): LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
}	}	closing brace) -	
~	~	tilde	~ space	
!	¡	inverted exclamation	! !	
c	¢	cent sign	c / or c	
L	£	pound sign	l - or l =	
Y	¥	yen sign	y - or y =	
-	§	section sign	s o or s 0 or s !	
-	¤	currency sign	x o or x 0	
-	©	copyright sign	c o or c 0	
a	ª	female ordinal indicator	a -	
<	«	double open angle brackets	< <	
-	°	degree sign	0 ^	
-	±	plus/minus sign	+ -	
1	¹	superscript 1	1 ^	
2	²	superscript 2	2 ^	
3	³	superscript 3	3 ^	
u	µ	micro sign	/ u	Order
-	¶	paragraph sign (pilcrow)	P !	
.	·	middle dot	^ .	
o	º	masculine ordinal indicator	o _	
>	»	double close angle brackets	> >	
-	¼	fraction one-quarter	1 4	Order
-	½	fraction one-half	1 2	Order
?	¿	inverted question mark	? ?	
A	À	A grave	A ^	Case
A	Á	A acute	A ^	Case
A	Â	A circumflex	A ^	Case
A	Ã	A tilde	A ~	Case
A	Ä	A umlaut	A " " " " " "	Case

(continued on next page)

Table A-1 (Cont.): LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
A	Å	A ring	A *	Case
-	Æ	A E ligature	A E	Order & Case
C	Ç	C cedilla	C ,	Case
E	È	E grave	E ‘	Case
E	É	E acute	E ’	Case
E	Ê	E circumflex	E ^	Case
E	Ë	E umlaut	E "	Case
I	Ì	I grave	I ‘	Case
I	Í	I acute	I ’	Case
I	Î	I circumflex	I ^	Case
I	Ï	I umlaut	I "	Case
N	Ñ	N tilde	N ~	Case
O	Ò	O grave	O ‘	Case
O	Ó	O acute	O ’	Case
O	Ô	O circumflex	O ^	Case
O	Õ	O tilde	O ~	Case
O	Ö	O umlaut	O "	Case
O	Ø	O slash	O /	Case
-	Œ	O E ligature	O E	Order & Case
U	Û	U grave	U ‘	Case
U	Ú	U acute	U ’	Case
U	Û	U circumflex	U ^	Case
U	Ü	U umlaut	U "	Case
Y	ÿ	Y umlaut	Y "	Case
-	ß	German small sharp s	s s	Case
a	à	a grave	a ‘	Case
a	á	a acute	a ’	Case
a	â	a circumflex	a ^	Case
a	ã	a tilde	a ~	Case
a	ä	a umlaut	a "	Case
a	å	a ring	a *	Case

(continued on next page)

Table A-1 (Cont.): LATIN_1 Compose Sequence Table

Fallback	Character	Name	Compose	Sensitivity
_	æ	a e ligature	a e	Order & Case
c	ç	c cedilla	c ,	Case
e	è	e grave	e `	Case
e	é	e acute	e ´	Case
e	ê	e circumflex	e ^	Case
e	ë	e umlaut	e "	Case
i	ì	i grave	i `	Case
i	í	i acute	i ´	Case
i	î	i circumflex	i ^	Case
i	ï	i umlaut	i "	Case
n	ñ	n tilde	n ~	Case
o	ò	o grave	o `	Case
o	ó	o acute	o ´	Case
o	ô	o circumflex	o ^	Case
o	õ	o tilde	o ~	Case
o	ö	o umlaut	o "	Case
o	ø	o slash	o /	Case
_	œ	o e ligature	o e	Order & Case
u	ù	u grave	u `	Case
u	ú	u acute	u ´	Case
u	û	u circumflex	u ^	Case
u	ü	u umlaut	u "	Case
y	ÿ	y umlaut	y "	Case

Note that the characters circumflex (^), tilde (~), and grave accent (`) are used frequently to compose MCS characters. Many NRC sets, however, replace them with an NRC character. Thus, they are not available on an NRC keyboard. For each one of these keys, TFF accepts a replacement key. If you use this replacement key, however, you must begin the compose sequence with it; these compose sequences are order sensitive. For example, you can compose Ê with ^E, E^ or \$E, but not with E\$. TFF offers the following replacement keys:

- \$ (dollar sign) replaces ^ (circumflex)
- % (percent) replaces ~ (tilde)
- & (ampersand) replaces ` (grave accent)

A.1.3 Setting TFF Terminal Parameters

After your system manager has installed and set up the environment for TFF on your system, you can use Terminal Fallback Utility (TFU) commands to set, change, and display TFF terminal parameters. You can choose and set the default conversion tables for your terminal.

Typically, your system manager sets one fallback table and one compose sequence table as system defaults. These, and perhaps other tables, are loaded into nonpaged dynamic memory pool, making them available to you. If you want to make use of the system conversion table defaults, you can log in to any local terminal and enter a DCL or TFU SET TERMINAL/FALLBACK command. In addition, you can use the TFU SET TERMINAL/FALLBACK command to set your own default fallback tables from any tables previously loaded into nonpaged dynamic memory pool. If you need a table that is available in the TFF library, but not loaded into nonpaged dynamic memory pool, you must ask your system manager to load the table.

After your system manager sets up the TFF environment, you can make full use of the Terminal Fallback Utility SET TERMINAL/FALLBACK command and its many options. For a detailed description of these options, see the TFU Commands section.

Use the following commands to manage TFF terminal parameters:

Command	Use
DIRECTORY	Displays the conversion tables available in the TFF library.
SET TERMINAL/FALLBACK[=option]	Enables or modifies TFF terminal parameters. This is the primary Terminal Fallback Utility command. Use this command to activate the desired behavior at the specified terminal.
SHOW DEFAULT_TABLE	Displays the default fallback character conversion table.
SHOW TABLES	Displays information about loaded conversion tables. This information is helpful before you try to enable TFF terminal parameters.
SHOW TERMINAL/FALLBACK	Displays TFF terminal parameters.

Appendix B

Character Sets

The following tables present the ASCII character set and the DEC Multinational Character Set.

B.1 ASCII Character Set

The ASCII character set consists of the characters shown in the following table. The characters with names are not printable. You can calculate the numeric value of a character by constructing a 2-digit hexadecimal number in which the column position of the character represents the 16 position of the hexadecimal number and the row position of the character represents the units position of the number. For example, an uppercase A has the numeric value 41 hexadecimal. String comparisons are made using these values.

Hex Values	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(8	H	X	h	x
9	HT	EM)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[k	{
C	FF	FS	'	<	L	\	l	
D	CR	GS	-	=	M]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

B.2 ASCII and DEC Multinational Character Set Tables

Figure B-1 represents the ASCII character set (characters with decimal values 0 through 127). The first half of each of the numbered columns identifies the character as you would enter it on a VT300-, VT200-, or VT100-series terminal or as you would see it on a printer (except for the nonprintable characters). The remaining half of each column identifies the character by the binary value of the byte; the value is stated in three radices—octal, decimal, and hexadecimal. For example, the uppercase letter A has, under ASCII conventions, a storage value of hexadecimal 41 (a bit configuration of 01000001), equivalent to 101 in octal notation and 65 in decimal notation.

The ASCII character set comprises the first half of the DEC Multinational Character Set. Figure B-2 represents the second half of the DEC Multinational Character Set (characters with decimal values 128 through 255). The first half of each of the numbered columns identifies the character as you would see it on a VT300- or VT200-series terminal or printer (these characters cannot be output on a VT100-series terminal).

Figure B-1: Graphical Representation of the ASCII Character Set

Row	Column								
	b8 Bits b7 b6 b5 b4 b3 b2 b1		0	1	2	3	4	5	6
0	0 0 0 0	NUL	DLE	SP	0	@	P	\	p
1	0 0 0 1	SOH	DC1 (XON)	!	1	A	Q	a	q
2	0 0 1 0	STX	DC2	"	2	B	R	b	r
3	0 0 1 1	ETX	DC3 (XOFF)	#	3	C	S	c	s
4	0 1 0 0	EOT	DC4	\$	4	D	T	d	t
5	0 1 0 1	ENQ	NAK	%	5	E	U	e	u
6	0 1 1 0	ACK	SYN	&	6	F	V	f	v
7	0 1 1 1	BEL	ETB	'	7	G	W	g	w
8	1 0 0 0	BS	CAN	(8	H	X	h	x
9	1 0 0 1	HT	EM)	9	I	Y	i	y
10	1 0 1 0	LF	SUB	*	:	J	Z	j	z
11	1 0 1 1	VT	ESC	+	;	K	[k	{
12	1 1 0 0	FF	FS	,	<	L	\	l	
13	1 1 0 1	CR	GS	-	=	M]	m	}
14	1 1 1 0	SO	RS	.	>	N	^	n	~
15	1 1 1 1	SI	US	/	?	O	_	o	DEL

Key

Character	ESC	33 27 1B	Octal Decimal Hex
-----------	-----	----------------	-------------------------

B-4 Character Sets

Figure B-2: Graphical Representation of the DEC Multinational Extension to the ASCII Character Set

8		9		10		11		12		13		14		15		Column	Row			
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	b8 b4	b7 b3	b6 b2	b5 b1	
	200 128 80	DCS	220 144 90	240 160 A0	°	260 176 B0	À	300 192 C0		320 208 D0	à	340 224 E0		360 240 F0		0	0	0	0	0
	201 129 81	PU1	221 145 91	241 161 A1	±	261 177 B1	Á	301 193 C1	Ñ	321 209 D1	á	341 225 E1	ñ	361 241 F1		0	0	0	1	1
	202 130 82	PU2	222 146 92	242 162 A2	¢	262 178 B2	Â	302 194 C2	Ò	322 210 D2	â	342 226 E2	ò	362 242 F2		0	0	1	0	2
	203 131 83	STS	223 147 93	243 163 A3	£	263 179 B3	Ã	303 195 C3	Ó	323 211 D3	ã	343 227 E3	ó	363 243 F3		0	0	1	1	3
IND	204 132 84	CCH	224 148 94	244 164 A4		264 180 B4	Ä	304 196 C4	Ô	324 212 D4	ä	344 228 E4	ö	364 244 F4		0	1	0	0	4
NEL	205 133 85	MW	225 149 95	245 165 A5	¥	265 181 B5	Å	305 197 C5	Ö	325 213 D5	å	345 229 E5	ö	365 245 F5		0	1	0	1	5
SSA	206 134 86	SPA	226 150 96	246 166 A6	¶	266 182 B6	Æ	306 198 C6	Ø	326 214 D6	æ	346 230 E6	ø	366 246 F6		0	1	1	0	6
ESA	207 135 87	EPA	227 151 97	247 167 A7	§	267 183 B7	Ç	307 199 C7	Ɔ	327 215 D7	ç	347 231 E7	œ	367 247 F7		0	1	1	1	7
HTS	210 136 88		230 152 98	250 168 A8	¨	270 184 B8	È	310 200 C8	Ø	330 216 D8	è	350 232 E8	ø	370 248 F8		1	0	0	0	8
HTJ	211 137 89		231 153 99	251 169 A9	©	271 185 B9	É	311 201 C9	Ù	331 217 D9	é	351 233 E9	ù	371 249 F9		1	0	0	1	9
VTS	212 138 8A		232 154 9A	252 170 AA	ª	272 186 BA	Ê	312 202 CA	Ú	332 218 DA	ê	352 234 EA	ú	372 250 FA		1	0	1	0	10
PLD	213 139 8B	CSI	233 155 9B	253 171 AB	«	273 187 BB	Ë	313 203 CB	Û	333 219 DB	ë	353 235 EB	û	373 251 FB		1	0	1	1	11
PLU	214 140 8C	ST	234 156 9C	254 172 AC	¼	274 188 BC	Ì	314 204 CC	Ü	334 220 DC	ì	354 236 EC	ü	374 252 FC		1	1	0	0	12
RI	215 141 8D	OSC	235 157 9D	255 173 AD	½	275 189 BD	Í	315 205 CD	Ý	335 221 DD	í	355 237 ED	ý	375 253 FD		1	1	0	1	13
SS2	216 142 8E	PM	236 158 9E	256 174 AE		276 190 BE	Î	316 206 CE		336 222 DE	î	356 238 EE		376 254 FE		1	1	1	0	14
SS3	217 143 8F	APC	237 159 9F	257 175 AF	¾	277 191 BF	Ï	317 207 CF	ß	337 223 DF	ï	357 239 EF		377 255 FF		1	1	1	1	15

Key

Character	ESC	33 27 1B	Octal Decimal Hex
-----------	-----	----------------	-------------------------

ZK-1753-GE

Appendix C

Expressions

The following table lists data operations and comparisons in order of precedence, beginning with the highest:

Operator	Precedence	Description
+	1	Indicates a positive number
-	1	Indicates a negative number
*	2	Multiplies two numbers
/	2	Divides two numbers
+	3	(1) Adds two numbers (2) Concatenates two character strings
-	3	(1) Subtracts two numbers (2) Subtracts two character strings
.EQS.	4	Tests if two character strings are equal
.GES.	4	Tests if first character string is greater than or equal
.GTS.	4	Tests if first character string is greater than
.LES.	4	Tests if first character string is less than or equal
.LTS.	4	Tests if first character string is less than
.NES.	4	Tests if two character strings are not equal
.EQ.	4	Tests if two numbers are equal
.GE.	4	Tests if first number is greater than or equal to
.GT.	4	Tests if first number is greater than
.LE.	4	Tests if first number is less than or equal to
.LT.	4	Tests if first number is less than
.NE.	4	Tests if two numbers are not equal
.NOT.	5	Logically negates a number
.AND.	6	Combines two numbers with a logical AND
.OR.	7	Combines two numbers with a logical OR

C-2 Expressions

The following tables demonstrate the results of logical operations on a bit-by-bit basis and a number-by-number basis. In logical operations, a character string beginning with an uppercase or lowercase T or Y is treated as the number 1; a character string beginning with any other character is treated as the number 0. In logical operations, odd numbers are considered true and even numbers and zero are considered false.

Given		Results		
Bit A	Bit B	.NOT. A	A .AND. B	A .OR. B
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

Given		Results		
Number A	Number B	.NOT. A	A .AND. B	A .OR. B
odd	odd	even	odd	odd
odd	even	even	even	odd
even	odd	odd	even	odd
even	even	odd	even	even

Appendix D

Terminal Keys

The following tables present the operating system's interpretation of keys on terminals in the VT300-, VT200-, and VT100-series.

D.1 VT300 and VT200 Terminal Series

The following table describes how the operating system responds when various keys and control characters are pressed on VT300- or VT200-series terminals. The table assumes that line editing is enabled (the default). (Characters not mentioned in the table are treated as null characters.)

Character	HEX	System Response
CTRL/A	01	Switches between overstrike and insert modes
CTRL/B	02	Recalls previous line
CTRL/C	03	Interrupts current image (image may define alternate CTRL/C action)
CTRL/D	04	Moves cursor left one character
CTRL/E	05	Moves cursor to end of line
CTRL/F	06	Moves cursor right one character
CTRL/H	08	Moves cursor to beginning of line
CTRL/I	09	Horizontal tab
CTRL/J	0A	Deletes previous word
CTRL/M	0D	Line terminator
CTRL/O	0F	Suspends/resumes echoing of output
CTRL/Q	11	Resumes output (see CTRL/S)
CTRL/R	12	Refreshes current line
CTRL/S	13	Suspends output (see CTRL/Q)
CTRL/T	14	Displays process information (must be enabled with SET CONTROL=T command)

D-2 Terminal Keys

Character	HEX	System Response
CTRL/U	15	Deletes characters from cursor to beginning of line
CTRL/V	16	Passes next character or escape sequence to the image without interpreting it as described in this table
CTRL/X	18	Purges type-ahead buffer; if characters are on the current line, deletes characters from cursor to beginning of line
CTRL/Y	19	Interrupts current image
CTRL/Z	1A	Indicates end of file
Data keys	-	Enter appropriate character
<X	-	Deletes previous character
CTRL	-	Modifies another key
CTRL/[(ESC)	1B	Begins escape sequence
CTRL/F5	-	Executes answerback message
DOWN ARROW	-	Repeats current line
F1 (No Scroll)	-	Suspends/resumes output
F5 (Break)	-	Shuts down transmission line
F6 (Interrupt)	-	Interrupts the current image
F10 (Exit)	-	Terminates the current image or command procedure
F12 (Backspace)	08	Moves cursor to beginning of line
F13 (Line Feed)	-	Deletes previous word
F14 (^A)	01	Switches between overstrike and insert modes
LEFT ARROW	-	Moves cursor left one character
PFn	-	Can be defined (see DEFINE/KEY)
RETURN	-	Line terminator
RIGHT ARROW	-	Moves cursor right one character
TAB	-	Horizontal tab
UP ARROW	-	Repeats current line

D.2 VT100 Terminal Series

The following table describes how the operating system responds when various keys and control characters are pressed on VT100-series terminals. The table assumes that line editing is enabled (the default). (Characters not mentioned in the table are treated as null characters.)

Character	HEX	System Response
CTRL/A	01	Switches between overstrike and insert modes
CTRL/B	02	Recalls previous line
CTRL/C	03	Interrupts current image (image may define alternate CTRL/C action)
CTRL/D	04	Moves cursor left one character
CTRL/E	05	Moves cursor to end of line
CTRL/F	06	Moves cursor right one character
CTRL/H	08	Moves cursor to beginning of line
CTRL/I	09	Horizontal tab
CTRL/J	0A	Deletes previous word
CTRL/M	0D	Line terminator
CTRL/O	0F	Suspends/resumes echoing of output
CTRL/Q	11	Resumes output (see CTRL/S)
CTRL/R	12	Refreshes current line
CTRL/S	13	Suspends output (see CTRL/Q)
CTRL/T	14	Displays process information
CTRL/U	15	Deletes characters from cursor to beginning of line
CTRL/V	16	Passes next character or escape sequence to the image without interpreting it as described in this table
CTRL/X	18	Purges type-ahead buffer; if characters are on the current line, deletes characters from cursor to beginning of line
CTRL/Y	19	Interrupts current image
CTRL/Z	1A	Indicates end of file
Data keys	-	Enter appropriate character
Backspace (^H)	08	Moves cursor to beginning of line
BREAK	-	Shuts down transmission line
CTRL	-	Modifies another key
CTRL/BREAK	-	Executes answerback message
DELETE	-	Deletes previous character
DOWN ARROW	-	Repeats current line
ESC	1B	Begins escape sequence
LEFT ARROW	-	Moves cursor left one character
LINE FEED	-	Deletes previous word
NO SCROLL	-	Suspends/resumes output

D-4 Terminal Keys

Character	HEX	System Response
PFn	-	Can be defined (see DEFINE/KEY)
RETURN	-	Line terminator
RIGHT ARROW	-	Moves cursor right one character
TAB	-	Horizontal tab
UP ARROW	-	Repeats current line

Index

A

- Absolute time
 - combined with delta time, 3-14
 - default values, 3-12
 - examples, 3-13
 - rules for entering, 3-12
 - syntax, 3-12
- Access control list
 - See ACL
- Access control string
 - copying files between nodes with, 4-6
 - in a logical node name, 11-14
- Access mode
 - and the DEFINE command, 11-11
 - for a logical name, 11-11
 - for a logical name table, 11-11
 - using qualifiers to specify, 11-11
- ACL (access control list)
 - default protection, 4-10
 - definition, 1-11
- ADVANCE command (EDT), 7-12
- ALLOCATE command (DCL), 1-7
- Arrow key
 - See Down arrow key, Left arrow key,
Right arrow key, Up arrow key
- ASCII
 - collating sequence, 9-3
- ASSIGN command (DCL), 11-2
- Assignment statement
 - creating a blank line with, 12-18
 - creating a global symbol with, 12-4
 - creating a local symbol with, 12-4
 - formatting output records with, 12-18
 - for numeric overlay, 12-18
 - including a symbol as part of a
character string, 12-11
- Assignment statement (cont'd.)
 - syntax, 12-4
 - for numeric overlay, 12-17
 - for string overlay, 12-17
- Asterisk (*) wildcard character
 - rules for using, 4-5
- ATTACH command
 - restriction on using, 6-33
- ATTACH command (DCL), 10-5

B

- BACKSPACE key, 3-17
- BACKUP command (EDT), 7-12
- Batch job
 - job number of, 10-8
 - log file, 10-8
 - output, 10-8
 - passing parameters to, 13-10, 13-13
 - restarting, 10-9, 13-24
 - submitting, 10-8
 - submitting command procedure as,
1-3
 - submitting program as, 1-3
 - submitting sort operation as, 9-5
- Batch mode
 - definition, 1-3
- BOTTOM command (EDT), 7-11
- Buffer
 - displaying, 6-14
 - displaying list of system buffers, 6-14
 - editing multiple buffers, 6-14
 - EDT commands for using, 7-25
 - listing all, 6-14
 - MAIN, 7-25
 - PASTE, 7-22

Index-2

Buffer (cont'd.)

- reading file into
 - with EDT, 7-25
- writing
 - with EDT, 7-25

Buffer (EVE), 6-1

BUFFER command

- changing buffers using, 6-14
- creating a new buffer with, 6-14
- putting specific buffer into current window, 6-14
- writing buffer to a file using, 6-15

Buffer List buffer

- displaying, 6-14

Built-in command

- definition, 1-3

C

CHANGE command (EDT), 7-6

Character data

- See also Character string
- alphanumeric, 12-7
- expression, 12-12
- nonprintable, 12-8
- special, 12-8

Character string

- comparison operators in expression, 12-12
- concatenation, 12-12
- creating, 12-8
- evaluation of, 12-5
- expression, 12-12
- multiple string values in an expression, 12-12
- passing to command procedure, 13-11
- reduction, 12-12
- substring replacement in, 12-17
- symbol substitution in, 12-6
- used as symbol, 12-3

CHAR command (EDT), 7-8, 7-18

CLOSE command (DCL), 13-18

Collating sequence

- ASCII, 9-3
- EBCDIC, 9-3
- multinational, 9-4

Combination time

- examples, 3-14
- rules for entering, 3-14
- syntax, 3-14

Command

- See also Command procedure
- abbreviating, 3-6
 - in command procedures, 3-6
- built-in, 1-3
- canceling, 3-5
- DCL command line syntax, 1-3, 3-3
- rules for entering, 3-5
- types, 1-3

Command image

- definition, 1-3, 1-8

Command level

- nesting, 13-5

Command line

- continuation over multiple lines, 3-6
- editing
 - list of keys for, 3-15, 3-18
- parts of, 1-3
- recalling, 3-10, 3-11
- syntax, 1-3

Command procedure

- and file I/O, 13-17
- cleanup, 13-25
- comments in, 13-2
- creating global symbol in, 13-15
- data line in, 13-9
- definition, 13-1
- directing output to terminal, 13-16
- executing
 - interactively, 13-3
 - on remote node, 13-3
- exiting, 13-5
- format, 13-2
- I/O errors in, 13-24
- input, 13-10
 - from file, 13-14
 - from terminal, 13-14
- invoking within a command procedure, 13-3
- loop in, 13-7
- nested, 13-4
- passing character string to, 13-11

- Command procedure (cont'd.)
 - passing data to, 13-9
 - passing parameters to, 12-5, 13-10
 - passing symbols to, 13-11
 - redirecting output, 13-16
 - returning status value in, 13-5
 - SET DEFAULT command (DCL) in, 13-17
 - submitting as batch job, 1-3
 - writing file from a, 13-18
 - Command qualifier, 3-8
 - definition, 3-8
 - rules for entering, 3-8
 - Command values
 - date and time formats, 3-12
 - Comment
 - in a command procedure, 13-2
 - CONTINUE command (DCL), 10-4
 - CONTINUE command (EDT), 7-7
 - Controller designation field
 - definition, 1-7
 - Converting from EDT to EVE, 6-33
 - COPY command (DCL), 4-6
 - COPY command (MAIL), 8-10
 - CREATE/DIRECTORY command (DCL), 5-3, 5-6
 - CREATE/NAME_TABLE command (DCL), 11-12
 - CREATE command (DCL), 4-5
 - CTRL/A
 - command line editing with, 6-6
 - CTRL/B, 6-7
 - recalling commands with, 3-10, 3-16
 - CTRL/C
 - canceling a MAIL message with, 8-5, 8-7
 - canceling EDT command with, 7-5
 - interrupting or canceling DCL commands with, 3-16
 - restriction with journaling facility, 6-13
 - CTRL/E
 - command line editing with, 6-6
 - CTRL/T
 - checking the status of your process, 2-5
 - CTRL/T (cont'd.)
 - interrupting DCL commands with, 3-16
 - CTRL/U
 - command line editing with, 6-6
 - CTRL/W, 6-12
 - refreshing screen display in EDT with, 7-7
 - refreshing screen display with, 3-16, 10-4
 - CTRL/Y
 - aborting remote session with, 2-7
 - interrupting an EDT editing session with, 7-7
 - interrupting an image with, 10-4
 - interrupting or canceling DCL commands with, 3-16
 - CTRL/Z
 - as end-of-file terminator, 3-15, 4-6
 - sending a file in MAIL with, 8-7
 - sending a MAIL message with, 8-5
 - writing a file in EDT with, 7-2
 - CTRL keys, 3-15, 3-18
 - Cursor control
 - in EDT, 7-8, 7-12
 - CUT command (EDT), 7-21
- ## D
- \$DEFAULTS\$ buffer, 6-18
 - Data
 - logical, 12-10, 12-16
 - numeric, 12-8, 12-14
 - passing to command procedure, 13-9
 - Date
 - See also Absolute time
 - See also Combination time
 - See also Delta time
 - specifying absolute and delta date and time combinations, 3-14
 - specifying absolute date and time, 3-12
 - specifying delta date and time, 3-13
 - DCL (DIGITAL Command Language)
 - definition, 2-1

Index-4

- DCL command
 - interrupting or canceling
 - with CTRL/C, 3-16
 - with CTRL/Y, 3-16
 - interrupting with CTRL/T, 3-16
 - recalling
 - with CTRL/B, 3-16
 - with Down arrow key, 3-16
 - with Up arrow key, 3-16
 - using in EVE, 6-33
- DEASSIGN command (DCL), 11-4
 - and process logical name table, 11-5
- DECK command (DCL), 13-9
- DECnet
 - See also Network
 - logging in to remote systems with, 1-2
- DECnet-VAX
 - access violation, 4-6
 - and logical node name, 11-14
 - file manipulation with, 4-6
- Default
 - definition, 2-4
- Default protection, 4-9
- Default settings, 6-15
- Default values
 - provided by system, 3-5, 3-6
- DEFINE/KEY command (MAIL), 8-13
- DEFINE command (DCL), 11-2
 - and process logical name table, 11-5
 - example with access mode qualifier, 11-11
 - specifying the access mode with, 11-11
- DEFINE KEY command (EDT), 7-42
- DEFINE MACRO command (EDT), 7-44
- DEL C command (EDT), 7-14
- DEL EOL command (EDT), 7-16
- DELETE/SYMBOL command (DCL), 12-3
- DELETE command (DCL), 4-8
 - and wildcard characters, 4-8
- DELETE command (MAIL), 8-11
- DELETE key, 3-17
- DEL L command (EDT), 7-15
- Delta time
 - combined with absolute time, 3-14
 - default values, 3-13
- Delta time (cont'd.)
 - examples, 3-14
 - rules for entering, 3-13
 - syntax, 3-13
- DEL W command (EDT), 7-15
- Detached process
 - batch job as, 1-3
 - definition, 1-7
- Device, 1-6
 - mass storage, 1-6
 - record oriented, 1-6
 - unit record, 1-6
- Device code
 - definition, 1-7
- Device field
 - in full file specification, 1-5
- Device name
 - See also Device field
 - See also Physical device name
 - generic, 1-7
 - logical name equated to, 1-7
 - rules for entering, 1-6
- DIGITAL Command Language
 - See DCL
- directory
 - definition, 5-1
- DIRECTORY command (DCL), 5-3
- DIRECTORY command (MAIL), 8-4, 8-10
- Directory field
 - in full file specification, 1-5
 - using an asterisk wildcard character in, 4-5
 - using a percent sign wildcard character in, 4-5
- Directory file
 - See also Directory structure
 - creating, 5-3
 - default, 1-5, 5-4
 - definition, 1-5
 - deleting, 5-5
 - login, 1-5
 - named format, 5-3
 - protection, 5-6
 - setting default to another, 5-4
 - top level, 1-5

- Directory name
 - named format in a file specification, 5-3
 - replacing
 - with the ellipsis (...) wildcard character, 5-7
 - with the hyphen (-) wildcard character, 5-8
- Directory structure
 - master file directory in, 1-5
 - sample, 5-1
 - subdirectory in, 1-5
 - top level directory in, 1-5
 - user file directory in, 1-5
- Disk, 1-6
 - See also Device
 - contents of, 1-5, 1-6
- Displaying EVE command list, 6-2
- Distribution list
 - creating in MAIL, 8-6
- Do key
 - entering commands with, 6-7
 - recalling EVE command with, 6-7
- Down arrow key
 - recalling commands with, 3-10, 3-16

E

- EBCDIC
 - collating sequence, 9-3
- EDIT/EDT command (DCL)
 - /READ_ONLY qualifier to, 4-8
- EDIT/TPU command
 - EVE
 - using /RECOVER qualifier with, 6-13
 - invoking EVE with, 6-1
- EDIT/TPU command (DCL)
 - /READ_ONLY, 4-8
- EDIT command (DCL), 7-1
- Editing session
 - beginning, 6-1
 - EVE
 - recovering after system interruption, 6-13
 - Editing session (cont'd.)
 - exiting from EDT, 7-2
 - recovering EDT after system interruption, 7-7
 - refreshing screen display during EDT, 7-7
 - refreshing the screen during, 6-12
 - EDT, defining keys in, 7-27
 - EDT conversion, 6-33
 - EDT editor
 - as default MAIL editor, 8-14
 - buffer
 - commands for using, 7-25, 7-27
 - definition, 7-1
 - changing modes in, 7-6
 - cursor control in, 7-8, 7-12
 - defining macros in, 7-44
 - displaying a file with, 7-3
 - exiting from, 7-2
 - invoking, 7-1
 - keypad commands, 7-3
 - line-editing commands, 7-3, 7-6
 - reading a file with, 7-25
 - recovering session after system interruption, 7-7
 - replacing text with, 7-19
 - setting screen display in, 7-42
 - writing text to a file with, 7-25
 - Ellipsis (...) wildcard character
 - in a directory name, 5-7
 - ENTER command (EDT), 7-6
 - EOB (End-of-buffer) symbol, 7-2
 - EOD command (DCL), 13-9
 - EOL command (EDT), 7-9
 - Equivalence name
 - definition, 1-10
 - Error message
 - description of, 2-4
 - EVE
 - status line
 - definition of, 6-2
 - EVE, defining keys in, 6-18
 - EVE command line
 - correcting mistakes on, 6-6
 - EVE commands
 - HELP command, 6-2

Index-6

EVE commands (cont'd.)

- recalling, 6-7
- REPEAT command, 6-7
- SPAWN command, 6-33

EVE editing keys

- Find key, 6-9

EVE editing session

- ending, 6-2

EVE editor

- as default MAIL editor, 8-14
- buffer
 - definition of, 6-2
- creating buffers, 6-14
- creating subprocess, 6-33
- editing command lines, 6-6
- editing session
 - saving text created during, 6-2
- entering commands, 6-4, 6-7
- finding text, 6-9
- getting started with, 6-1
- leaving subprocess, 6-33
- message window
 - description of, 6-2
- modes of editing, 6-2
- reaching DCL, 6-33
- reading batch job log file with, 10-8
- reading file into buffer, 6-14
- replacing text, 6-11
- window
 - definition of, 6-2

Executable image

See Image

Execute procedure (@)

- executing command procedure
 - interactively with, 13-3

EXIT command (DCL), 13-5

EXIT command (EDT), 7-2

EXIT command (EVE), 6-3

Expression

- character, 12-12
- definition, 12-11
- logical, 12-16
- numeric, 12-14
- rules for determining the value of, 12-20
- string comparison operators, 12-12

Expression (cont'd.)

- summary of operators, 12-19

EXTRACT command (MAIL), 8-8

F

F\$ELEMENT lexical function, 13-9

F\$ENVIRONMENT lexical function, 13-26

F\$EXTRACT lexical function, 13-8

F\$GETJPI lexical function, 13-25

F\$SEARCH lexical function, 13-19

F6 through F14 keys, 3-15, 3-18

File

See also Directory file

copying, 4-6

- between nodes, 4-6

- with access control string, 4-12

creating in command procedure, 13-18

definition, 4-1

editing in command procedure, 13-21

merging, 9-5

- and sequence checking, 9-6

merging multiple, 1-9

open file quota, 13-25

purging, 4-9

reading from command procedure, 13-20

renaming, 4-7

sorting, 1-9

writing in command procedure, 13-18

FILE command (MAIL), 8-9

File name

See also File name field

definition, 4-2

rules for entering, 4-2

valid characters in, 4-2

File name field

in full file specification, 1-5

using an asterisk wildcard character in, 4-5

using a percent sign wildcard character in, 4-5

File protection, 4-9

See also Protection

File specification

See also Wildcard character
 as a search list, 11-13
 device field in, 1-5
 directory field in, 1-5
 file name field in, 1-5
 file type field in, 1-5
 file version number field in, 1-5
 format, 1-5
 logical name in, 1-10, 11-1
 node field in, 1-5
 node name in, 4-6

File type

definition, 4-2
 list of default, 4-2
 rules for entering, 4-2

File type field

asterisk wildcard character in, 4-5
 in full file specification, 1-5
 using a percent sign wildcard character
 in, 4-5

File version number

definition, 4-4

File version number field

in full file specification, 1-5
 using an asterisk wildcard character in,
 4-5

FILL command (EDT), 7-23, 7-24

FIND command (EDT), 7-17

FIND command (EVE)

using, 6-9

Find key

using in EVE, 6-9

FNDNXT command (EDT), 7-19

Foreign command

definition, 1-3

Function keys, 3-15, 3-18

G**Generic device name**

definition, 1-7

GET FILE command

creating a new buffer with, 6-14
 reading file into buffer with, 6-14

Global symbol

command levels available to, 12-4
 creating in command procedure, 13-15

Global symbol table

DCL reserved symbols, 12-5
 definition, 12-5

GOLD key

in EDT, 7-3

Group logical name table

definition, 11-6
 logical name for, 11-6

H

HELP command (DCL), 2-5

HELP command (EDT), 7-5

HELP command (EVE), 6-2

displaying command list, 6-2
 with specific command, 6-2

HELP command (MAIL), 8-2

HELP facility

EDT, 7-5

Hyphen

and command line continuation, 3-6

Hyphen (-) wildcard character

in a directory name, 5-8

I**I/O error**

in command procedures, 13-24

Image

See also Command image

definition, 1-7, 1-8

noncommand, 1-8

See also Foreign command, 1-3

INCLUDE command (EDT), 7-25

INCLUDE FILE command

reading file into buffer with, 6-14

Initialization file

for default settings, 6-17

Input stream

definition, 11-5

INQUIRE command (DCL), 13-13

Index-8

Integer

See Number

Interactive mode

definition, 1-3

Interactive utility

See Utility

Iterative translation

See also Logical name translation

J

Job logical name

definition, 11-6

in a job tree, 11-6

Job logical name table

list of default contents of, 11-6

logical name for, 11-6

Job tree

definition, 11-5

Journal file

EDT, 7-7

EVE, 6-12

Journaling facility

EVE

using to recover editing session,
6-13

Journaling facility (EVE)

restrictions on using, 6-13

K

Key

See also Key definition

function, 3-15, 3-18

sort, 9-3

Key definition

assigning, 3-15

in EDT, 7-42

Key names (EVE), 6-7

Keypad

default editing keys for EVE, 6-4

EDT, 7-3

MAIL diagram, 8-13

Keyword

definition, 1-4, 3-4

L

Label

definition, 1-4, 3-3

in DCL command line, 1-3, 3-3

Left arrow key

moving cursor with, 3-17

Lexical function

definition, 12-9

evaluating, 12-10

invoking, 12-9

list of functions used to save and
restore process characteristics,
13-26

symbol substitution in, 12-6

syntax, 12-9

using in command procedure, 12-9,
13-16

LINE command (EDT), 7-10

LINEFEED key, 3-17

LNLM\$GROUP, 11-6

LNLM\$JOB, 11-6

LNLM\$PROCESS, 11-5

LNLM\$PROCESS_DIRECTORY, 11-8

LNLM\$SYSTEM, 11-7

LNLM\$SYSTEM_DIRECTORY, 11-8

Local node

definition, 1-2

Local symbol, 12-4

Local symbol table

definition, 12-5

P1 through P8, 12-5

Log file

for batch job, 10-8

Logical name

See also Job logical name

See also Logical name table

See also Process logical name

access mode, 11-11

as device name, 1-7

defined as a search list, 11-13

equivalence name, 1-10

Logical name (cont'd.)

- for a mounted disk or tape, 11-6
- for a network, 11-14
- for a node specification, 11-14
- overview, 1-10
- preventing definition in subprocesses, 10-7
- process-permanent, 11-15, 11-16
- rules for creating, 11-2
- search lists, 11-13
- system-created, 11-15
- system-permanent, 11-18
- translation in file specifications, 11-2

Logical name directory table

- definition, 11-8
- process, 11-8
- system, 11-9

Logical name table

- See also Group logical name table
- See also Job logical name table
- See also Process logical name table
- See also System logical name table
- creating, 11-12
- defining access mode, 11-11
- definition, 11-4
- deleting, 11-12
- list of system-provided, 11-4
- process-private, 11-4
- search order, 11-4
- shareable, 11-6
 - definition, 11-4

Logical name translation

- and wildcards, 11-13

Logical operators, 12-13**Login**

- manual, 2-1
- network, 2-2

LOGIN.COM

- See Login command procedure

Login command procedure

- personal, 13-5
 - defining logical names in, 1-10
 - definition, 1-11, 13-5
 - executed as batch jobs, 10-8
 - location of, 13-5
 - sample, 13-6

Login command procedure (cont'd.)

- system, 1-11

Login directory file, 1-5**Logout, 2-8**

- network, 2-7

LOGOUT command (DCL), 2-8, 10-5**M****MAIL command (DCL), 8-2****MAIL folder**

- creating, 8-9
- deleting, 8-10
- displaying list of, 8-10
- MAIL, 8-3
- NEWMAIL, 8-2
- selecting, 8-10
- WASTEBASKET, 8-11

Mail subdirectory

- creating, 8-12

Mail Utility (MAIL)

- creating mail files, 8-11
- deleting a message in, 8-11
- extracting a message to a file with, 8-8
- keypad
 - commands, 8-12
 - diagram, 8-13
- reading a message in, 8-2
- sending a file from DCL level with, 8-7
- sending a file in MAIL with, 4-6, 8-7
- sending a message over network with, 8-5
- sending a message to a distribution list with, 8-6
- setting default editor in, 8-14
- using text editor in, 8-13

Mass storage device

- definition, 1-6

MERGE command (DCL), 1-9, 9-5

- See also Sort/Merge Utility

MFD (master file directory), 1-5

- See also Directory structure

MOUNT command (DCL), 1-7

Index-10

MOVE command (MAIL), 8-9
Multinational collating sequence, 9-4

N

Named directory specification
 definition, 5-3
 format in a file specification, 5-3
 rules for entering, 5-3
Network
 executing programs across, 10-7
 login, 2-2
 logout, 2-7
 sending mail over, 8-5
Network file specification
 See File specification
NEW command
 writing buffer to a file using, 6-15
Node field
 in full file specification, 1-5
Node name
 using a logical name, 11-14
Noncommand image, 1-8
Nondefinable keys, 6-29
Number
 as fraction, 12-9
 assigning to a symbol, 12-8
 converting to a string value, 12-16
 evaluation of, 12-5
 in an expression, 12-14
 integer values recognized by DCL,
 12-8
 internal storage of, 12-9

O

Offset
 definition, 12-17
OPEN command
 creating a new buffer with, 6-14
 reading file into buffer with, 6-14
OPEN command (DCL), 13-18
OPEN LINE command (EDT), 7-10
OPEN SELECTED command
 creating a new buffer with, 6-14

OPEN SELECTED command (cont'd.)
 reading file into buffer with, 6-14

Operator
 character string, 12-12
 concatenation, 12-12
 definition, 12-11
 logical, 12-13, 12-16
 numeric, 12-14
 order of evaluation, 12-19
 reduction, 12-12
 string comparison, 12-12

Output stream
 definition, 11-5
Overlay, numeric, 12-17, 12-18

P

P1 through P8, 12-5
PAGE command (EDT), 7-11
Parameter
 definition, 1-4, 3-4
 in DCL command line, 1-3, 3-3
 passing to a command procedure,
 12-5, 13-10
 rules for entering, 3-7
 syntax, 3-7
Parameter list
 syntax, 3-7
Parameter qualifier
 definition, 3-8
Parent process
 definition, 1-7
Password
 changing, 2-3
 creating, 2-3
PASTE command (EDT), 7-21
Percent sign (%) wildcard character
 rules for using, 4-5
Physical device name
 controller designation field, 1-6
 device code field, 1-6
 format in a file specification, 1-6
 unit number field, 1-6
PID (process identification number)
 and process context, 10-2

Positional qualifier
 definition, 3-8
 rules for entering, 3-8

PRINT command (DCL), 4-12

Print job, 4-11
 delaying, 4-12
 list of DCL commands to use with,
 4-12
 obtaining multiple copies of, 4-12
 priorities, 4-11

Print queue
 and print job execution, 4-11
 controlling, 4-12

Process
 and job tree, 10-3
 checking status with CTRL/T, 2-5
 creating, 10-1
 definition, 1-7, 10-1
 types of, 10-1

Process characteristics
 lexical functions used to save and
 restore, 13-26
 obtained from UAF, 1-7

Process context
 list of characteristics, 10-1

Process directory logical name table
 list of default contents of, 11-8

Process identification number
 See PID

Process logical name
 in a job tree, 11-5

Process logical name table
 definition, 11-5
 list of default contents in, 11-5
 logical name for, 11-5

Process-permanent logical names
 list of, 11-15

Process privilege
 and process context, 10-3

Process rights identifier
 and process context, 10-3

Program
 as batch job, 1-3
 command image, 1-8
 definition, 1-8

Program (cont'd.)
 executing
 across network, 10-7
 noncommand image, 1-8

Prompt
 system in a command line, 3-5, 3-6

Protection, 1-11
 default, 4-9
 directory, 5-6
 file, 4-9

PURGE command (DCL), 4-9

Q

Qualifier
 abbreviating, 3-8
 command, 3-8
 default values, 3-9
 definition, 1-4, 3-4
 format, 3-9
 in DCL command line, 1-3, 3-3
 parameter, 3-9
 positional, 3-8
 rules for entering, 3-8
 types of, 3-8, 3-9

Qualifier values
 date and time formats, 3-12
 rules for entering, 3-12
 types of, 3-12

QUIT command (EDT), 7-2
 QUIT command (EVE), 6-3

R

Radix
 specifying in symbol assignment, 12-9

READ command (DCL), 13-14, 13-20

READ command (MAIL), 8-2

RECALL command (DCL), 3-10

Recalling commands, 3-10, 3-11

Record
 deleting in command procedure, 13-23
 modifying in command procedure,
 13-21

Index-12

Record (cont'd.)

writing from command procedure,
13-23

Record-oriented device

definition, 1-6

Record sort, 9-1

/RECOVER qualifier

EDIT/TPU command, 6-13

Remote node

definition, 1-2

printing file on, 4-12

RENAME command (DCL), 4-7

REPEAT command (EVE), 6-7

REPLACE command (EVE)

case sensitivity of, 6-11

using, 6-11

RESET command (EDT), 7-21

\$RESTART global symbol, 12-6

RETURN key, 3-15

Right arrow key

moving cursor with, 3-17

RUN (Image) command (DCL), 1-8

RUN (Process) command (DCL), 10-1

S

SEARCH command (MAIL), 8-4

Search list

definition, 11-13

example, 11-13

in a file specification, 11-13

translation, 11-13

Search string

definition of, 6-9

SECT command (EDT), 7-11

SELECT command (EDT), 7-17

SELECT command (MAIL), 8-10

SEND/EDIT command (MAIL), 8-14

SEND command (MAIL), 8-5, 8-7

SET DEFAULT command (DCL), 5-4

SET EDITOR command (MAIL), 8-14

SET ENTITY command (EDT), 7-43

SET HOST command (DCL), 2-2

SET LINES command (EDT), 7-42

SET MODE command (EDT), 7-43

SET NUMBERS command (EDT), 7-17

SET PASSWORD command (DCL), 2-3

SET PROTECTION command (DCL),
4-9, 4-10

SET QUIET command (EDT), 7-43

SET SCREEN command (EDT), 7-42,
7-43

SET SEARCH command (EDT), 7-18,
7-20

SET TERMINAL command (DCL), 10-5

SET TRUNCATE command (EDT), 7-43

SET WRAP command (EDT), 7-23, 7-43

\$SEVERITY global symbol, 12-6

Shareable tables

group logical name table, 11-6

system logical name table, 11-7

SHOW BUFFER command (EDT), 7-26

SHOW BUFFERS command

using, 6-14

SHOW DEFAULT command (DCL), 5-5

SHOW ENTRY command (DCL), 4-11,
10-8

SHOW LOGICAL command (DCL)

and logical name access mode, 11-11

and logical name table structure, 11-8

SHOW PROCESS command (DCL), 10-1

SHOW QUEUE command (DCL), 4-11

SHOW SYSTEM BUFFERS command

using, 6-14

Sort

batch job, 9-5

character data, 9-3

collating sequence, 9-3

key, 9-3

single key, 9-2

terminal input, 9-4

types of, 9-1

Sort/Merge Utility (SORT/MERGE)

See also Sort

collating sequences

ASCII, 9-3

EBCDIC, 9-3

entering records from terminal with,
9-4

invoking, 1-9

merging files with, 9-5

Sort/Merge Utility (SORT/MERGE)
(cont'd.)

- sorting noncharacter data files with, 9-4
- sorting records with, 9-1
- SORT command (DCL), 1-9, 9-2
- See also Sort/Merge Utility
- SPAWN command
 - restriction on using, 6-33
 - using, 6-33
- SPAWN command (DCL), 10-3
- \$STATUS global symbol, 12-6
- STOP command (DCL), 13-5
- Subdirectory
 - creating, 5-3
 - definition, 1-5
 - setting default to another, 5-4
 - syntax, 5-3
- SUBMIT command (DCL), 10-8
- Subprocess, 1-7
 - and job tree, 10-3
 - and process identification number, 10-6
 - context, 10-6
 - creating, 6-33, 10-4
 - definition, 10-3
 - deleting, 10-5
 - exiting from, 10-5
 - leaving, 6-33
- SUBSTITUTE command (EDT), 7-19
- Substring replacement, 12-17
- Symbol
 - as another symbol, 12-11
 - as foreign command, 1-3
 - assignment, 12-4
 - creating, 12-4
 - defined as a lexical function, 12-9
 - defining in command procedure, 13-13
 - definition, 1-10
 - deleting, 12-3
 - displaying, 12-3
 - in command procedure, 13-16
 - evaluation, 12-5
 - global, 12-4
 - indicating a numeric value, 12-5, 12-8
 - local, 12-4

Symbol (cont'd.)

- logical data, 12-10
- numeric overlay with, 12-18
- passing to a command procedure, 13-11
- precedence, 12-5
- preventing assignment in subprocesses, 10-7
- substitution, 12-6
 - automatic, 12-6
 - forced, 12-6
 - order of, 12-7
- substring replacement with, 12-17
- used as variable, 12-4
- used in expressions, 12-7, 12-8, 12-11
- uses of, 12-3
- Symbol table
 - See Local symbol table, Global symbol table
- SYS\$BATCH, 10-8
- SYS\$COMMAND
 - redefining, 11-18
- SYS\$ERROR
 - redefining, 11-17
- SYS\$INPUT
 - redefining, 11-16
 - in command procedure, 13-14
- SYS\$OUTPUT
 - redefining, 11-16
- SYS\$PRINT, 4-11
 - and batch job log files, 10-8
- System directory logical name table
 - list of default contents of, 11-9
- System interruption
 - recovering editing session after, 6-13
- System logical name table
 - definition, 11-7
 - list of default contents of, 11-7
 - logical name for, 11-7
- System-permanent logical name, 11-18

T

- TAB key, 3-17

Index-14

Terminal (cont'd.)

display

stopping and starting, 3-18

I/O

in command procedure, 13-14,
13-16

Time

See also Absolute time

See also Combination time

See also Delta time

specifying absolute and delta date and
time combinations, 3-14

specifying absolute date and time,
3-12

specifying delta date and time, 3-13

TOP command (EDT), 7-11

Top level directory file, 1-5

TPU editor

as default MAIL editor, 8-14

TYPE command (DCL), 13-17

and wildcard characters, 4-8

displaying files with, 4-7

executing command procedure on
remote node with, 13-3

U

UAF (user authorization file)

and process characteristics, 1-7

UFD (user file directory)

See also Directory structure

contents of, 1-5

location of, 1-6

UIC (user identification code)

and process context, 10-3

default protection, 4-10

UND C command (EDT), 7-14

UND L command (EDT), 7-16

UND W command (EDT), 7-15

Unit number field

definition, 1-7

Unit record device

definition, 1-6

Up arrow key

recalling commands with, 3-10, 3-16

User authorization file

See UAF

/USER_MODE qualifier

redefining SYS\$INPUT with, 11-16

redefining SYS\$OUTPUT with, 11-17

Utility

definition, 1-8

types of, 1-8

V

Value

definition, 1-4, 3-4

in DCL command line, 1-3, 3-3

W

Wildcard

matching file names with, 6-14

Wildcard character

asterisk (*), 4-5

ellipsis (...), 5-7

hyphen (-), 5-8

in file specifications containing logical
names, 11-13

percent sign (%), 4-5

WORD command (EDT), 7-8

WPS Ruler key

for tab stops, 6-36

WRITE command (DCL), 13-16, 13-18

WRITE command (EDT), 7-25

WRITE FILE command

restriction with journaling facility,
6-13

writing buffer to a file using, 6-15

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local DIGITAL subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	_____	Local DIGITAL subsidiary or approved distributor
Internal ¹	_____	SDC Order Processing - WMO/E15 or Software Distribution Center Digital Equipment Corporation Westminster, Massachusetts 01473

¹For internal orders, you must submit an Internal Software Order Form (EN-01740-07).



Reader's Comments

VMS User's Manual
AA-LA98B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less _____

What I like best about this manual is _____

What I like least about this manual is _____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

I am using **Version** _____ of the software this manual describes.
Name/Title _____ Dept. _____
Company _____ Date _____
Mailing Address _____
_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™



No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987



Do Not Tear - Fold Here

Cut Along Dotted Line

Reader's Comments

VMS User's Manual
AA-LA98B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less _____

What I like best about this manual is _____

What I like least about this manual is _____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

I am using **Version** _____ of the software this manual describes.
Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

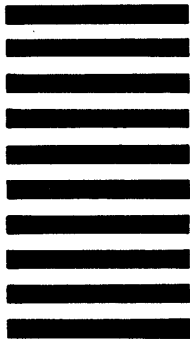
_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™



No Postage
Necessary
if Mailed
in the
United States



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987



Do Not Tear - Fold Here

Cut Along Dotted Line