OS/8
System Reference Manual

Order No. AA-H607A-TA

ABSTRACT

This document describes OS/8 system conventions, keyboard commands, and utility programs.

SUPERSESSION/UPDATE INFORMATION: This manual supersedes sections of OS/8 Handbook (DEC-S8-OSHBA-A-D) and the OS/8 Handbook Update (DEC-S8-OSHBA-A-DN4).

OPERATING SYSTEM AND VERSION: OS/8 V3D

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, March 1979

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

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

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

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

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

CONTENTS

			Page
DOCUMENTATION SET FOR OS/8			xiii
CHAPTER	1	OVERVIEW	1-1
	1.1	INTRODUCTION TO OS/8	1-1
	1.2	OS/8 I/O DEVICES	1-2
	1.3	HARDWARE CONFIGURATIONS	1-3
	1.4	SYSTEM SOFTWARE COMPONENTS	1-4
CHAPTER	2	SYSTEM CONVENTIONS	2-1
	2.1	PERMANENT DEVICE NAMES	2-1
	2.2	FILE NAMES AND EXTENSIONS	2-2
	2.3	UNITS OF STORAGE	2-4
CHAPTER	3	OS/8 KEYBOARD COMMANDS	3-1
	3.1	INTRODUCTION	3-1
	3.2	COMMAND FORMAT	3-4
	3.3	COMMAND AND FILE OPTIONS	3-4
	3.3.1	Command Options	3-5
	3.3.1.1	The Slash Construction -- Single-Letter Options	3-5
	3.3.1.2	The Parenthesis Construction -- Multiple-Letter Options	3-5
	3.3.1.3	The Equal Sign Construction -- Octal Number Options	3-5
	3.3.2	File Options	3-5
	3.3.3	Additional Switch Options -- the Dash Construction	3-6
	3.4	COMMANDS THAT REMEMBER FILE SPECIFICATIONS	3-6
	3.5	USING WILDCARDS	3-7
	3.5.1	Wildcards in Input Filenames	3-7
	3.5.2	Wildcards in Output Specifications	3-7
	3.5.3	Warnings and Suggestions	3-8
	3.6	INDIRECT COMMANDS	3-8
	3.7	USING DEFAULTS	3-9
	3.8	GETTING HELP -- THE HELP COMMAND	3-9
	3.9	ENTERING A COMMAND LINE -- CORRECTING AND PREVENTING ERRORS	3-10
	3.10	ASSIGN	3-11
	3.10.1	Canceling a Logical Name	3-11
	3.10.2	Checking for Duplicate Names	3-11
	3.11	BACKSPACE	3-12
	3.12	BASIC	3-13
	3.13	BOOT	3-14
	3.14	CCL	3-15
	3.15	COMPARE	3-16
	3.15.1	COMPARE Output	3-16
	3.15.2	COMPARE Options	3-17

		Page
3.16	COMPILE	3-19
3.16.1	COMPILE Input	3-19
3.16.2	COMPILE Output	3-20
3.16.2.1	Output File 1 -- the Binary Code	3-20
3.16.2.2	Output File 2 -- the Listing File	3-21
3.16.2.3	COMPILE Options and Errors	3-21
3.17	COPY	3-22
3.17.1	COPY Input	3-22
3.17.2	COPY Output	3-22
3.17.3	COPY Terminal Display	3-22
3.17.4	Predeletion and Postdeletion	3-23
3.17.5	COPY Options	3-23
3.18	CREATE	3-24
3.19	CREF	3-25
3.19.1	CREF Options	3-25
3.20	DATE	3-26
3.21	DEASSIGN	3-27
3.22	DELETE	3-28
3.22.1	The Conditional DELETE	3-28
3.22.2	DELETE Terminal Display	3-29
3.22.3	DELETE Options	3-29
3.23	DIRECT	3-30
3.23.1	DIRECT Output	3-30
3.23.2	DIRECT Options	3-30
3.24	DUPLICATE	3-32
3.25	EDIT	3-33
3.25.1	Recalling Arguments	3-33
3.26	EOF	3-34
3.27	EXECUTE	3-35
3.28	GET	3-36
3.29	HELP	3-37
3.30	LIST	3-38
3.30.1	LIST Options	3-38
3.31	LOAD	3-39
3.32	MAKE	3-40
3.33	MAP	3-41
3.33.1	MAP Output	3-41
3.33.2	MAP Options	3-41
3.34	MEMORY	3-43
3.35	MUNG	3-44
3.36	ODT	3-45
3.37	PAL	3-46
3.38	PRINT	3-47
3.39	PUNCH	3-48
3.40	R	3-49
3.41	RENAME	3-50
3.41.1	RENAME Options	3-50
3.42	RES	3-51
3.43	REWIND	3-52
3.44	RUN	3-53
3.45	SAVE	3-54
3.45.1	The Job Status Word	3-55
3.46	SET	3-56
3.47	SKIP	3-57
3.48	START	3-58
3.49	SQUISH	3-59
3.50	SUBMIT	3-60

			Page
	3.51	TECO	3-61
	3.52	TERMINATE	3-62
	3.53	TYPE	3-63
	3.53.1	TYPE Options	3-63
	3.54	UA, UB, and UC	3-64
	3.55	UNLOAD	3-65
	3.56	VERSION	3-66
	3.57	ZERO	3-67
CHAPTER	4	THE OS/8 SYMBOLIC EDITOR	4-1
	4.1	INTRODUCTION	4-1
	4.2	CALLING THE EDITOR	4-1
	4.3	MODES OF OPERATION	4-2
	4.3.1	Text Mode	4-3
	4.3.2	Command Mode	4-4
	4.3.2.1	Input Commands	4-5
	4.3.2.2	Listing Commands	4-6
	4.3.2.3	Output Commands	4-6
	4.3.2.4	Editing Commands	4-8
	4.3.2.5	Search Commands	4-9
	4.3.2.6	Special Command Mode Characters	4-10
	4.4	SEARCHING A TEXT	4-12
	4.4.1	Single-Character Search -- the S Command	4-12
	4.4.2	The Character String Search	4-13
	4.4.2.1	Intrabuffer String Search	4-13
	4.4.2.2	Interbuffer String Search -- J Command	4-16
	4.5	EDITOR OPTIONS	4-17
	4.6	EDITOR ERROR MESSAGES	4-18
	4.7	SUMMARY OF EDITOR COMMANDS AND SPECIAL CHARACTERS	4-19
CHAPTER	5	THE COMMAND DECODER	5-1
	5.1	ENTERING I/O SPECIFICATIONS	5-1
	5.2	COMMAND DECODER ERROR MESSAGES	5-3
	5.3	THE CCL AND THE COMMAND DECODER	5-3
CHAPTER	6	BATCH	6-1
	6.1	INTRODUCTION	6-1
	6.2	BATCH PROCESSING UNDER OS/8	6-1
	6.2.1	Input Files	6-2
	6.2.2	Output Files	6-2
	6.2.3	I/O Devices	6-2
	6.2.4	Spooling	6-2
	6.2.5	Entering File Specifications	6-2
	6.3	BATCH MONITOR COMMANDS	6-4
	6.3.1	Defining a BATCH Job	6-4
	6.3.2	Using OS/8 Keyboard Commands	6-5
	6.3.3	Using the Command Decoder	6-6
	6.3.4	Additional Features	6-6
	6.4	THE BATCH INPUT FILE	6-7
	6.5	BATCH ERROR MESSAGES	6-9
	6.6	RUNNING BATCH FROM PUNCHED CARDS	6-11
	6.7	RESTRICTIONS UNDER OS/8 BATCH	6-12
	6.8	BATCH DEMONSTRATION PROGRAM	6-13

			Page
	6.9	LOADING AND SAVING BATCH	6-18
	6.10	LOADING AND SAVING PROGRAMS FOR USE UNDER BATCH	6-18
	6.11	TRANSFERRING THE SYSTEM SOFTWARE FROM CASSETTE TO THE SYSTEM DEVICE	6-19
	6.12	RUNNING FORTRAN IV UNDER BATCH IN 32K	6-21
CHAPTER	7	BITMAP	7-1
	7.1	FILE AND DEVICE SPECIFICATIONS	7-1
	7.2	BITMAP OUTPUT	7-2
	7.3	BITMAP ERROR MESSAGES	7-3
	7.4	ASSEMBLY INSTRUCTIONS	7-3
CHAPTER	8	BOOT	8-1
	8.1	BOOTING WITH BOOT	8-1
	8.2	BOOT PRIORITIES	8-2
CHAPTER	9	BUILD	9-1
	9.1	OS/8 DEVICE HANDLERS	9-1
	9.1.1	Cassette Systems	9-3
	9.1.2	Paper Tape Systems	9-3
	9.2	CALLING AND USING BUILD	9-6
	9.3	BUILD COMMANDS	9-7
	9.3.1	The Hyphen Construction	9-8
	9.3.2	PRINT	9-8
	9.3.3	QLIST	9-9
	9.3.4	LOAD	9-9
	9.3.5	INSERT	9-10
	9.3.6	DELETE	9-11
	9.3.7	REPLACE	9-12
	9.3.8	UNLOAD	9-13
	9.3.9	NAME	9-13
	9.3.10	ALTER	9-14
	9.3.11	EXAMINE	9-15
	9.3.12	DSK	9-15
	9.3.13	CORE	9-16
	9.3.14	DCB	9-16
	9.3.15	CTL	9-17
	9.3.16	VERSION	9-17
	9.3.17	SIZE	9-17
	9.3.18	SYSTEM	9-17
	9.3.19	BUILD	9-18
	9.3.20	BOOTSTRAP	9-19
	9.4	BUILD ERROR MESSAGES	9-20
	9.5	BUILD DEVICE HANDLER FORMAT	9-21
	9.5.1	Header Block	9-22
	9.5.2	Descriptor Block	9-22
	9.5.3	Breakdown of DCB Word	9-23
	9.5.4	Entry Point Offset	9-24
	9.6	CREATING A SYSTEM HANDLER	9-25
CHAPTER	10	CASSETTE AND MAGNETIC TAPE POSITIONER (CAMP)	10-1

			Page
	10.1	CAMP COMMANDS	10-1
	10.1.1	BACKSPACE Command	10-1
	10.1.2	EOF Command	10-2
	10.1.3	HELP Command	10-2
	10.1.4	REWIND Command	10-3
	10.1.5	SKIP Command	10-3
	10.1.6	UNLOAD Command	10-4
	10.1.7	VERSION Command	10-4
	10.2	CAMP ERROR MESSAGE SUMMARY	10-5
CHAPTER	11	CROSS-REFERENCE PROGRAM (CREF)	11-1
	11.1	CALLING AND USING CREF	11-1
	11.1.1	CREF Options	11-1
	11.1.2	Examples of CREF Usage	11-2
	11.2	PSEUDO-OP HANDLING	11-3
	11.3	INTERPRETING CREF OUTPUT	11-3
	11.4	RESTRICTIONS	11-5
	11.5	CREF ERROR MESSAGES	11-6
CHAPTER	12	DIRECT	12-1
	12.1	CALLING AND USING DIRECT	12-1
	12.1.1	DIRECT Options	12-2
	12.2	DIRECT EXAMPLES	12-3
	12.3	DIRECT ERROR MESSAGES	12-5
CHAPTER	13	DECTAPE COPY AND FORMAT PROGRAMS	13-1
	13.1	DTFRMT	13-1
	13.1.1	Loading Procedure	13-1
	13.1.2	Using the Program	13-1
	13.1.3	Error Messages	13-3
	13.1.4	Details of DTFRMT Operation and Storage	13-4
	13.2	TDFRMT	13-5
	13.2.1	Operating Procedures	13-5
	13.2.2	Error Messages	13-7
	13.2.3	Details of TDFRMT Operation and Storage	13-8
	13.3	DTCOPY	13-10
	13.3.1	Error Messages	13-11
	13.4	TDCOPY	13-12
	13.4.1	Error Messages	13-13
	13.4.2	Details of Operation	13-15
CHAPTER	14	DUMP	14-1
	14.1	FORM FEEDS	14-2
	14.2	ADDING THE DUMP HANDLER TO YOUR SYSTEM	14-2
	14.3	FORMAT OF THE DUMP	14-2
CHAPTER	15	EPIC	15-1
	15.1	LOADING EPIC	15-1
	15.2	RESTART PROCEDURE	15-2
	15.3	PAPER TAPE FACILITY	15-2
	15.4	COMMAND FORMAT	15-2
	15.5	DEFAULT OPTIONS	15-3

			Page
	15.6	ERROR CONDITIONS	15-4
	15.7	LOW SPEED I/O	15-4
	15.8	DEVICE CODES	15-4
	15.9	EDITING CAPABILITY	15-5
	15.9.1	Initial Command Format	15-5
	15.9.2	Editing Commands	15-5
	15.10	COMPARE CAPABILITY	15-8
	15.11	ERROR MESSAGES	15-8
	15.12	PAPER TAPE FORMAT	15-10
	15.13	LOADING EPIC FROM PAPER TAPE	15-11
	15.14	EPIC ASSEMBLY INSTRUCTIONS	15-11
CHAPTER	16	FILE-ORIENTED TRANSFER PROGRAM (FOTP)	16-1
	16.1	CALLING FOTP	16-1
	16.1.1	Input Specifications	16-1
	16.1.2	Output Specifications	16-3
	16.2	USING FOTP	16-3
	16.2.1	Additional FOTP Commands	16-5
	16.2.2	Advantages of Predeletion	16-7
	16.2.3	Advantages of Postdeletion	16-7
	16.2.4	Control Characters	16-7
	16.3	FOTP OPTIONS	16-7
	16.3.1	Examples of FOTP Specification Commands	16-10
	16.4	ERROR MESSAGES	16-11
CHAPTER	17	FUTIL	17-1
	17.1	INTRODUCTION	17-1
	17.1.1	Special Characters Used in FUTIL	17-1
	17.1.2	Running FUTIL	17-2
	17.1.3	Access Method	17-3
	17.1.4	Referencing Words on the Device	17-5
	17.1.5	Numeric Item (or Numbers)	17-6
	17.1.6	Errors and Error Messages	17-7
	17.2	SINGLE-CHARACTER (ODT-LIKE) COMMANDS	17-7
	17.2.1	Symbolic Output Formats	17-9
	17.3	WORD-TYPE COMMANDS	17-11
	17.3.1	Output Formats	17-12
	17.3.1.1	DUMP	17-13
	17.3.1.2	LIST	17-13
	17.3.1.3	MODIFY	17-14
	17.3.2	Search Limits	17-15
	17.3.2.1	WORD (Search)	17-16
	17.3.2.2	STRING (Search)	17-17
	17.3.2.3	SMASK	17-18
	17.3.2.4	SET	17-18
	17.3.2.5	SHOW	17-19
	17.3.2.6	FILE	17-20
	17.3.2.7	WRITE	17-22
	17.3.2.8	SCAN	17-22
	17.3.2.9	REWIND	17-22
	17.3.3	File Output	17-23
	17.3.3.1	OPEN	17-23
	17.3.3.2	CLOSE	17-24
	17.3.4	Batch Operation	17-24
	17.3.4.1	IF	17-24

			Page
	17.3.4.2	END	17-24
	17.3.4.3	COMMENT	17-25
	17.3.4.4	EXIT	17-25
	17.3.4.5	EVAL	17-25
	17.4	EXAMPLES	17-26
	17.5	PROGRAM EXECUTION AND MEMORY ALLOCATION	17-33
	17.6	COMMAND SUMMARY	17-34
	17.7	SINGLE-CHARACTER COMMAND OUTPUT FORMAT SUMMARY	17-35
CHAPTER	18	MAGTAPE/CASSETTE PERIPHERAL INTERCHANGE PROGRAM (MCPIP)	18-1
	18.1	CALLING AND USING MCPIP	18-1
	18.1.1	MCPIP Options	18-2
	18.2	MCPIP ERROR MESSAGES	18-4
CHAPTER	19	OCTAL DEBUGGING TECHNIQUE (ODT)	19-1
	19.1	FEATURES	19-1
	19.2	CALLING AND USING ODT	19-1
	19.3	COMMANDS	19-2
	19.3.1	Special Characters	19-2
	19.3.2	Illegal Characters	19-4
	19.3.3	Control Commands	19-4
	19.4	ADDITIONAL TECHNIQUES	19-4
	19.4.1	Current Location	19-7
	19.4.2	Indirect References	19-7
	19.5	ERRORS	19-7
	19.6	PROGRAMMING NOTES SUMMARY	19-7
	19.7	SUMMARY OF ODT COMMANDS	19-8
CHAPTER	20	PERIPHERAL INTERCHANGE PROGRAM (PIP)	20-1
	20.1	CALLING AND USING PIP	20-1
	20.1.1	PIP Options	20-1
	20.1.2	Examples of PIP Specification Commands	20-6
	20.2	ADDITIONAL INFORMATION WORDS IN FILE DIRECTORIES	20-8
	20.3	PIP ERROR MESSAGES	20-8
CHAPTER	21	PIP10	21-1
	21.1	CALLING AND USING PIP10	21-1
	21.2	HOW TO COPY LARGE FILES WITH PIP10 (SR)	21-2
	21.3	PIP10 OPTIONS	21-2
	21.4	PIP10 EXAMPLES	21-3
	21.5	ERROR MESSAGES	21-3
CHAPTER	22	RESOURCES (RESORC)	22-1
	22.1	CALLING AND USING RESORC	22-1
	22.2	RESORC OPTIONS	22-2
	22.2.1	Fast Mode (/F Option)	22-2
	22.2.2	Limited Mode (/L Option)	22-2
	22.2.3	Extended Mode (/E Option)	22-3
	22.3	RESORC ERROR MESSAGES	22-6

			Page
CHAPTER	23	RKLFMT DISK FORMATTER PROGRAM	23-1
	23.1	RUNNING THE PROGRAM	23-1
	23.2	STANDARD TEST PROCEDURES	23-2
	23.2.1	RK05J Drive Cartridge Mounting Procedure	23-2
	23.2.2	RK05F Drive Setup Procedure	23-2
	23.3	FORMAT PROGRAM	23-3
	23.4	ERRORS	23-4
	23.5	PROGRAM DESCRIPTION	23-4
	23.6	CONTROL CHARACTERS	23-5
	23.7	MISCELLANEOUS	23-5
	23.7.1	Waiting Message	23-5
	23.7.2	End of Pass	23-6
	23.7.3	Errors	23-6
	23.7.4	Location Changes	23-6
CHAPTER	24	RXCOPY PROGRAM	24-1
CHAPTER	25	SET PROGRAM	25-1
	25.1	TERMINAL ATTRIBUTES	25-3
	25.1.1	Arrow	25-3
	25.1.2	CODE n	25-3
	25.1.3	COLumn n	25-4
	25.1.4	ECHO	25-4
	25.1.5	ESCape	25-4
	25.1.6	FILL	25-5
	25.1.7	FLAG	25-5
	25.1.8	HEIGHT m	25-5
	25.1.9	LC	25-6
	25.1.10	PAGE	25-6
	25.1.11	PAUSE n	25-6
	25.1.12	SCOPE	25-7
	25.1.13	TAB	25-7
	25.1.14	WIDTH n	25-7
	25.2	CARD READER ATTRIBUTES	25-8
	25.2.1	CODE n	25-8
	25.3	MAGNETIC TAPE ATTRIBUTES	25-8
	25.3.1	PARITY x	25-8
	25.3.2	FILES	25-8
	25.4	SYSTEM ATTRIBUTES	25-9
	25.4.1	INITIAL xxxxx	25-9
	25.4.2	OS8	25-9
	25.4.3	OS78	25-9
	25.5	LINE PRINTER ATTRIBUTES	25-10
	25.5.1	LA78	25-10
	25.5.2	LA8A	25-10
	25.5.3	LC	25-10
	25.5.4	LV8E	25-10
	25.5.5	WIDTH n	25-11
	25.6	ANY DEVICE ATTRIBUTES	25-11
	25.6.1	FILES	25-11
	25.6.2	DVCode nn	25-12
	25.6.3	LOCation n=m or LOCation n	25-12
	25.6.4	READOnly	25-13
	25.6.5	VERSION x	25-13
	25.6.6	BLOCK b, LOCation n=m or BLOCK b, LOC n	25-13

CONTENTS (Cont.)

			Page
CHAPTER	26	SRCCOM	26-1
	26.1	SRCCOM ASSEMBLY INSTRUCTIONS	26-1
	26.2	LOADING SRCCOM	26-1
	26.3	SRCCOM OUTPUT	26-2
	26.4	ERROR MESSAGES	26-4
APPENDIX	A	CHARACTER CODES	A-1
APPENDIX	B	LOADING PROCEDURES	B-1
	B.1	INITIALIZING THE SYSTEM	B-1
	B.2	LOADERS	B-1
	B.2.1	Binary (BIN) Loader	B-6
APPENDIX	C	OS/8 DEMONSTRATION RUN	C-1
APPENDIX	D	OS/8 FILE NAME EXTENSIONS	D-1
APPENDIX	E	OS/8 DEVICE HANDLERS	E-1
	E.1	HIGH-SPEED READER/PUNCH	E-1
	E.2	LOW-SPEED READER/PUNCH	E-1
	E.3	TTY HANDLERS	E-1
	E.4	LINE PRINTERS	E-2
	E.5	VR12 SCOPE	E-2
	E.6	CARD READER	E-3
	E.7	DECTAPES	E-3
	E.8	MAGNETIC TAPE	E-3
	E.9	CASSETTES	E-3
	E.10	BATCH HANDLER	E-4
	E.11	DSK AND SYS	E-4
APPENDIX	F	OBTAINING OS/8 PROGRAM VERSION NUMBERS	F-1

FIGURES

FIGURE	6-1	Sample BATCH Input File	6-7
	6-2	Punched Card Input File	6-12
	B-1	Loading the RIM Loader	B-4
	B-2	Checking the RIM Loader	B-5
	B-3	Loading the BIN Loader	B-7
	B-4	Loading a Binary Tape Using BIN	B-9

TABLES

TABLE	2-1	Permanent Device Names	2-1
	2-2	OS/8 File Name Extensions	2-3
	3-1	Keyboard Monitor Error Messages	3-1
	3-2	Switch Options	3-6
	4-1	Editor Key Control Commands	4-2

Page

TABLES (Cont.)

TABLE	4-2	Editor Input Commands	4-5
	4-3	Editor Listing Commands	4-6
	4-4	Editor Output Commands	4-7
	4-5	Editing Commands: Deletion and Alteration	4-9
	4-6	Editor Search Commands	4-10
	4-7	Editor Special Characters: Command Mode	4-10
	4-8	Aborting Editor String Search Commands	4-17
	4-9	Nonfatal Editor Error Messages	4-18
	4-10	Editor Error Codes	4-19
	4-11	Editor Command and Special Characters	4-19
	5-1	Examples of Output to the Command Decoder	5-1
	5-2	Examples of Input to the Command Decoder	5-2
	5-3	Command Decoder Error Messages	5-3
	6-1	Run-Time Options	6-3
	6-2	BATCH Monitor Commands	6-5
	6-3	BATCH Error Messages	6-9
	7-1	Bitmap Options	7-2
	8-1	BOOT Mnemonics	8-2
	9-1	Standard DECtape System Device Handlers	9-2
	9-2	Standard Cassette System Device Handlers	9-3
	9-3	Standard Paper Tape System Device Handlers	9-3
	9-4	OS/8 Device Handlers	9-4
	9-5	BUILD Editing Characters	9-7
	9-6	BUILD Error Messages	9-20
	9-7	DCB Word	9-23
	10-1	CAMP Error Messages	10-5
	11-1	CREF Options	11-2
	11-2	CREF Error Messages	11-7
	12-1	DIRECT Options	12-2
	12-2	DIRECT Error Messages	12-5
	15-1	EPIC Commands	15-6
	15-2	EPIC Error Messages	15-9
	16-1	FOTP Options	16-8
	16-2	FOTP Error Messages	16-12
	18-1	MCPIP Options	18-2
	18-2	MCPIP Error Messages	18-4
	19-1	ODT Command Summary	19-8
	20-1	PIP Options	20-1
	20-2	PIP Error Messages	20-8
	21-1	PIP10 Error Messages	21-4
	22-1	RESORC Device Types	22-3
	22-2	RESORC Error Messages	22-6
	24-1	RXCOPY Options	24-1
	24-2	RXCOPY Error Messages	24-2
	25-1	SET Command Attributes	25-1
	25-2	SET Error Messages	25-2
	26-1	SRCCOM Run-Time Options	26-2
	A-1	ASCII Character Set	A-1
	B-1	RIM Loader for Low-Speed Reader	B-2
	B-2	RIM Loader for High-Speed Reader	B-3

DOCUMENTATION SET FOR OS/8

OS/8 SYSTEM GENERATION NOTES (AA-H606A-TA)

The System Generation Notes provide the information you need to get a new OS/8 system running.

OS/8 SYSTEM REFERENCE MANUAL (AA-H607A-TA)

The System Reference Manual describes OS/8 system conventions, keyboard commands, and utility programs.

OS/8 TECO REFERENCE MANUAL (AA-H608A-TA)

The TECO Reference Manual describes the OS/8 version of this character-oriented text editing and correcting program.

OS/8 LANGUAGE REFERENCE MANUAL (AA-H609A-TA)

The Language Reference Manual describes all languages supported by OS/8, including BASIC, FORTRAN IV, and the PAL8 assembly language.

OS/8 ERROR MESSAGES (AA-H610A-TA)

This manual lists in alphabetical order all error messages generated by OS/8 system programs and languages.

CHAPTER 1

OVERVIEW

1.1 INTRODUCTION TO OS/8

This manual describes the OS/8 keyboard commands, the OS/8 Editor, and the OS/8 library of utility programs. These commands and system programs enable you to develop user-written programs in PAL8 assembly language, BASIC, FORTRAN IV, and other languages available under OS/8.

The OS/8 library includes the following system programs.

BATCH
: The BATCH monitor enables you to prepare a job on punched cards, high-speed paper tape, or the system device and then leave it for OS/8 to run.

BITMAP
: The BITMAP program produces a table to show the locations that a binary file occupies in memory.

BOOT
: The BOOT program loads standard hardware bootstraps into memory.

BUILD
: The BUILD program lets you alter the device configuration in your system to insert new devices or add user-written handlers.

CAMP
: The Cassette and Magnetic Tape Positioner program enables you to manipulate cassettes and magnetic tapes.

CREF
: The Cross Reference Program produces a table in assembly listings that enables you to locate references to symbols and literals.

DIRECT
: The DIRECT program produces various types of directories.

DTFRMT, DTCOPY,TDFRMT, TDCOPY
: These programs format and copy DECtapes.

DUMP
: The DUMP program sends listings to the LP08 line printer.

EPIC
: The Edit, Punch, and Compare program reads and punches paper tapes, edits files, and compares files in any format.

FOTP
: The File-Oriented Transfer Program transfers groups of files between file-structured devices.

FUTIL
The File Utility program enables you to examine and modify the contents of mass storage devices.

MCPIP
The Magtape/Cassette Peripheral Interchange Program is a file-transfer program for cassettes and magnetic tapes.

ODT
The Octal Debugging Technique enables you to run and debug a program by typing instructions at the keyboard.

PIP
The Peripheral Interchange Program transfers files between devices and provides file and directory maintenance functions.

PIP10
This is a file-transfer program that reads and writes ASCII DECtape files using a TC08 or TD8E DECtape controller.

RESORC
The RESORC program prints a listing of active device handlers.

RKLFMT
The RKLFMT program formats RK05 disks.

RXCOPY
The RXCOPY program copies diskettes.

SET
The SET program makes it possible for you to modify the operating characteristics of OS/8.

SRCCOM
The Source Compare program compares two source files line by line and prints the differences.

1.2 OS/8 I/O DEVICES

OS/8 provides device independence. You can write programs without concern for specific I/O devices. In running a program, you can select the most effective I/O devices available. Furthermore, if the system configuration is altered, you need not rewrite programs to take advantage of the new configuration.

The OS/8 system controls the copying of data from any medium to any other medium by means of subroutine calls to execute I/O routines. Logical names can be assigned to devices within the system to enable symbolic referencing of devices.

Variable-length I/O buffers can be specified by the user program. Large buffers ensure efficient use of storage devices and a minimum of time spent in data-transfer operations by minimizing disk and tape motion. OS/8 takes full advantage of the RK8E disk pack for fast bulk storage, yet full system services are possible with a single DECtape.

1.3 HARDWARE CONFIGURATIONS

The OS/8 system can operate with the following devices as the system device.

TC01/TC08 DECtape
LINCtape (PDP-12)
TD8E DECtape
DF32/RF08 disk
RK8E disk
RK8 disk
RX01 diskette

The term system device refers to the device on which the OS/8 system resides and which it utilizes for system functions. Thus, DECtape unit 0 is the system device for a DECtape-based system. A nonsystem device is any peripheral not specifically used for system functions, such as LPT:, PTR:, DTA2:, etc.

TD8E DECtape can be used either with 12K words of core memory or with 8K words of core memory and 256 words of Read-Only-Memory (ROM).

If DF32 is the system device, at least 64K (2 platters) must be available.

The minimum OS/8 configuration is a PDP-8 series computer with 8K words of memory, one DECtape used as the system device, and a console terminal. A multiple DECtape system performs appreciably faster than a single DECtape system. The multiple DECtape system reduces DECtape motion since it is possible to copy directly (without intermediate searching) from the system DECtape to another DECtape (or vice versa) when editing or assembling.

A typical medium-sized system might contain a PDP-8/E with at least 8K words of core memory, TD8E DECtape and control, and an RK8E disk pack and control. A disk system offers the additional convenience of easy and fast access to files and large amounts of storage.

Up to 15 of the following devices can be included in a single OS/8 system:

- As many as 8 DECtape units (TC01/TU55, TC08/TU56, or TD8E/TU56)
- TA8E/TU60 cassette units
- TM8E/TU10 magnetic tape units
- High-speed paper tape reader/punch
- Up to four RK8E disks
- Up to four RK8 disks
- Up to four RS08 disks

- Up to four DF32 disks
- Card reader (optical mark or punched cards)
- Line printer
- PDP-12 LINCtape
- PDP-12 scope
- Any other device for which it is possible to write a device handler in one or two pages of core

1.4 SYSTEM SOFTWARE COMPONENTS

The main software components of the OS/8 system include

Keyboard Monitor
The Concise Command Language
Command Decoder
Library of system programs
Device handlers
User Service Routine (USR)

The Keyboard Monitor provides communication between you and the OS/8 executive routines by accepting commands from the console terminal. The commands enable you to create logical names for devices, run system and user programs, and save programs.

The Concise Command Language (CCL) provides an extended set of Monitor Commands.

The Command Decoder allows you to communicate with a system library program by accepting a command string from the keyboard indicating input/output files. Following your keyboard command to run a system library program, the Command Decoder prints an asterisk and then accepts the command line containing device and file specifications.

The library of system programs contains the programs mentioned in Section 1.1 and any of the extension programs you choose.

Device handlers are subroutines designed to transfer data to and from peripheral devices. OS/8 is able to interface with as many as 15 different peripherals at a time. During system generation, device handlers become an integral part of the system; both system and user programs have access to any available device. (The BUILD program allows quick and easy alteration of any available device.)

The User Service Routine (USR) controls the directory operations for the OS/8 system. A program can use the USR by means of standard subroutine calls such as those used to activate device handler subroutines. Some of the functions performed by the USR are loading device handlers, searching file directories, creating and closing output files, calling the Command Decoder, and chaining of programs. The details on the operation and use of the USR are contained in the OS/8 Software Support Manual (DEC-S8-OSSMB-A-D). For normal OS/8 usage, the USR function is unseen by the user and need be of no concern.

When OS/8 is operating, the Command Decoder, Keyboard Monitor, and USR are swapped into core from the system device as required. When their operation has been completed, the previous contents of core are restored.

The memory-resident portion of OS/8 is extremely small (256 words), allowing for a maximum use of memory by user programs.

CHAPTER 2

SYSTEM CONVENTIONS

OS/8 observes the following conventions in the names of devices, files, and units of storage.

2.1 PERMANENT DEVICE NAMES

During configuration, the OS/8 BUILD program assigns permanent names to the devices in a system. You can change these names by reconfiguring the system, but you must keep in mind that some CCL commands and system programs operate on the assumption that certain names are present. The DIRECT command, for example, uses the name TTY: as a default device for listings, and the CREF program assumes LPT: as a default output device. Therefore, it is good practice to keep the following names always present on the system.

```
SYS:
DSK:
TTY:
LPT:
```

Table 2-1 lists all the device names used by OS/8.

Table 2-1
Permanent Device Names

Permanent Name	I/O Device
SYS	System device (disk if the system has a large disk -- RK8 or RF08; otherwise DTA0)
DTAn	DECtape n, where n is an integer in the range 0 to 7, inclusive
LTAn	When using BUILD, LINCtapes may be called LTA rather than DTA. n is an integer in the range 0 to 7 inclusive.
DSK	Default storage device for all files. The assignment of DSK is specified at system generation time. Usually DSK is the disk on a single disk system or DTA0 on a DECtape system.

(continued on next page)

Table 2-1 (Cont.)
Permanent Device Names

Permanent Name	I/O Device
TTY	Terminal keyboard and printer
PTP	Paper tape punch
PTR	Paper tape reader (Before accepting input, the system prints an up-arrow (↑), to which the user replies by typing any key.)
CDR	Card reader
LPT	Line printer (Performs a form feed before it begins printing output from a new program.)
CSAn	Cassette drive n, where n is an integer in the range 0 to 7, inclusive
MTAn	Magnetic tape drive n, where n is an integer in the range 0 to 7 inclusive
DF	DF32 disk
RF	RF08 disk
RKAn	RK01 or RK05 disk unit n, where n is an integer in the range 0 to 3
TV	VR12 scope (PDP-12 only)
BAT	Pseudo device which reads from BATCH input stream (see BATCH section in Chapter 2)
RXAn	Diskette n (floppy), where n is an integer in the range of 0-7 inclusive
RKBn	DECpack n, where n is an integer in the range 0-1
NULL	Device which on input returns an immediate end-of-file and on output ignores characters.
DUMP	Prints contents of device blocks on LPT.

2.2 FILE NAMES AND EXTENSIONS

File names may contain up to six alphanumeric characters, followed optionally by a period and an extension of two alphanumeric characters. The extension usually identifies the file by type. For example, a .PA extension after a file name indicates that the file contains a PAL8 source program.

In most cases, you will want to conform to the standard extensions established for OS/8. If you omit the extension on an output file specification, some system programs append assumed extensions. PAL8, for example, will add .PA to an output file. If you specify a file for input and omit the extension, some system programs will look for a file with an assumed extension. For example, if you specify a program called PUMP as input to PAL8, PAL8 looks for PUMP.PA. If it fails to find it, it looks for the file name and no extension. Table 2-2 lists the file name extensions used by OS/8.

Table 2-2
OS/8 File Name Extensions

.BA	BASIC source file (default extension for a BASIC input file)
.BI	Batch input file
.BK	Backup ASCII file (default extension for a TECO output file)
.BN	Absolute binary file (default extension for ABSLDR, BUILD, and BITMAP input files; also used as default extension for PAL8 binary output file)
.DA	Data file
.DC	Documentation file
.DI	Directory listing
.FT	FORTRAN language source file (default extension for FORT input files)
.HL	Help file (default extension for HELP input files)
.LD	F4 load mode (default assumed by run-time system, F4 loader)
.LS	Assembly listing output file (default extension for PAL8 and SABR)
.MA	Macro source file
.MP	File containing a loading map (used by the Linking Loader)
.PA	PAL8 source file
.RA	RALF assembly language file
.RB	Relocatable binary source file
.RL	Relocatable binary file (default extension for a Linking Loader input file; also used as the default extension for an 8K SABR output file)

(continued on next page)

Table 2-2 (Cont.)
OS/8 File Name Extensions

.SB	8K SABR source file
.SV	Core image file or SAVE file; appended to a file name by the R, RUN, SAVE, and GET Keyboard Monitor commands
.SY	System head
.TE	TECO macro file (default extension for a MUNG input file)
.TM	Temporary file generated by FORTRAN or SABR for system use (default extension for CREF input files and PAL8 output files)
.TX	Text files

2.3 UNITS OF STORAGE

OS/8 uses the terms "word", "page", "record", and "block" to describe units of storage. In directory listings, for example, OS/8 lists file lengths in blocks or records. The terms are defined thus:

1 block=1 record=2 pages=256(decimal) words

A word consists of 12 bits.

CHAPTER 3

OS/8 KEYBOARD COMMANDS

3.1 INTRODUCTION

The OS/8 Monitor and the Concise Command Language program -- CCL.SV -- provide you with more than fifty different keyboard commands.

- The Monitor provides the following commands, which you may abbreviate to the first two letters.

  ASSIGN
  DEASSIGN
  GET
  SAVE
  ODT
  RUN
  R
  START
  DATE

  The Monitor displays a dot to indicate that it is ready to accept a command.

  To execute a command that you have typed, press RETURN or ALTMODE.

  Any error that you make in the use of these commands causes the Monitor to print an error message, display the dot, and wait for you to try again. For a description of these error messages, see Table 3-1.

Table 3-1
Keyboard Monitor Error Messages

Message	Meaning
BAD ARGS	The arguments to the SAVE command are not consistent and violate restrictions listed in 1, 2, 3 under SAVE command.
BAD CORE IMAGE	The file requested was not a core-image file (it could have been an ASCII or binary file).

(continued on next page)

Table 3-1 (Cont.)
Keyboard Monitor Error Messages

Message	Meaning
BAD DATE	The date has not been entered correctly (using slashes), or incorrect arguments were used, or the date was out of range.
ILLEGAL ARG.	The SAVE command was not expressed correctly; illegal syntax used.
MONITOR ERROR 2 AT xxxx (DIRECTORY I/O ERROR)	Attempt made to output to a WRITE-LOCKed device, usually DECtape; or an error has occurred reading/writing a directory.
MONITOR ERROR 5 AT xxxx (I/O ERROR ON SYS)	An error occurred while doing I/O to the system device. This error is normally the result of not WRITE-ENABLing the system device.
MONITOR ERROR 6 AT xxxx (DIRECTORY OVERFLOW)	A directory overflow has occurred (no room for tentative file entry in directory).
name NOT AVAILABLE	The device with the name given is not listed in any system table, or it is not available for use at the moment (check the device in question), or the user tried to obtain input from an output-only device (such as the high-speed paper tape punch).
name NOT FOUND	The file with the name given was not found on the device indicated, or the user tried to input from an output-only device.
NO!!	The user attempted to start (with .ST) a program that cannot be started. The user must not restart any user program or system library program that modified itself while in core (bit 2 of the Job Status Word is set; see the GET command for details).
NO CCL!	The command was not a legal keyboard monitor command. It was, however, a valid CCL command; but the file CCL.SV was not found, or an I/O error occurred while trying to read the file.
SAVE ERROR	An I/O error has occurred while saving the program. The program remains intact in core.

(continued on next page)

Table 3-1 (Cont.)
Keyboard Monitor Error Messages

Message	Meaning
SYSTEM ERR	An error occurred while doing I/O to the system device. The system should be restarted at 7600 or 7605. Do not press CONTinue, as this is sure to cause futher errors.
TOO FEW ARGS	An important argument has been omitted from a command. For example, .RUN DSK would generate this message, as the program to be run has not been entered in the command.
USER ERROR 0 AT xxxx	An input error was detected while loading the program. xxxx refers to the Monitor location where the error was generated.
abcd?	Where abcd is not a legal command; for example, if the user typed: .HELLO the system would echo: HELLO?

- The Concise Command Language program (CCL) provides an extended set of Monitor commands. Some of these commands allow you to call a system program indirectly, perform an operation, and return automatically to the Monitor. This method is simpler to use than the standard calling sequence for a program. For example, the following two-line sequence causes PAL8 to assemble a source program called SCOOP.PA and send a binary and listing file to DSK, the default device.

```
.R PAL8
*SCOOP.BN,SCOOP.LS<SCOOP.PA
```

  You can obtain the same results faster by using the CCL PAL command with the -LS option.

```
.PAL SCOOP.PA-LS
```

  Other CCL commands perform special functions not available through OS/8 utility programs.

  You can write your own CCL commands and add them to the CCL program. For instructions, see the OS/8 Software Support Manual.

You enter a CCL command the same way you enter a Keyboard Monitor Command -- in response to the terminal dot. Normally, you terminate the command line with the RETURN key. Depending on the characteristics of the command you are using, control may return to the Monitor when the operation is completed or may remain within another OS/8 program. To remain under program control when control would normally return to you, terminate the CCL command with an ALTMODE. (Note that this termination procedure is the opposite of the way most OS/8 programs work.)

A special CCL command -- called CCL -- deactivates the entire Concise Command Language Program and all the commands that run under it. To reactivate the program, you must run it with the R command.

CCL provides the following commands, which you may abbreviate to the letters printed as capitals:

BACkspace	DUplicate	PRInt	UC
BAsic	EDIT	PUnch	UNLoad
BOot	DOF	RENname	VERsion
CCL	EXEcute	RES	ZERO
COMPare	HELp	REWind	
COMpile	LIst	SET	
COPy	LOad	SKIP	
CREate	MAKe	SQuish	
CREF	MAP	SUbmit	
DAte	MEMory	TECo	
DEassign	MUNG	TYpe	
DELete	ODT	UA	
DIRect	PAL	UB	

3.2 COMMAND FORMAT

The general format of the command line is

```
command output:file<input:file/option
```

where

command is a legal OS/8 command

output: is the name of the device you specify to receive output

file is the name and extension of an output file

input: is the name of the device you specify for input

file is the name and extension of an input file

/option is a command qualifier

Some commands permit multiple file and device specifications; refer to the descriptions of the commands you want to use for details.

3.3 COMMAND AND FILE OPTIONS

OS/8 command options let you choose the way you want to execute a command. File options let you optimize the storage on an output device.

3.3.1 Command Options

OS/8 recognizes single letters, letter strings, and numbers as symbols for command options. All options are defined in this manual along with the commands they modify.

3.3.1.1 **The Slash Construction -- Single-Letter Options** - Single-letter options follow a slash (/) and may appear anywhere in the command line even in the middle of a filename. For example, this line

```
.COPY RXA1:SECOND.EX<RXA0:FIRST.EX/T
```

and this line

```
.COPY RXA1:SECOND.EX/T<RXA0:FIRST.EX
```

both specify the /T option, which causes the system to assign the current date to the output file.

3.3.1.2 **The Parenthesis Construction -- Multiple-Letter Options** - If you use two or more letter options in a command line, you may group them together as a string within parentheses. This construction may appear anywhere in the line. For example, this command

```
.COPY RXA1:OUTPUT.EX<SYS:INPUT.EX(QT)
```

is the same as typing

```
.COPY RXA1:OUTPUT.EX<SYS:INPUT.EX/Q/T
```

3.3.1.3 **The Equal Sign Construction -- Octal Number Options** - An octal number option, preceded by an equal sign, may occur only once in a command line. If you place it in the middle of the line, you must follow it with a separator character (a comma or left-angle bracket) or another option and a separator character. For example, this line, which includes an octal 3,

```
.DIRECT SYS:=3
```

causes DIRECT to list the directory of SYS: on the terminal in three columns.

3.3.2 File Options

A file option places an upper limit on the number of blocks an output file may use. (One block contains 256 words.) This option allows the system to optimize file storage. For example, this command line

```
.PAL BINARY[19],LIST[200]<SOURCE.PA
```

calls for two output files, BINARY and LIST, which may have a maximum length of 19 blocks and 200 blocks respectively.

3.3.3 Additional Switch Options--the Dash Construction

A special set of command options enable you to send output to the lineprinter or terminal, generate a listing file or a memory map, or call for a particular compiler or assembler. These options are described in Table 3-2.

Table 3-2
Switch Options

Option	Meaning
-L	Send output to LPT.
-LS	Generate a listing file (used with the COMPILE, EXECUTE, and PAL commands). The listing file is written onto SYS: if no output device is specified and is given a .LS extension. The listing filename is the same as the filename that immediately preceded the CCL -LS option.
-MP	Generate a memory map (used with the COMPILE, EXECUTE, and PAL commands).
-NB	Do not create a binary file (used with the COMPILE, EXECUTE, and PAL commands).
-T	Send output to terminal.
-PA	Selects the PAL8 compiler when the files extension does not determine it (used with the COMPILE and EXECUTE commands).
-FT	Selects the FORTRAN IV compiler when the file extension does not determine it (used with the COMPILE and EXECUTE commands).

3.4 COMMANDS THAT REMEMBER FILE SPECIFICATIONS

If you omit the device and file specifications in a CREATE, EDIT, PAL, COMPILE, LOAD, or EXECUTE command, OS/8 assigns it the last argument to appear in any command in the group. For example, these two commands

```
.COMPILE TEST.PA
.EXECUTE
```

instruct the system to compile TEST.PA, then load and execute it. When you enter the COMPILE command, the system stores the argument in a temporary file for later reference by the EXECUTE command. This feature works only with commands that you enter on the same day.

3.5 USING WILDCARDS

Wildcards, which certain OS/8 commands accept, make it possible for you to refer to a group of related files with a single file specification. OS/8 provides two wildcards:

- the asterisk (*), which replaces an entire filename or extension
- the question mark (?), which replaces any single character

3.5.1 Wildcards in Input Filenames

The following commands permit both the asterisk and question mark as wildcards in input specifications.

```
COPY
DELETE
DIRECT
LIST
RENAME
TYPE
```

Here are some examples of the various ways you can abbreviate input specifications with wildcards.

.DEL TEST1.*	deletes all files on DSK with the name TEST1 and any extension
.DIR *.BN	displays a directory of all files on DSK with a .BN extension and any name
.DIR TES??.*	displays a directory of all files with names beginning TES and any extension
.LIST ???.??	lists the contents of all DSK files with names of three characters or less

A filename may not contain embedded asterisks. For example, TE*T.* is an illegal specification and will produce the following error message:

```
ILLEGAL *
```

If you use a wildcard in a command other than the ones listed above, OS/8 prints the error message

```
ILLEGAL * OR ?
```

3.5.2 Wildcards in Output Specifications

You may use the asterisk wildcard in an output file name. The question mark, however, is illegal. If you omit the output file name altogether, the system assumes *.* -- that is, all files with any extension.

For example, this command

```
.COPY RXA1:*.BK<SYS:*.PA
```

copies all files from SYS with a PA extension to RXA1, adding the extension BK.

3.5.3 Warnings and Suggestions

Use wildcards in COPY and DELETE commands with extreme caution to avoid destroying irreplacable files. Always observe the following fail-safe measures.

- Keep a backup copy of the system diskette and all other important files.
- Use the Q option with COPY and DELETE. The system pauses to make sure you have specified the file you intended. If you wish to go through with the operation, type Y in response to the query. If not, type any other character.

For further discussion of wildcards, see the File-Oriented Transfer Program (FOTP).

3.6 INDIRECT COMMANDS

You may occasionally wish to refer to the same group of files in several commands. To avoid typing the same filenames and extensions in each command line, use the indirect -- @ file -- feature.

An indirect file specification has the following format.

```
@device:file.ex
```

where

file.ex is a file containing the file specifications you want to include in the command

To use the @ construction, you must first create a file containing the list of file names you wish to include in the command line. For example, assume you have created a file called FLIST.CM that contains the string

```
FILEB,FILEC/L,FILED
```

To include these names in a COMPILE command, type

```
.COMPILE FILEA,@FLIST,FILEZ
```

The system ignores carriage returns and line feeds -- but not nulls -- within the command line. A null signifies end-of-line.

Command files may not exceed one block in length. If a command line contains more than 512 characters, the system prints the following message:

```
COMMAND LINE OVERFLOW
```

The following commands will not accept indirect files.

```
ASSIGN
DEASSIGN
GET
START
R
RUN
SAVE
ODT
DATE
```

3.7 USING DEFAULTS

A default device, file name, or extension is the name the system assumes if you omit the specification in a command line. You can often reduce the amount of typing necessary to enter a command by taking advantage of the following system defaults.

- DSK is the default device for input and output devices in most commands. This means that you can omit the device whenever you refer to a file on DSK. For example, the following command makes a copy of a file called HUMPTY on DSK and calls it DUMPTY.

```
.COPY DUMPTY<HUMPTY
```

- Any device -- stated or assumed -- in an input specification becomes the default device for any additional input files in the command line. For example, this command lists three files on RXA1.

```
.LIST RXA1:ONE,TWO,THREE
```

- Some commands assume special default devices. DIRECT and TYPE, for example, default to the terminal for output. The following command will display the directory of DSK on the terminal.

```
.DIR
```

The description of the OS/8 command includes information about the defaults that each command accepts.

3.8 GETTING HELP--THE HELP COMMAND

To obtain additional information about the use and format of OS/8 commands, use the HELP command. The format is

```
.HELP command
```

where

command is the name of any OS/8 command

HELP retrieves and displays a file called HELP.HL, which contains the information you request.

To obtain a hard copy of the information from the line printer, use the -L option.

```
.HELP PAL-L
```

3.9 ENTERING A COMMAND LINE--CORRECTING AND PREVENTING ERRORS

The RETURN key enters a command line and causes the system to take the action you have called for. Therefore, before you press RETURN, check the line carefully for errors.

- To correct single-character typing errors, use the DELETE key. This key erases the last character you have typed. Successive DELETEs will erase characters back to the beginning of the line.

- To remove an entire line, type CTRL/U by holding down the CTRL key and striking U. The monitor will echo the command and display a dot to indicate that it is ready to accept another command.

- To verify the contents of the line you are typing, strike the LINE FEED key. The system will display whatever characters it has received so far. Use the LINE FEED key to check a line in which you have made numerous corrections.

If you enter a command line that you have typed incorrectly, one of the following will result:

- The system will fail to recognize or accept the command. In this case, it will display a question mark and a dot and wait for you to try again.

- The Monitor will accept the command and attempt to execute it. If you notice your error at this point, type CTRL/C immediately (simultaneously pressing CTRL and C). Depending on the type of command and the files involved, this may halt execution.

3.10 ASSIGN

The ASSIGN command assigns a logical name -- that is, a name that you create -- to one of the available permanent devices. The format of the command is

```
ASSIGN perm user
```

where

perm is the permanent name of the device

user is the one-to-four-character name you want to assign

Note that a device name does not require a colon when it follows the ASSIGN command.

The following rules apply to the assignment and use of logical names.

- You may assign only one logical name to a device at a time.
- Once you have assigned a name to a device, you may refer to it by either its logical or permanent name. ASSIGN makes the two names equivalent.

For example, this command

```
.ASSIGN RXA1 DEV2
```

assigns the logical name DEV2 to RXA1. You may now refer to the device by either name in any command line.

3.10.1 Canceling a Logical Name

To cancel a logical name, type the ASSIGN command with the permanent device name only. For example, to remove DEV2 as a logical name for RXA1, enter

```
.ASSIGN RXA1
```

3.10.2 Checking for Duplicate Names

To determine if a logical name is unique in the OS/8 system, enter the name by itself in an ASSIGN command line. For example, to see if DEV2 already exists, type

```
.ASSIGN DEV2
```

If the name does not appear in any of the system tables, ASSIGN displays the message

```
DEV2 NOT AVAILABLE
```

All 1- and 2-character names are unique in OS/8. You need test only 3- and 4-character names.

ASSIGN is a Monitor command.

3.11 BACKSPACE

The BACKSPACE command runs the OS/8 CAMP program and spaces a magnetic tape or cassette backward a specified number of files or records. BACKSPACE is equivalent to the CAMP BACKSPACE command. When CAMP has completed a BACKSPACE operation, it returns control to the Monitor.

The format is

```
BACKSPACE dev:nnnn X
```

where

dev: is the permanent name of a cassette or magnetic tape drive

nnnn is an unsigned decimal number representing the number of records or files you wish to backspace. If you omit the number, BACKSPACE assumes nnnn=1.

X is an R or F to indicate records or files. If you do not specify records or files, BACKSPACE assumes F.

For example, this command

```
.BACKSPACE CSA0:2 F
```

positions the cassette mounted on CAS0 backward two files.

For complete information on the BACKSPACE command, see the chapter on the CAMP program.

BACKSPACE is a CCL command and runs the CAMP program.

3.12 BASIC

The BASIC command invokes the BASIC Editor. The format is

```
BASIC
```

As soon as it is ready to accept your first instruction, BASIC prints the query

```
NEW OR OLD --
```

to determine if you want to create a new file or work on an old one.

For example, this command

```
.BASIC
NEW OR OLD -- NEW STUFF.BA
```

tells the BASIC Editor to accept a new program called STUFF.BA

For complete information on OS/8 BASIC, see the OS/8 Language Reference Manual.

3.13 BOOT

The BOOT command makes it possible for you to bootstrap onto another device or onto another PDP/8 system. The format is

```
BOOT/dv
```

where

dv is a mnemonic listed in the BOOT chapter in this manual

If you type BOOT with no argument, BOOT prints a slash to indicate that you must enter a mnemonic.

For example, this command

```
.BOOT/RF
```

bootstraps onto the RF08 disk.

If you wish to halt before doing the bootstrap, type the command, a mnemonic, and a period. For example:

```
.BOOT/CA.
```

The period causes the computer to halt, giving you time to mount a new device. To continue the operation, press the CONTINUE switch on the console. This form of the command is useful when only one disk or DECtape drive exists on the system.

For complete information, see the BOOT chapter in this manual.

3.14 CCL

The CCL command disables the Concise Command Language program on the system device. The format is

```
CCL
```

The command accepts no arguments.

The CCL command totally deactivates the CCL feature of OS/8 so that the system will not accept any CCL command. If you wish to use CCL again, you must reactivate it with the R command. To do this, type

```
.R CCL
```

3.15 COMPARE

The COMPARE command makes a line-by-line comparison of two input source files and sends the results to an output device. In most COMPARE operations, the two source files are different versions of the same program. COMPARE prints the editing changes, making it a useful tool for debugging.

The format is

```
COMPARE output:file<input:filel,input:file2
```

where

output:file	is the file containing the results of the comparison and the device you want to send it to
input:filel	is the first input source file for comparison
input:file2	is the second source file

COMPARE makes the following assumptions:

- If you omit an input or output device, COMPARE assumes DSK
- If you omit the output specification altogether, COMPARE assumes TTY. (In most cases you will want to see the results on the terminal.)

For example, this command

```
.COMPARE RXA1:APPLE.FT,RXA1:ORANGE.FT
```

compares two FORTRAN source files on RXA1 -- APPLE and ORANGE -- and sends the results to the terminal.

3.15.1 COMPARE Output

COMPARE produces the following output sequence:

1. the current version number of the utility program SRCCOM
2. the header line of both input files (the header is the first line of the file and usually contains the file name and creation date)
3. a difference group (see below)
4. additional difference groups, if any, until it reaches the end of the shorter file

COMPARE reads two input files one line at a time until it encounters three consecutive matching lines. Then it outputs all lines from both files up to and including the first matching line. This output is called a difference group. For a complete description of difference groups, see the SRCCOM chapter in this manual.

For example, consider two files on DSK--NITTY and GRITTY.

```
NITTY          GRITTY

B              X
C              C
D              D
E              E
F              G
G              H
H              J
I
J
```

To compare these two files and have the results displayed on the terminal, type

```
.COMPARE NITTY,GRITTY
```

COMPARE prints the current version number of SRCCOM, the utility program that does the comparison,

```
SRCCOM V4A
```

the header lines

```
1)NITTY
2)GRITTY
```

and the results of the comparison in two difference groups

```
1)002   B
1)      C
****
2)002   X
2)      C
*********
1)002   F
1)      G
1)      H
1)      I
1)      J
****
2)002   G
2)      H
2)      J
```

If COMPARE discovers two identical files, it prints

```
NO DIFFERENCES
```

in the output file.

3.15.2 COMPARE Options

COMPARE provides the following options:

- =k In normal operation, COMPARE outputs lines until it encounters three consecutive matching lines. To change the number of lines that interrupt processing, use the =k option, where k is the number of lines (in octal) that you want to specify.

- /C COMPARE ignores comment fields during the comparison of assembly language source files.
- /B COMPARE treats a blank line as valid input containing blanks instead of a carriage return.
- /S COMPARE ignores all tabs and spaces during the comparison.
- /T COMPARE converts all tabs from the input file to spaces on the output device.
- /X COMPARE ignores all comment fields during the comparison and does not send comments to the output device.

COMPARE is a CCL command and runs SRCCOM.

3.16 COMPILE

The COMPILE command

- assembles a PAL8 source program and outputs an absolute binary file and a listing file or a CREF file
- compiles and assembles a FORTRAN IV source program and outputs a relocatable binary file and a listing file
- compiles, loads, and executes a BASIC source program

The format for a PAL8 program is

```
COM output:prog.BN,output:list.LS<input:file1.PA,...file9.PA
```

where

output:prog.BN	is a PAL8 binary file
output:list.LS	is a PAL8 listing file
input:file1,...file 9	is a single source program, which you may enter in up to nine separate files

For example, this command line

```
.COMPILE RXA1:ACE.BN,RXA1:ACE.LS<RXA1:ACE.PA
```

assembles a PAL8 source program on RXA1 called ACE.PA and produces a binary file, ACE.BN, and a listing file, ACE.LS.

The format for a FORTRAN source program is

```
COMPILE output:prog.RL,output:list.LS<input:file1.FT,...file9.FT
```

where

output:prog.RL	is an assembled FORTRAN program
output:list.LS	is a listing file
input:file1,...file 9	is a FORTRAN source program stored in up to nine files

For example, this command

```
.COMPILE RXA2:DEUCE.RL,RXA2:DEUCE.LS<RXA1:DEUCE.FT
```

assembles DEUCE.FT and outputs a binary and a listing file -- DEUCE.RL and DEUCE.LS.

3.16.1 COMPILE Input

Enter your PAL8, FORTRAN, or BASIC program as an input file or files. You may include up to nine input files in a single COMPILE command line. COMPILE assumes that they all contain sections of the same program.

The extension on the file name identifies the type of program the file contains and tells the command which assembler or compiler to summon.

- .PA identifies a PAL8 source program
- .FT identifies a FORTRAN source program
- .BA identifies a BASIC source program

If you use nonstandard extensions, you must specify the assembler or compiler your program requires with processor switch options -PA, -FT, or -BA.

COMPILE makes the following assumptions about input device names:

- If you omit the device in the first input specification, COMPILE assumes DSK.
- If you omit the device in subsequent input entries, COMPILE assumes the last device you name.

For example, this command line

```
.COMPILE PART1.PA,PART2.PA,PART3.PA
```

assembles a 3-part PAL8 program on DSK.

3.16.2 COMPILE Output

COMPILE outputs binary and listing files, assigning the names you specify.

3.16.2.1 Output File 1 -- the Binary Code

- If your input file is a PAL8 source program, COMPILE assigns the first output file name to the binary code generated by the assembler, adding a .BN extension if you omit it.
- If your input file is a FORTRAN program, COMPILE assigns the first output file name to the relocatable binary code produced by the FORTRAN assembler, adding .RL if you omit the extension.

If you omit the device, COMPILE assumes DSK. If you omit the file name, COMPILE assumes the name of the first input file.

For example, this command line

```
.COMPILE RXA1:SUNDAY<RXA2:MONDAY.FT
```

compiles and assembles the source program MONDAY.FT and sends it to RXA1 as a binary file called SUNDAY.RL.

In certain COMPILE operations, you may wish to suppress the output binary file and generate only a listing file. To suppress a binary file, use the -NB (no binary) switch. For example,

```
.COMPILE SAMPLE.PA-LS-NB
```

assembles SAMPLE.PA and produces only a listing file.

3.16.2.2 Output File 2 -- the Listing File - You can request a listing file of a PAL8 or FORTRAN program in two ways:

- Enter a second output file in the COMPILE line. COMPILE generates the listing and assigns it the name you specify, adding .LS if you omit the extension.
- Type the -LS switch option after an input file name. COMPILE generates the listing, gives it the name immediately preceding the switch, and adds the .LS extension.

For example, this command

```
.COMPILE PARTY,PARTY<PARTY.PA
```

and this command

```
.COMPILE PARTY.PA-LS
```

both generate a listing file PARTY.LS.

If you use the PAL8 /C option, the second output file is passed to the CREF program, which produces a cross-reference listing.

3.16.2.3 COMPILE Options and Errors - For a complete description of COMPILE options and errors, see the sections on PAL8, FORTRAN IV, and BASIC in the OS/8 Language Reference Manual.

COMPILE is a CCL command.

3.17 COPY

The COPY command transfers files from one device to another. The format is

```
COPY output:file<input:file1,...input:file5
```

COPY sends files to the output device in exactly the same format and order in which they appear in the command. Since the operation makes no changes at all in the files, you may transfer any kind of file -- memory image, binary, source -- with the COPY command.

3.17.1 COPY Input

You enter the file or files you want to transfer as input in the command line.

A complete COPY input specification includes a device, a file name, and an extension. You may enter up to five input files in a command line. COPY uses the following input defaults:

- If you omit the device name in the first input specification, COPY assumes DSK.
- If you omit the device name in succeeding input specifications, COPY assumes the last device entered.

You may use the wildcards * and ? to transfer an entire group of related files with a single command. The specification *.* tells COPY to transfer all the files on a device.

3.17.2 COPY Output

A complete output specification includes a device, a file name, and an extension. You may enter only one output specification in a command line. If you want your transferred file to have a different name from the original, you must enter that name as the output file.

COPY uses the following output defaults:

- If you omit the device, COPY assumes DSK.
- If you omit the output file name, COPY assumes *.* -- that is, it assumes that the output file has the same name as the input file.

You may use only the wildcard *; the question mark (?) is illegal. Keep in mind that as output the specification *.* tells COPY to give the output file the same name as the input file.

3.17.3 COPY Terminal Display

During execution, COPY prints on the terminal the names of the files it has transferred. For example, the following command transfers three files from DSK to RXA1.

```
.COPY RXA1:FLOWER.PA<ROSE.PA,DAISY.PA,ZINNIA.PA
```

COPY displays

```
FILES COPIED
ROSE.PA
DAISY.PA
ZINNIA.PA
```

3.17.4 Predeletion and Postdeletion

Before COPY transfers a file to an output device, it checks the file name against the output file directory. If it finds a file on the device with the same name and extension as the file it is going to transfer, COPY automatically deletes it before it does the transfer. This operation -- called predeletion -- makes space for the new file on the output device, which may not otherwise be able to hold another file. However, it may also cause you to lose a valuable file if for some reason the input fails. To help protect against such loss, COPY provides a second method of transfer called postdeletion. In this mode, COPY deletes any file with the same name as the input file only after it has completed the transfer. To specify postdeletion, use the /N option.

3.17.5 COPY Options

- /C COPY transfers all input files with the current date onto the output device.

- /F A file will not fit on the output device. COPY prints

  ```
  MOUNT NEW DEVICE
  ```

  on the terminal. You remove the current device and mount a new one on the same unit. To continue the transfer, type any character. If possible, ZERO the directory of the new device.

- /T As part of the transfer operation, COPY changes the creation date on the output file to the current date. Without /T, COPY transfers the original date.

- /U COPY transfers input files in the exact order that they appear in the command line -- not the order in which they occur on the device.

- /V COPY transfers all files on a device except the ones you specify in the command line.

- /W COPY prints its current version number.

- /N COPY uses postdeletion.

- /O COPY transfers all files on a device except those with the current date.

- /Q COPY pauses before a transfer to make sure you want to go through with the operation. If you do, type Y; if not, type any other character.

COPY is a CCL command and runs FOTP. For complete information on file transfer, see the FOTP chapter in this manual.

3.18 CREATE

The CREATE command summons the OS/8 Editor to let you open and write a new file. CREATE accepts no input specifications and only one file name and device for output. The format is

```
CREATE output:file
```

where

output:file	is the name of the file you want to create and the device you want to store it on.

For example, this line

```
.CREATE BIRDY.PA
```

opens a file called BIRDY.PA on the default device, DSK.

After you press the RETURN key to execute the command, the OS/8 Editor displays a number sign (#) to indicate that it is ready to receive your instructions and text. To CREATE a file with the OS/8 Editor, see Chapter 4 in this manual.

Each time you enter a CREATE command, the Monitor holds the argument (the device and file name) in a temporary location. If you type an EDIT command later without an argument, EDIT reads the file name in this location. This convenient feature works only with files that you CREATE and EDIT on the same day.

CREATE is a CCL command and runs EDIT.SV.

3.19 CREF

The CREF command assembles a PAL8 program and produces a cross-reference listing, usually on the line printer.

The format is

```
CREF outdev:file.LS<indev:file.PA
```

CREF makes the following assumptions about output and input specifications.

- If you omit the extension on the input file, CREF assumes PA.
- If you omit the extension on the output file, CREF adds LS.
- If you omit the output specifications altogether, CREF sends the file to the line printer.

3.19.1 CREF Options

- /P CREF disables pass-one listing output until it encounters a $ in the source program. Thus, if you use the /P option, CREF prints a dollar sign and the symbol table.
- /U CREF disables pass-one listing output and the symbol table.
- /X CREF does not process literals. This option provides space for CREF to operate on large programs with many symbols and literals.
- /E CREF does not eliminate the intermediate CREFLS.TM file that is output from assembly and used as input to CREF.
- /M CREF cross-references mammoth files in two major passes. Pass one processes the symbols from A through LGnnnn; pass two processes the symbols from LHnnnn through Z and the literals.

CREF is a CCL command and runs PAL8.SV and CREF.SV.

3.20 DATE

The DATE command lets you set and inspect the current system date. You should always set the date as soon as you bootstrap the system.

To set the date, type

```
DATE dd-mmm-yy
```

where

dd	is a two-digit number representing the day of the month
mmm	are the first three letters of the month
yy	are the last two digits of the year

For example:

```
.DATE 23-MAY-77
```

To inspect the current system date, type

```
.DATE
```

For example:

```
.DA 1-JUN-77
.DA
.WEDNESDAY JUNE 1, 1977
```

The system uses the current date in directories, newly created files, and files transferred from one device to another. If you enter the date after booting, the only valid directory entry dates are those for the current year and seven years preceding it. The system will print any earlier date incorrectly.

If you enter the date incorrectly, the Monitor prints an error message.

```
BAD DATE
```

The DATE command runs program CCL.SV.

3.21 DEASSIGN

DEASSIGN invalidates all logical (user-defined) names that you have given to permanent devices. (See the ASSIGN command.)

The format is

```
DEASSIGN
```

For example, the following pair of commands assign the logical name DEV1 to SYS and then cancel it.

```
.ASSIGN SYS DEV1
.DEASSIGN
```

The Monitor performs the DEASSIGN function.

3.22 DELETE

The DELETE command removes files from the directory of the device you specify. The format is

```
DELETE input:file1,...file5
```

Enter the files -- up to five -- that you want to delete as input files in the command line. You may specify only one device in a command line. DELETE makes the following assumptions about input specifications.

- If you omit the device name in the first input specification, DELETE assumes DSK.
- If you omit the device name in succeeding specifications, DELETE assumes the last device you entered. For example, this command

  ```
  .DELETE RXA1:EAST,WEST,NORTH,SOUTH
  ```

  deletes four files from RXA1.

Note that DELETE does not actually remove the file from the device. It simply erases its name from the directory, making the space it occupies available for a new file. This means that in some cases you may be able to retrieve a file you have mistakenly "deleted." For details on retrieving lost files, see the PIP /I option in this manual. Also see the SUPERTECO section in the OS/8 TECO Reference Manual.

You may use the wildcard asterisk (*) to specify file names and extensions and the question mark (?) to indicate single characters. Wildcards enable you to remove an entire group of related files with a single DELETE command. For example, this command

```
.DELETE *.PA
```

removes all files with a PA extension from the system device. Use wildcards with extreme caution to avoid deleting irreplacable files.

3.22.1 The Conditional DELETE

In most DELETE operations you simply list the files you want to remove from a device as input files. In some cases, however, you may wish to remove a file only on the condition that some other related file exists -- a source file, for example, if the device also contains the program in binary code.

To call for this kind of DELETE, use the following format:

```
DELETE output:file<input:file
```

where

output:file	is the file you wish to delete if file f is present
input:file	is file f

This command deletes, for example, any DSK file with a .PA extension, only on the condition that DSK contains a file with the same name and a .BN extension.

```
.DELETE *.PA<*.BN
FILES DELETED:
TEST1.PA
TEST2.PA
```

Only the wildcard (*) is legal in an out specification. You may not use the question mark.

3.22.2 DELETE Terminal Display

During execution, DELETE prints on the terminal the names of the files it has removed. For example:

```
.DELETE RXA2:SNOW.BA,RAIN.BA
FILES DELETED:
SNOW.BA
RAIN.BA
```

3.22.3 DELETE Options

You may qualify a DELETE command with the following slash options:

- /C DELETE removes only those files that have the current date.
- /O DELETE removes all files except those with the current date.
- /V DELETE removes all the files from a device except the ones you specify in the command line.
- /Q DELETE prints a question mark before execution to make sure that you specified the right files for removal. If your answer is yes, type Y; if no, type any other character.
- /N DELETE displays a log of all files it has found for deletion but does not remove the file names from the directory.

DELETE causes the execution of program CCL.SV and FOTP.SV with the /D option. For further information on deleting files, see the FOTP chapter in this manual.

3.23 DIRECT

The DIRECT command produces listings of OS/8 device directories. The format is

```
DIRECT output:file<input:file1,...input:file5
```

DIRECT prints a directory of all the files on all the devices (up to 5) that you specify in the command line. DIRECT makes the following assumptions about input and output specifications:

- If you omit the input device, DIRECT assumes DSK.
- If you omit input file names, DIRECT assumes *.* -- that is, all files with any extension.
- If you omit the output device, DIRECT prints the directory on the terminal.
- DIRECT automatically adds a DI extension on an output file.

Always use wildcards in your input specifications when requesting a list of related files on a device. For example, this command calls for a list of all files on DSK that have a .PA or .BN extension.

```
.DIR *.PA,*.BN
```

You may use wildcards to represent an output file name or extension.

3.23.1 DIRECT Output

The standard DIRECT listing has the following format:

```
filespec          nnn          dd-mmm-yy
```

(file name and extension) (number of blocks used in decimal) (file-creation date)

If you do not enter the current date with the DATE command when you boot the system, your directory listings will not include a file-creation date.

For examples of DIRECT output, see the DIRECT chapter.

3.23.2 DIRECT Options

DIRECT options enable you to produce the following special directories.

- =n DIRECT lists a directory in n columns, where is a number from 1 to 7.
- /C DIRECT lists only files with the current system date.
- /E DIRECT includes empty files in the listing.
- /F DIRECT lists only file names, omitting lengths and dates.
- /M DIRECT lists empty files only.

- /O DIRECT lists only files with a date other than the current date.
- /R DIRECT lists the file name you specify and all files that follow it in the device directory.
- /U DIRECT lists files in the same order that you enter them in the command line. That is, it treats each input file specification as a separate directory request.
- /V DIRECT lists all files on a device except the ones you specify in the command line.
- /W DIRECT displays its current version number.

3.24 DUPLICATE

The DUPLICATE command copies the entire contents of one diskette to another diskette. The format is

```
DUPLICATE output diskette:<input diskette:
```

You may use DUPLICATE to transfer the contents of diskettes only. For example, the following command transfers the contents of RXA0 to RXA1, the device specified for output.

```
.DUPLICATE RXA1:<RXA0:
```

DUPLICATE is a CCL command and runs the RXCOPY program. For a description of the options you can use with DUPLICATE, see the RXCOPY chapter in this manual.

3.25 EDIT

The EDIT command summons the OS/8 Editor to let you retrieve and work on a source program that you have stored as a file. The format is

```
EDIT output:file<input:file1,...input:file9
```

where

input:file1...9	is a program (stored in one to nine files) you want to work on and the device on which it is located
output:file	is the name of the modified file and the device you want to send it to

The Editor signals with a number sign (#) as soon as it is ready to accept your first instruction. For a discussion of these instructions, see Chapter 4.

3.25.1 Recalling Arguments

The EDIT command can recall arguments from a previous EDIT or CREATE command entered on the same day. If you enter both an input and an output file, EDIT remembers only the output specification.

3.26 EOF

The EOF (End of File) command runs the CAMP program and writes a single mark (file gap) on the specified magnetic tape or cassette. The EOF command has the format

```
EOF device
```

where

device	is either MTAn or CSAn, signifying the device on which the file gap mark is to be written

For example, this command

```
.EOF MTA3
```

writes an end-of-file mark on the magnetic tape mounted on MTA3.

For a complete description of CAMP commands, see the CAMP chapter in this manual.

3.27 EXECUTE

The EXECUTE command

- assembles or compiles, links, loads, and executes a source program
- links, loads, and runs an assembled or compiled program
- runs a linked and loaded program

The format is

```
EXECUTE output:file.bn,file.ls<input:filel,...file9
```

The EXECUTE command is the same as the COMPILE command with the /G option. The input and output specifications depend on the compiler or assembler you invoke. For complete information, see the COMPILE command and the various language chapters in the OS/8 Language Reference Manual.

3.28 GET

The GET command loads a memory-image file -- that is, an SV file you have created with the SAVE command -- back into memory. The format is

```
GET input:file.SV
```

If you omit the extension, GET looks for a file with the name you specify and an .SV extension. You must specify the device; GET does not assume DSK.

For example, to load into memory a file called JOBCNT.SV on RXA0, type

```
.GET RXA0:JOBCNT
```

During execution, GET loads the file and its Core Control Block into memory, then transfers the CCB to a special area on the system device for reference and maintenance. GET also places the Job Status Word into location 7746 of field 0 to indicate what parts of memory the file uses and how. It loads the block number of the first block of the file into location 7747.

To run a program that you have loaded into memory with GET, use the START or EXECUTE command.

The Monitor performs the GET operation.

3.29 HELP

The HELP command sends information on OS/8 system programs to an output device, usually the terminal. The format is

```
HELP output:file<OS/8
```

where

OS/8	is the name of an OS/8 keyboard command or system program

The default output device for HELP is TTY, the terminal.

To see a complete listing of keyboard commands, type

```
.HELP
```

For information on a specific command, enter the command name as the argument in a HELP line:

```
.HELP PAL
```

To obtain a listing of all legal arguments for HELP, type

```
.HELP HELP
```

HELP runs the program HELP.SV, which uses a reference file HELP.HL. This file, which must be located on the system device, contains a list of all the HELP subfiles available along with the HELP text itself.

3.30 LIST

The LIST command sends to the line printer the contents of the files (up to five) that you specify as input in the command line. The format is

```
LIST input:file1,...input:file5
```

LIST requires no output specification, assuming LPT. If you omit the input device, LIST looks for the file on DSK.

For example, this command prints the source program PROG.BA, located on DSK, on the line printer.

```
.LIST PROG.BA
```

LIST outputs the contents of each input file in the same order that you enter the files in the command line.

3.30.1 LIST Options

- /C LIST prints all files with the current date.
- /O LIST prints all files except those with the current date.
- /V LIST prints all files except the ones you specify in the command line.
- /Q LIST displays each file name and a question mark. If you want to list that file, type Y; if not, type any other character.

LIST runs CCL.SV and FOTP.SV.

3.31 LOAD

The LOAD command lets you load a PAL8 absolute binary file or a FORTRAN relocatable binary file into memory.

The extension on the input file determines which loader the command summons.

- BN identifies a PAL8 program in absolute binary form and causes LOAD to summon the ABSLDR.
- RL identifies a FORTRAN program in relocatable code and causes LOAD to summon the FORTRAN loader.

To LOAD a PAL8 program, use the following format:

```
LOAD input:filel.BN,...input:file9.BN
```

where

input:filel.BN,...9.BN	is a PAL8 absolute binary file contained in 1 to 9 files

If your input file is a PAL8 program, LOAD ignores output specifications.

For example, this command

```
.LOAD RXA1:TIC.BN,TAC.BN,TOE.BN
```

places a 3-part program in memory.

To LOAD a FORTRAN program, use the following format:

```
LOAD output:image.LD,output:map.LS<input:filel.RL,...input:file9.RL
```

where

output:image.LD	is an optional loader image file
output:map.LS	is an optional loader symbol map
input:file.RL...9.RL	is a FORTRAN program in relocatable binary form stored in 1 to 9 files

Once you have placed the program in memory, you can

- run it with the START command
- create an .SV file with the SAVE command
- debug the program with ODT

For a list of the options that LOAD accepts, see the chapters on the PAL8 ABSLDR and the FORTRAN loader.

3.32 MAKE

The MAKE command runs TECO and opens the file you specify for output. The format is

```
MAKE output:file
```

If you omit the device name and the file extension, MAKE assumes DSK and .PA.

If the file you specify already exists, MAKE prints the message

```
%SUPERSEDING
```

For example, this command

```
.MAKE DTA1:TEXT.TX
```

is the same as typing

```
.R TECO
*EWDTA1:TEXT.TX$$
```

For further information, see the OS/8 TECO Reference Manual.

3.33 MAP

The MAP command produces a map -- usually on the line printer -- of all the memory locations used by the absolute binary files you specify in the command line.

The format is

```
MAP output:file<input:file1,...input:file9
```

MAP accepts a minimum of nine files as input in a single command line. To specify more than nine files, press the ESCAPE key after the ninth entry. This causes the Command Decoder to print an asterisk (*), indicating that you may continue to specify input files, terminating each with the RETURN key. To execute this command, type another ESCAPE.

If you omit the extension from an input file name, MAP assumes .BN. If you omit the output device, MAP sends the map to the line printer. To display a map on the terminal, use the /T option.

3.33.1 MAP Output

MAP depicts MEMORY as a series of 100-digit lines (in octal), grouped in pairs. Each pair of lines represents one memory page; each digit in a line represents one memory location. Depending on the contents of a location, MAP prints the digit 0, 1, 2, or 3.

- 0 means the program did not load into this location.
- 1 means the location was loaded into once.
- 2 means the location was loaded into twice.
- 3 means the location was loaded into three times.

If you specify the terminal as the output device, MAP prints a set of octal numbers across the top of the map. Each number -- ranging from 00 to 77 -- is the vertical co-ordinate for the column of digits below it. To determine the memory address of any entry in the map, add the line number at the left to the octal number directly above.

For examples of MAP output, see the BITMAP chapter in this manual.

3.33.2 MAP Options

You can modify MAP output with the following options:

- /n MAP confines the construction of maps to the field you specify as n.
- /R MAP resets the map just constructed in memory to look as though nothing has been read in. If you specify the wrong file in a MAP command, use the /R option at the end of the line.

- /S MAP reads every absolute binary program in an input file. In normal operation, MAP accepts only the first file.

- /T MAP changes the format of the output map -- that is, it sends a map to the line printer in terminal format and vice versa.

The MAP command runs CCL.SV and BITMAP.SV.

3.34 MEMORY

The MEMORY command finds the highest field available in hardware or limits the fields available in software.

The format is

```
MEMORY
```

or

```
MEMORY n
```

where

n is an octal number from 0 to 7 representing the number of fields (each containing 4K words of memory) in software.

For example, this command line

```
.MEMORY 3
```

limits the amount of memory available in the system to 16K words.

The following list shows all the values of n and their meaning:

```
0    all available memory fields
1    8K words of memory
2    12K words
3    16K words
4    20K words
5    24K words
6    28K words
7    32K words
```

To find the amount of memory currently being used by OS/8, type the command with no argument. The following output indicates that a MEMORY 4 command, entered previously, has restricted a 32K system to only 20K words of available memory.

```
.MEMORY
20K/32K MEMORY
```

If the system is using all available memory, the Monitor prints the total amount. For example:

```
.MEMORY
32K MEMORY
```

The MEMORY command causes the execution of CCL.SV.

3.35 MUNG

The MUNG command lets you call a predefined TECO macro to operate on a source file. The format is

```
MUNG device:file,text
```

where

device:file	is a file containing a TECO macro. If you omit the extension, MUNG assumes .TE. If you type a period after the file name, MUNG assigns no extension.
text	is an argument to the macro. If the macro requires no argument, omit the comma in the command line.

MUNG reads the first page of the specified file -- the macro -- into Q-register Y. Then it enters the text into the TECO text buffer. With the pointer at the end of the buffer, TECO executes the macro in Q-register Y. If the text argument is too long, MUNG prints the error message

```
COMMAND TOO LONG
```

For complete information on TECO, see the TECO Reference Manual.

3.36 ODT

The ODT command enables you to debug the program currently in memory, control its execution, and make alterations by typing ODT instructions at the terminal.

The format is

```
ODT
```

Once you have entered the command with the RETURN key, you may examine and modify any memory location of the program currently in memory or use the breakpoint feature to control program execution.

When using ODT to debug a program, you must call I/O devices by their permanent names. As long as ODT is in control of the system, all user-defined names are invalid.

For a complete discussion of ODT, see the ODT chapter.

3.37 PAL

The PAL command assembles a PAL8 source file, producing an absolute binary file with a .BN extension.

The format is

```
PAL output:binary.BN,output:listing.LS,CREF.LS<input:source.PA
```

where

output:binary.BN	is an absolute binary file and output device
output:listing.LS	is an optional listing file and output device
output:CREF.LS	is an optional file used by CREF
input:source	is a PAL8 program and an input device

If you omit the extension in the input specification, PAL assumes PA. If you omit the extension from the output files, PAL assumes BN and LS.

The following example causes PAL to assemble a file called BOOMER.PA and produce a listing file.

```
.PAL BOOMER,BOOMER<BOOMER.PA
```

PAL can recall arguments from any previous COMPILE, LOAD, or EXECUTE command that you enter on the same day.

The /C option causes PAL8 to chain to the CREF program, which produces a cross-reference file and assigns it the name of the second output file.

For a complete list of PAL options, see the PAL8 chapter in the OS/8 Language Reference Manual.

PAL runs CCL.SV and PAL.SV.

3.38 PRINT

The PRINT command runs a program called LPTSPL if you have such a program on your OS/8 system. LPTSPL can be a user-written program or a program obtained from DECUS.

3.39 PUNCH

The PUNCH command runs PIP and punches the file specified on the paper tape. The format is

```
PUNCH output:file<input:file
```

If you omit the output specification, PUNCH sends the file to PTP.

3.40 R

The R command loads and starts a memory-image file from the system device. The format is

```
R file.SV
```

R writes the block number of the first block in the file in location 7747 in field 0.

Since the R command loads files from the system device only, you may not specify an input device other than DSK in the command line. If you omit the file extension, R assumes SV.

For example, this command

```
.R TEST
```

looks for a program called TEST.SV on the system device and loads and executes it.

The R command differs from the RUN command in that it does not send the Core Control Block to the system device. To save a program that does not have its Core Control Block in the usual place on SYS, you must include all the optional arguments in the SAVE command.

Always use R to call a system program, since these do not have to be resaved. If you want to run a program that you eventually plan to update (using ODT, for example) and then save, use the RUN and GET command rather than R.

3.41 RENAME

The RENAME command lets you change the name of a file. The format is

```
RENAME device:newname<device:oldname
```

where

device:oldname	is the file name you want to change and the device on which it is located
device:newname	is the new name and the same device

You must specify the same device for input and output in the command line.

RENAME changes the input file name and prints the message FILES RENAMED on the screen, followed by the old file name. Thus, if you type

```
.RENAME RXA1:FILE.PA<RXA1:RECORD.PA
```

the file RECORD.PA on RXA1 becomes FILE.PA. The creation date and the contents of the input file remain the same.

You may use wildcards with the RENAME command.

3.41.1 RENAME Options

RENAME provides the following options:

- /C RENAME changes the name of the input file only if it has the current date.
- /O RENAME changes the name of the input file only if it does not have the current date.
- /V RENAME changes the name of all files on a device except the ones specified as input in the command line.
- /T RENAME changes the name of the input file and gives it the current date.

RENAME runs CCL.SV and FOTP.SV with the /R extension.

3.42 RES

The RES command runs the RESORC program and lists the device handlers present on an OS/8 system. The format is

```
RES output:file<input:file
```

For a description of the input and output specifications for RES and a list of RES options, see the chapter on the RESORC program in this manual.

3.43 REWIND

The REWIND command runs the CAMP program and issues a rewind command to a specified OS/8 device controller. This command operates in the same way as the CAMP REWIND command. For a complete description of this command, see the CAMP chapter in this manual.

3.44 RUN

The RUN command loads a memory image (SV) file into memory, transfers its Core Control Block to the system device, and begins execution at the starting address of the program. It places the block number of the first block in the file into location 7747 of field 0.

The format is

```
RUN input:file
```

If you enter a file name without an extension, RUN assumes SV. You must specify a device; RUN does not assume DSK.

For example, the following RUN command GETs and STARTs PROG.SV on RXA1.

```
.RUN RXA2:PROG.SV
```

3.45 SAVE

The SAVE command makes an executable binary file of the program currently in memory, assigns it a name, and stores it on a device. If you do not specify the locations in memory that you want to save, the SAVE command automatically looks for the information on the current Core Control Block.

The format is

```
SAVE device:file fnnnn-fmmmm,fpppp;fssss=cccc
```

where

fnnnn	is a 5-digit octal number representing the field (f) and starting address of a continuous portion of memory that you want to save
fmmmm	is the final address (in the same field) of that part of memory you intend to save
fpppp	is a 5-digit octal number representing the address of one location in memory. A single address causes SAVE to save the entire page on which the location occurs
;fssss	is a 5-digit octal number representing the starting address of the program you want to save
=cccc	is a 4-digit octal number representing the contents of the Job Status Word

If you omit the extension on the file name, SAVE appends SV. If you omit the other arguments, SAVE finds the locations it requires in the current Core Control Block.

The SAVE command places the following restrictions on arguments in the command line.

- You must specify the output device. SAVE does not default to DSK.
- The beginning and ending addresses of an area in memory (fnnnn-fmmmm) must both appear in the same field.
- When you specify a location or an area on one page, SAVE takes the entire page. If you call for another part of that same page in the same command line, SAVE sends an error message to the terminal, informing you that it has already saved the page.
- If you omit the field number, SAVE assumes field 0.
- Avoid saving locations 7600-7777 in fields 0-2. The resident monitor code resides in these areas of memory. To avoid accidentally destroying a portion of the Monitor, restrict SAVEs involving 7600 to fields above field 2.
- If you specify an address on an odd-numbered page, SAVE can save it only if it also saves the preceding page. The system does this automatically.

3.45.1 The Job Status Word

The Job Status Word, which resides in memory with the file (at location 7746 in field 0), indicates what parts of the file use memory and how.

Bit Condition	Meaning
Bit 0=1	File does not load into locations 0-1777 in field 0 (0000-1777).
Bit 1=1	File does not load into locations 0-1777 in field 1 (10000-11777).
Bit 2=1	Program must be reloaded before it can be restarted because it modifies itself during execution.
Bit 3=1	Program being run will not destroy the BATCH monitor.
Bit 4=1	A memory image file that was generated through the LINKER contains overlays.
Bit 5	Reserved for OS/78 system programs.
Bits 6-9	Unused, and reserved for future expansion.
Bit 10=1	Locations 0-1777 in field 0 need not be saved when calling the Command Decoder overlays.
Bit 11=1	Locations 0-1777 in field 1 need not be saved when calling the USR.

The Monitor runs the SAVE command.

3.46 SET

The SET command enables you to modify the operating characteristics of OS/8 by specifying certain attributes of system programs in the command line. Use SET to make frequently required changes in system programs, especially those with I/O handlers.

The format is

```
SET device [NO] attribute [argument]
```

where

device	indicates the handler of the device you want to modify
NO	indicates that the following attribute does not apply
attribute	is the characteristic you want to modify
argument	is an optional parameter required by certain SET commands

For details, see the chapter on the SET program in this manual.

3.47 SKIP

The SKIP command runs the CAMP program and advances over the number of files or records on a magnetic tape that you specify. The format is

```
SKIP MTAn:  nnnn keyword
```

where

MTAn	is any magnetic tape drive
nnnn	is an unsigned decimal number representing the number of files you want to advance over
keyword	specifies FILE, RECORD, or EOD (end-of-data)

If you omit nnnn or EOD, SKIP assumes 1. If you omit the keyword, SKIP assumes FILE.

For example, this command

```
.SKIP MTA0:2 RECORDS
```

advances the tape on MTA0 forward two records.

For complete information on the SKIP command, see the chapter on the CAMP program in this manual.

3.48 START

The START command begins execution of the memory image program currently in memory at the address you specify in the command line. If you omit the address, START uses the starting location in the current Core Control Block.

The format is

```
START fnnnn
```

where

fnnnn	is a 5-digit octal number representing a field (f) and the location in memory (nnnn) you want to use as a starting address

For example, this command

```
.START 10555
```

starts executing the program currently in memory at location 555 in field 1.

This command

```
.START
```

starts the program at the address contained in the current Core Control Block.

The Monitor runs the START command.

3.49 SQUISH

The SQUISH command eliminates any embedded empty files on the device you specify for input.

The format is

```
SQUISH device
```

Before it executes a command, SQUISH prints a message to ask if you are sure you have specified the right device. For example:

```
.SQUISH RXA1:

ARE YOU SURE?
```

If you want to continue, type Y. If not, type any other character.

If you specify both an input and an output device in the command line, SQUISH copies all the files from the input device to the output device and eliminates any embedded empty files. For example:

```
.SQUISH RXA0:<RXA1
```

If the output device is a system device, SQUISH always preserves the system programs.

NOTE

An error during a SQUISH can corrupt the entire contents of a device in a way that may not be immediately apparent to you. Therefore, do not use SQUISH unless you have a copy of both the system programs and your other files.

SQUISH runs CCL.SV and PIP.SV with the /S option.

3.50 SUBMIT

The SUBMIT command performs batch processing with optional spooling to a file-structured output device. SUBMIT runs multiple programs and sequences of system commands that require little or no interaction with the user or operator.

The format is

```
SUBMIT spool device:<input:file
```

where

spool device:	is optional
input:file	is an input file name and device. If you omit the device and extension, the system assumes DSK and BI.

For a complete discussion of the BATCH program, see the BATCH chapter.

SUBMIT runs CCL.SV and BATCH.SV.

3.51 TECO

The TECO command summons the TECO editor, opens the files you specify for input, and creates an output file.

The format is

```
TECO output:file<input:file1,...input:file5
```

If you omit the output specification, TECO does an edit backup on the input file you specify. If you omit the input file extension, TECO assumes PA.

TECO reads the first page of the input file into the text buffer before it returns control to you.

TECO remembers the name of the output file. Thus, the next TECO command you enter without arguments will use this file for input.

For complete information, see the TECO section in the OS/8 Language Reference Manual.

3.52 TERMINATE

The TERMINATE command causes the system to emulate a terminal with no knowledge of disk drives, processor, or memory. The format is

```
TERMINATE
```

This command runs CCL.SV.

3.53 TYPE

The TYPE command displays on the terminal the contents of the files you specify for input. The format is

```
TYPE input:file1,...input:file5
```

TYPE displays the contents of each input file on the terminal in the same order that you enter them in the command line. Although the command accepts no more than five files in a line, you can extend this number with wildcards.

For example, this command displays all files with a BS extension

```
.TYPE RXA1:*.BS
```

3.53.1 TYPE Options

TYPE provides the following options:

- /C TYPE displays only files with the current date.
- /O TYPE displays only files with a noncurrent date.
- /V TYPE displays all the files on a device except the ones you specify.
- /Q TYPE prints each file name on the terminal, followed by a question mark. To display the file, type Y. To skip it, type any other character.

TYPE runs CCL.SV and FOTP.SV.

3.54 UA, UB, and UC

The UA, UB, and UC commands let you store CCL commands and their arguments in temporary files and recall them for later use. Unlike other commands that remember arguments, UA, UB, and UC do not forget command lines that you have entered on previous days. This is because the system does not delete the files each time the date changes.

The format is

```
UA
UB  } command line
UC
```

where

command line — is the CCL command with arguments that you want to recall

For example, this command

```
.UA COPY RXA1:<DSK:RECALL.BS
```

stores the COPY command and its arguments in a temporary file. To execute the command, type

```
.UA
```

Note that you can store and recall only three commands at a time.

Use UA, UB, and UC to recall command lines that recur throughout a BATCH job.

3.55 UNLOAD

The UNLOAD command

- turns a magnetic tape controller off line and rewinds the tape, returning to the CAMP program during the rewind. To use the magnetic tape after the UNLOAD command, you must turn it on line manually.

- unloads TC08 and TD8E DECtapes from their reels. UNLOAD rewinds the DECtape on the unit you specify, selects a different unit, and returns control to CAMP for another command. This DECtape unit cannot be used until you issue another legal command -- for example, an ASSIGN command -- to the DECtape controller.

- write-locks an RK8E disk.

The format is

```
.UNLOAD device
```

where

device is a magnetic tape, a TC08 or TC8E DECTAPE, an RK8E disk

UNLOAD runs CCL.SV and CAMP.SV.

3.56 VERSION

The VERSION command prints the version numbers of the OS/8 Monitor and the CCL program. The format is

```
VERSION
```

3.57 ZERO

The ZERO command clears the directory of the device you specify, creating an empty file directory. The format is

```
ZERO device
```

For example, the following example clears the directory of RXA1.

```
.ZERO RXA1:
```

Use ZERO only on devices that contain user programs and data files. If you zero the system device, you will destroy the system programs. ZERO will not clear the directory of SYS until it has printed a message to ask if you are sure you want to proceed. If you do, type Y; if you do not, type any other character.

CHAPTER 4

THE OS/8 SYMBOLIC EDITOR

4.1 INTRODUCTION

The Editor allows you to create and modify ASCII source files. These files may contain assembly language programs, FORTRAN and BASIC programs, or any other information that has the format of character strings.

The Editor is a helpful tool; however, it must be told precisely what to do. You direct its operation by typing commands in the form of a single letter or a letter with arguments and, in most cases, pressing the RETURN key directly after the command line.

This chapter describes the procedures you follow to create a file and the commands you use to modify it.

4.2 CALLING THE EDITOR

The CREATE and EDIT commands call and run the OS/8 Editor.

The CREATE command summons the Editor to let you open and write a new file. The format is

```
CREATE outdev:file
```

CREATE accepts no input specifications and only one file name and device for output. You provide the input by typing in text at the terminal.

After you press the RETURN key to execute the command, the system Editor displays a number sign (#) on the screen to indicate that it is ready to receive your first instruction.

Thus,

```
.CREATE RXA1:RUN1.PA
#
```

opens a file named RUN1.PA on output device RXA1.

To enter text, you must put the Editor into text mode with the I or A instruction. (For details on text mode, see Section 4.3.1.)

The EDIT command summons the OS/8 Editor to let you retrieve and work on a source program previously stored as a file. The format is

```
EDIT outdev:file=indev:file1,...indev:file9
```

The Editor signals with a number sign (#) as soon as it is ready to accept your first instruction.

To work on a source program that you have created and stored as a file (or sequence of files), enter the file or files as input in the EDIT command line. EDIT will accept up to nine input files in a line.

The Editor allows only one output file in a command line. You must specify an output file to receive the modified version of your source program.

For example, the following command opens input file TABLE.FT on RXA0 and a file called FILE1.FT on RXA1 for output (The Editor signals when ready.):

```
.EDIT RXA1:FILE1.FT<RXA0:TABLE.FT
#
```

To cause the Editor to read in the first page of the input file, type R in response to the number sign. (For details on Editor commands, see Section 4.3.2.)

4.3 MODES OF OPERATION

The OS/8 Editor operates in two modes: the command mode and the text mode.

In the command mode, the Editor prints a # on the terminal to indicate that it is waiting for you to type a command on the keyboard.

In text mode, the Editor accepts anything you type at the keyboard as part of the file you are creating or modifying.

The key commands in Table 4-1 enable you to transfer between modes or return control to the Keyboard Monitor.

Table 4-1
Editor Key Control Commands

Command	Mode in Which Used	Meaning
CTRL/C	Text and Command Modes	Returns control to the Keyboard Monitor. All text that has been edited is lost. CTRL/C should be used with utmost caution, since no output file will be stored.
CTRL/O	Command Mode	Stops the listing of text. Returns control to Command Mode.
CTRL/L	Text Mode	Returns the Editor to Command Mode.

4.3.1 Text Mode

To put the Editor in text mode so that you can enter a new file -- or add to one that you have already created -- type the Insert or Append command. The format is:

```
I RETURN
```

or

```
A RETURN
```

These commands cause the Editor to place the text that you enter at the terminal into its text buffer. If you use the Insert command, the Editor stores the text before the first line of any existing material in the buffer. The Append command instructs the Editor to place the text you enter after the last line of existing text in the buffer.

The Editor accepts text in both upper and lower case.

To enter a line of text that you have typed on the terminal, press the RETURN key.

For example:

```
*I
HEAD OF THE BUFFER
```

or

```
*A
BOTTOM OF THE BUFFER
```

Before you type RETURN, read the line over for errors. Make corrections with the DELETE key or the CTRL/U key command. DELETE erases the last character you typed. CTRL/U deletes the entire line. (CTRL/U is equivalent to typing DELETE back to the beginning of the line.)

To correct a line that you have sent to the buffer with RETURN, you must put the Editor in command mode with CTRL/L and use the appropriate editing commands (see Section 4.3.2.4).

The buffer holds approximately 5600 characters (decimal). When 256 locations remain, the Editor rings the warning bell on the terminal. From this point until the buffer is full, typing RETURN causes the Editor to enter a line of text, then switch to command mode and ring the terminal bell. You may continue to enter text by this method one line at a time until the Editor detects the absolute end of its buffer.

To continue, you must first empty the buffer. The Page command enables you to send the contents of the buffer -- or any part of it -- to an output device. To use the Page command, return to command mode with CTRL/L. The format of the command is

```
P RETURN
```

or

```
nP RETURN
```

or

```
m,nP RETURN
```

where

n	is a line you want to send to an output device.
n,m	is a sequence of lines (n through m) that you wish to send to an output device

The P command automatically appends a form feed to the output, thus producing a page of text. This allows you to paginate the contents of your file.

Before you start typing in the next page, make sure that no text remains in the buffer. To do this, use the Kill command (see Section 4.3.2.3), which clears the buffer. Then type the Append command (to put the Editor back in text mode) and continue entering your source program.

To return to command mode at any point, type CTRL/L.

To end the session -- that is, to place all remaining text in the output file, close the file, and return control to the Keyboard Monitor -- use the Exit command. The format is:

```
E RETURN
```

4.3.2 Command Mode

In command mode, the Editor performs the operations you specify on the text in the buffer.

To enter text into the buffer from your input device, use the Read command. The format is:

```
R RETURN
```

The Read command instructs the Editor to read a page of text from an input device into the buffer -- that is, to read text until it encounters a form feed character. If the buffer contains text already, the Editor adds the new page to it.

The Editor provides five types of command: Input, Listing, Output, Editing, and Search.

Each command consists of a single letter, preceded optionally by one or two numeric arguments. The letter indicates the operation; the arguments in most cases tell the Editor which lines to act upon.

Enter the commands after the number sign prompt in upper case only. The general format is

```
X RETURN
```

or

```
nX RETURN
```

or

```
m,nX RETURN
```

where

X	is a command
m,n	are line numbers (m must be less than n.)

Except for noted exceptions, you terminate the command with the RETURN key.

4.3.2.1 Input Commands - Input commands instruct the Editor to accept text from the terminal (text that you type in) or from an input device (text that you have stored as a file). To execute the commands, type RETURN key.

Note:

- Special characters, including lower-case letters may be input to the file. The ESCape character is echoed as a dollar sign ($) for readability.
- In these commands, the Editor ignores ASCII codes 340 through 376. These codes include the codes for the lower-case alphabet (ASCII 341-372). The Editor returns to the command mode only after encountering a form feed or when the text buffer becomes full.

Table 4-2
Editor Input Commands

Command	Meaning
A	Append the text being typed at the keyboard until a form feed (ASCII 214 or CTRL/L) is encountered. The form feed returns control to command mode. Text input following the A command is appended to whatever is currently in the text buffer.
I	Insert whatever text is typed before line 1 of the text buffer. The form feed (CTRL/L) terminates the insertion process and returns control to the command mode.
nI	Insert whatever text is typed (until a form feed is typed) before line n of the text buffer.
R	Read one page from the input device specified to the EDIT or CREATE commands, and append the new text to the current contents of the buffer. If no input file was indicated or if no input remains, a question mark (?) is printed and the Editor returns to the command mode.

4.3.2.2 **Listing Commands** - List commands display on the terminal all or part of the contents of the text buffer. Type RETURN key to execute.

Table 4-3
Editor Listing Commands

Command	Meaning
L	List entire contents of the text buffer on the terminal.
nL	List line n of the text buffer on the terminal.
m,nL	List lines m through n of the text buffer on the terminal.
G	Get and list the next line that has a label associated with it. A label in this context is any line of text that does not begin with one of the following: space (ASCII 240) / (ASCII 257) TAB (ASCII 211) RETURN (ASCII 215) At the termination of a G command, control returns to the command mode with the current line counter equal to the line just listed.
nG	Get and list the first line that begins with a label, starting the search at line n.
B	Print the number of available memory locations in the text buffer. The Editor returns the number of locations on the next line. To estimate the number of characters that can be accommodated in this area, multiply the number of free locations by 1.7.

The Editor remains in command mode after a list command and updates the value of the current line counter to be equal to the number of the last line printed.

4.3.2.3 **Output Commands** - Output commands send text from the buffer to a device you specify for output. Type RETURN key to execute.

Table 4-4
Editor Output Commands

Command	Meaning
E	Output the current buffer and transfer all remaining pages of input to the output file; close the output file and enter it in the directory. When this buffer is full, the text is output to the indicated output file. The E command automatically outputs a form feed after the last line of output, and returns control to the Monitor. NOTE If you do not use the E command to close a file after editing, any changes, additions, or corrections will not appear in the output file. Thus, the E command should usually be the last command that you enter in an editing session (also see Q command).
P	Write the entire text buffer to the output file.
nP	Write line n of the text buffer to the output file.
m,nP	Write lines m through n, inclusive, to the output file. NOTE The P command automatically appends a form feed to its output, thus producing a page of text. This command allows you to paginate your listing. However, if the K command is not used after a P command, the text remains in the buffer and is again output with the new text read in before the next P command.
K	Kill the buffer. All text is deleted from the text buffer. NOTE The Editor ignores the commands nK or m,nK, with the result that you cannot destroy the buffer by mistyping a List command (m, nL).

(continued on next page)

Table 4-4 (Cont.)
Editor Output Commands

Command	Meaning
Q	Immediate end-of-file. The Q command causes the text buffer to be output. The file is then closed (entered into the directory with the current date as its creation date), and control returns to the Monitor.
N	Write the current buffer to the indicated output file and read the next logical page. The N command is equivalent to a P, K, R command sequence.
nN	Write the current buffer to the output file, kill the buffer, and read the next logical page. This is done n times until the nth logical page is in the text buffer. Control then returns to command mode. (The N command cannot be used with an empty text buffer, since there is no text to be written. If the buffer is empty when the N command is attempted, a question mark (?) is printed.) For example, to read in the fourth page of a file, give the commands ↑R (to read the first page) and ↑3N (to read three more pages)
V	The V command causes the entire text buffer to be listed on the line printer. The V command only works with the LA78 line printer. It does not work with the LQP78 line printer.
nV	List line n of the buffer on the line printer.
m,nV	List lines m through n, inclusive, on the line printer.

4.3.2.4 Editing Commands - The following commands permit deletion or alteration of text in the buffer. Type RETURN key to execute.

Table 4-5
Editing Commands: Deletion and Alteration

Command	Meaning
nC	Change the text of line n to the line(s) typed after the command is entered (typing a form feed terminates the text input). The C command is equivalent to a D command followed by an I command.
m,nC	Delete lines m through n, and replace with the text line(s) typed after the command is entered. (Typing CTRL/L indicates the end of the changed lines.) The C command utilizes the text collector in altering text.
nD	Delete line n from the buffer.
m,nD	Delete lines m through n from the buffer.
nY	Yank (read) in n pages from the input file into the text buffer, without writing any output. For example, *5Y reads through four logical pages of input, deleting them without producing output. The fifth page is read into the text buffer, and control automatically returns to command mode. NOTE Use this command with caution; it irrevocably deletes the contents of the text buffer.
m,n$pM	Move lines m through n directly before line p in the text buffer. The $ character means that you type the dollar sign key, not ESCape, ALTMODE, or other possibilities. The old occurrence of the moved text is then removed. This command can move one line, but it needs three arguments. You can provide three arguments by specifying the same line number twice. For example, *6,6$21M moves line 6 in front of line 21.

4.3.2.5 Search Commands - Search commands cause the Editor to search a text for occurrences of characters and strings that you specify. The Editor sets the current line pointer at the line containing the characters you want to find.

Search commands are discussed in detail in Section 4.4.

Table 4-6
Editor Search Commands

Command	Meaning
S	Perform a character search (Section 4.4.1).
J	Perform an interbuffer search for character strings (Section 4.4.2.2).
F	Look for next occurrence of the string currently being sought.
ESC($)	Perform an intrabuffer character string search.

4.3.2.6 **Special Command Mode Characters** - The Editor recognizes the following special characters in command mode:

Table 4-7
Editor Special Characters: Command Mode

Character	Function
Period (.)	The Editor assigns an implicit decimal number to the line on which it is currently operating. At any given time the period, which represents this decimal number, may be used as an argument to a command. In the following example, the L command is used since it allows text to be listed. Typing .L means list the current line. Typing .-1,.+1L means list the line preceding the current line, the current line, and the line following it, and then update the current line counter to the decimal number of the last line printed. The Editor updates the current line counter, represented by the period, as follows: • After an R (Read page) or A (Append) command, the period is equal to the number of the last line in the buffer. • After an I (Insert) or C (Change) command, the period is equal to the number of the last line entered.

(continued on next page)

Table 4-7 (Cont.)
Editor Special Characters: Command Mode

Character	Function
Period (.) (Cont.)	• After an L (List) or S (Search) command, the period is equal to the number of the last line listed. • After a D (Delete) command, the period is equal to the number of the line immediately after the deletion. • After a K (Kill) command, the period is equal to 0. • After a G (Get and list) command, the period is equal to the number of the line displayed by the G. • After an M (Move) command, the period is not updated and remains whatever it was before the command.
Slash (/)	The symbol slash (/) has a value equal to the decimal number of the last line in the buffer. It may also be used as an argument to a command. For example, **#10,/L** means list from line 10 to the end of the buffer.
LINE FEED Key	When the Editor is in command mode, pressing the LINE FEED key has the same effect as **#.+1L** which causes the Editor to display the line following the current one and to increment the value of the current line counter (dot) by one. LINE FEED does not perform this function while in the text mode.
Right-Angle Bracket (>)	Typing the right-angle bracket (>) while in command mode is equivalent to typing **#.+1L** and causes the Editor to echo > and then display the line following the current line. The value of the current line counter is increased by one so that it refers to the last line displayed.

(continued on next page)

Table 4-7 (Cont.)
Editor Special Characters: Command Mode

Character	Meaning
Left-Angle Bracket (<)	In command mode, typing the left-angle bracket (<) is equivalent to typing ‡.-1L and causes the Editor to echo < and then print the line preceding the current line. The value of the current line counter is decreased by one so that it refers to the last line printed.
Equal Sign (=)	In the command mode, using the equal sign in conjunction with either the line indicator period (.) or slash (/) causes the Editor to display the decimal value of the argument preceding it. You can find by this method the number of the current line (.=nnnn) or the total number of lines in the buffer (/=nnnn).
Colon (:)	The colon performs exactly the same function as the equal sign (=).
ESCape Key	When the Editor is in command mode, pressing the ESCape key signals an intrabuffer character search. It echoes as a dollar sign ($) on the terminal screen. When the Editor is in text mode, the Escape key echoes as a dollar sign, but it is stored in the file as an ESCape character (033).

4.4 SEARCHING A TEXT

The following search commands enable you to make additions and corrections in your text. The Editor searches for occurrences of the single character or character string that you specify.

4.4.1 Single-Character Search -- the S Command

The format of a single-character search is:

```
S RETURN
X
```

where

X is the alphanumeric character you want to search for.

To specify a line or a sequence of lines that you want to search, use the following format:

```
nS RETURN
n,mS RETURN
```

For example, the following command causes the Editor to search lines 20-40 for an occurrence of the character B:

```
#20, 40S
B
```

The Editor displays the character it is searching for and everything preceding it in the line. At this point you can perform the following operations.

- Delete the entire portion of the line not yet displayed and terminate the line and the search by pressing the RETURN key.
- Delete characters from right to left by typing the DELETE key.
- Insert characters after the last one printed simply by typing them.
- Insert a carriage return/line feed, thus dividing the line into two, by pressing the LINE FEED key followed by CTRL/L.
- Continue searching the line to the next occurrence of the search character by typing CTRL/L.
- Change the search character in the line and continue searching by typing CTRL/G(BELL) followed by the new search character. This allows all editing to be done in one pass.
- Type CTRL/G(BELL) twice to terminate the command.

The usual form of the character search command is #.S, followed by the RETURN key and the character to be located. Use this form of the command to modify the current line.

4.4.2 The Character String Search

The Editor can search the buffer for any unique combination of characters. In a character string search, the Editor sets the current line pointer at the line containing the first occurrence of the string.

Two types of character string search are available: intrabuffer and interbuffer.

4.4.2.1 Intrabuffer String Search - In an intrabuffer search, the Editor scans the text in the buffer for the string you specify. If it fails to find an occurrence of the string, it prints a question mark and returns to command mode.

To initiate an intrabuffer search, type the ESCape key in response to the Editor's prompt and enter the string. (ESCape echoes as a dollar sign.) The string must occur in one line.

If you wish to begin the search at line 1 of the buffer, terminate the string with a single quotation mark ('). If you wish to begin the search at the current line + 1, use a double quotation mark (") to terminate the string.

The format of an intrabuffer search command is:

```
$string' RETURN
```

or

```
$string" RETURN
```

where

$	is a prompt character printed by the Editor
string	is a group of up to 20 ASCII characters
' (single quote)	causes the Editor to begin searching at line 1 of the buffer
" (double quote)	causes the Editor to begin searching at current line +1

NOTE

Do not include single or double quotation marks in a string because the Editor recognizes them as instructions.

The Editor places the number of the first line containing the search string in the current line indicator and displays the prompt sign (*). To display the number on the terminal, type the indicator dot (.) followed by an equal sign. The format is:

```
.=
```

You can use a line number you obtain this way as an argument in any Editor command.

For example, the following command causes the Editor to search for the first occurrence of the string CDF10, beginning at line 1 of the buffer:

```
*$CDF10'
*
```

The response to this command is revealed as line 35:

```
*.=35
```

Command lines can include more than one instruction. For example, assume that the buffer contains the following text:

```
ABC DEF GJO
1A2B3C4D5E6
.STRINGABCD
.
.
```

To list the line that contains ABC, type

```
*$ABC'L
```

The search begins with line 1 and continues until the Editor finds the string. The Editor sets the current line counter equal to the line in which the string ABC occurred. The L (List) command causes the line to be printed as follows:

```
ABC DEF GJO
```

The Editor returns to command mode, awaiting further commands. If you want to find the next reference to ABC, type:

```
*"L
```

In this case, the quotation marks (") cause the last string the Editor searched for to be used again, with the search beginning at the current line +1. It is not necessary to enter the search string again. The command may be used several times in succession. For example, if you want to find the fourth occurrence of a string containing the characters FEWMET, type

```
*$FEWMET' """L
```

This command will list the line which contains the fourth occurrence of that string. The L command (or any other command code) can follow either ' or ". The L command causes the line to be listed if the Editor finds the string.

To clear the text string buffer, type

```
*$'
```

The Editor responds with a question mark and clears the text string buffer.

The properties of the commands ' and " allow for easy and useful editing, as the following example illustrates. To change the CIF 20 to CIF 10, enter the following commands:

```
*$DUM, '$CIF 20"C
CIF 10      /NEW FIELD
```

The above set of instructions first causes the Editor to start at line 1 and search for the line beginning with DUM,. Then it searches for CIF 20, starting from the line after the line containing DUM,. The line number of the line containing the string CIF 20 becomes the current line number. The C command applies the instructions of the command line to what is typed in the next line -- that is, the string CIF 10.

Since this search feature produces a line number as a result, any operations which require a line number will accept a string instead. For example:

```
*$STRING'+4L
```

lists the fourth line after the first occurrence of the text STRING in the text buffer.

```
*$LABEL1,',$LABEL2,"L
```

lists all lines between the two labels, inclusive.

```
*$PFLUG'S
```

performs a character search on the line which contains PFLUG. (Type the search character after typing the RETURN key that enters the line.)

In commands that include both strings and explicit numbers, strings should appear first. For example, the following commands:

```
*1+$BAD!'L
```

will not list the next line after the string BAD! occurs. The correct syntax is:

```
*$BAD!'+1L
```

4.4.2.2 **Interbuffer String Search -- J Command** - In an interbuffer search, the Editor scans the contents of the text buffer for the character string you specify. If it fails to find an occurrence of the string, it sends the buffer to an output file, clears the buffer, and reads in the next page of text from the input file. The Editor then resumes the search at line 1 of the new buffer. When the input file is exhausted, the Editor prints the number sign prompt (#) and awaits your next instruction.

If the search is successful, the Editor sets the current line indicator equal to the number of the line containing the first occurrence of the string.

The format for an interbuffer search is:

```
J RETURN
$string'
```

where

$	is a prompt character printed by the Editor
string	is a group of up to 20 ASCII characters
' (single quote)	causes the Editor to begin searching at line 1 of the buffer.

To display the number of the line containing the string, type the current line indicator dot (.) after the Editor's number sign prompt (#), followed by an equal sign. The format is:

```
.=
```

For example, the following command instructs the Editor to make an interbuffer search for the string WRITE, beginning at line 1 of the current buffer. The .= construction reveals that line 4 of the current buffer contains the string.

```
*J
$WRITE'
*.=0004
```

To find further occurrences of the string WRITE, type the F command. The F command searches the buffer for the last character string entered, starting from the current line count + 1. The displayed line following the F command line contains a number prompt sign (#), the format you type to obtain a line number (.=), and the line number. The result is:

```
#F

#.=0008
```

This example causes a search for the string WRITE, starting at the current line + 1. If you have specified no output file, the J or F command reads the next input buffer without attempting to produce any output.

NOTE

Use the J command for interbuffer searches only. After the J or F command has processed the entire input file, execute either an E or Q command to close the output file.

The following commands may be used to abort the string search command, once given:

Table 4-8
Aborting Editor String Search Commands

Command	Explanation
CTRL/U	A CTRL/U will return control to the Editor command mode if you type it while entering text in a string search command.
DELETE	Pressing the DELETE key while entering text for a string search causes the text so far entered to be ignored and allows a new string to be inserted. The Editor displays a dollar sign ($) in response.

4.5 EDITOR OPTIONS

The Editor provides the following options:

- /B The Editor converts two or more spaces to a TAB when reading from an input device.
- /D The Editor deletes the old copy of the output file (if one exists) before opening the new output file on the device. If you do not specify /D, the Editor does not delete the old copy of the output file until you have transferred all data to the new file with the E or Q command.

4.6 EDITOR ERROR MESSAGES

Two types of error messages, nonfatal and fatal, are generated when an error is made while running the Editor.

Nonfatal errors, such as an incorrect format in a command string or a search for nonexistent information, cause the Editor to display a question mark. For example, if a command requires two arguments, and only one is provided, the Editor will display a question mark (?), perform a carriage return/line feed, and ignore the command as typed. Similarly, if you type an illegal or unrecognized command character, the error message ? will be displayed, followed by a carriage return/line feed; the command will be ignored. However, if you provide an argument for a command that does not require one, the argument may be ignored and the normal function of the command performed. The following examples illustrate nonfatal errors that you may encounter while using the Editor.

Table 4-9
Nonfatal Editor Error Messages

Condition/Message	Explanation
L ?	The buffer is empty. Nonexistent information is requested.
7,5L ?	The arguments are in the wrong order. The Editor cannot list backward.
17$10M ?	This command requires two arguments before the $; only one was provided.
H ?	Nonexistent command letter.

Major errors cause control to return to the Monitor and may be due to one of the causes listed in Table 4-2. These errors cause a message to be printed in the form

```
?n ^C
```

where

n is an error code listed in the table

^C indicates that control has passed to the Monitor.

These errors generally result in complete loss of the output file.

Table 4-10
Editor Error Codes

Error Code	Meaning
0	Editor failed in reading a device. Error occurred in device handler; most likely a hardware malfunction.
1	Editor failed in writing onto a device; generally a hardware malfunction.
2	File close error occurred. For some reason the output file could not be closed; the file does not exist on that device.
3	File open error occurred. This error occurs if the output device is a read-only device or if no output file name is specified on a file-oriented output device.
4	Device handler error occurred. The Editor could not load the device handler for the specified device. This error should not normally occur.

4.7 SUMMARY OF EDITOR COMMANDS AND SPECIAL CHARACTERS

The command and special characters discussed in this chapter are summarized in Table 4-11.

Table 4-11
Editor Command and Special Characters

Command	Format	Meaning
A	A	Append the following text being typed at the keyboard until a CTRL/L (form feed) is typed. The form feed returns control to the command mode. Text input following the A command is appended to whatever is present in the text buffer.
B	B	List the number of available memory locations in the text buffer. The Editor returns the number of locations on the next line. To estimate the number of characters that can be accommodated in this area, multiply the number of free locations by 1.7.
C	nC	Change the text of line n to the line(s) typed after the command is entered. (Typing a CTRL/L terminates the input.)

(continued on next page)

Table 4-11 (Cont.)
Editor Command and Special Characters

Command	Format	Meaning
C (Cont.)	m,nC	Delete lines m through n and replace with the text line(s) typed after the command is entered. (Typing CTRL/L indicates the end of the inserted lines.)
D	nD	Delete line n from the buffer.
	m,nD	Delete lines m through n from the buffer.
E	E	Output the text buffer and transfer all remaining pages of the input file to the output file, closing the output file and returning to the Monitor.
F	F	Follows a string search. Look for next occurrence of the string currently being sought (by the J command).
		NOTE If the search fails while you are using the F command, further commands cause the system to prompt with a ?. The file must be closed and then reopened.
G	G	Get and list the next line that has a label associated with it. A label in this context is any line of text that does not begin with one of the following: space (ASCII 240) / (ASCII 257) TAB (ASCII 211) RETURN (ASCII 215) At the termination of a G command, control goes to the command mode with the current line indicator (.) equal to the line just listed.
	nG	Get and list the first line that begins with a label, starting the search at line n.
I	I	Insert whatever text is typed before line 1 of the text buffer. (Typing CTRL/L terminates the entering process and returns control to the Editor command mode.)
	nI	Insert whatever text is typed (until a CTRL/L is typed) before line n of the text buffer.

(continued on next page)

Table 4-11 (Cont.)
Editor Command and Special Characters

Command	Format	Meaning
J	J	Interbuffer search command for character strings (see Section 4.4.2.2 describing the InterBuffer Character String Search).
		NOTE If the search fails while you are using the J command, further commands cause the system to prompt with a ?. The file must be closed and then reopened.
K	K	Kill the buffer. Delete all text from the text buffer.
		NOTE The Editor ignores the commands nK and m,nK with the result that you cannot destroy the buffer by mistyping a List command (m,nL).
L	L	List entire contents of the text buffer on the terminal.
	nL	List line n of the text buffer on the terminal.
	m,nL	List lines m through n of the text buffer on the terminal. Control then returns to command mode.
M	m,n$pM	Move lines m through n directly before line p in the text buffer. The $ character represents typing the dollar sign key, and not other possible keys. The old occurrence of the moved text is removed.
N	N	Write the current buffer to the output file and read the next page.
	nN	Write the current buffer to the output file, kill the buffer, and read the next page. This action is repeated n times until the nth page is in the text buffer. Control then returns to command mode. You may not use the N command with an empty text buffer. A question mark (?) is printed if you attempt to do this.

(continued on next page)

Table 4-11 (Cont.)
Editor Command and Special Characters

Command	Format	Meaning
P	P	Write the entire text buffer to the output file. The P command automatically outputs a FORM character (214) after the last line of output.
	nP	Write line n of the text buffer to the output file and a FORM character.
	m,nP	Write lines m through n, inclusive, to the output file and a FORM character.
Q	Q	Immediate end-of-file. Q causes the text buffer to be output and the file closed.
R	R	Read one page from the input device and append the new text to the current contents of the text buffer. If no input file was indicated or if no input remains, a question mark (?) is displayed and control returns to the command mode.
S	S	Character search command (see Section 4.4.1).
V	V	List the entire text buffer on the line printer.
	nV	List line n of the text buffer on the line printer.
	m,nV	List lines m through n, inclusive, on the line printer.
Y	nY	Yank (read) in a logical page from the input file, without writing any output. For example, #5Y reads through four logical pages of input, deleting them without producing output. The fifth page is read into the text buffer, and control automatically returns to the command mode.
$(ESC)	$TEXT' "	Perform a character string search for the string TEXT. Following a string search, #" causes a search for the next occurrence of the string (see Section 4.4.2.1 describing the Intrabuffer Character String Search).

(continued on next page)

Table 4-11 (Cont.)
Editor Command and Special Characters

Command	Format	Meaning
.= or .: /= or /:		Typing these characters obtains the current line number (.=) and the last line number (/=) in the text buffer. The number is printed by the Editor immediately after you type the equal sign. (The colon character is equivalent to the equal sign.)
>	>	Equivalent to .+1L, list the next line in the text buffer.
<	<	Equivalent to .-1L, list the preceding line in the text buffer.
LINE FEED Key		Equivalent to .+1L, list the next line in the text buffer.
#	#	Print the current Editor version number.

CHAPTER 5

THE COMMAND DECODER

The Command Decoder is a subroutine that OS/8 system programs use to interpret the I/O specifications for devices and files and for options that you enter in the command line.

5.1 ENTERING I/O SPECIFICATIONS

When you run a utility program with the R command, the program calls the Command Decoder, which prints an asterisk to request your I/O specifications.

I/O specifications to the Command Decoder have the following general format (Device and file names must adhere to the conventions described in Chapter 2.):

```
.R utility
*output specs<input specs/options
```

The Command Decoder accepts 0 to 3 output files and 0 to 9 input files. Keep in mind, however, that the system programs using the Command Decoder determine their own I/O requirements and restrictions. These are described in the chapters on system programs in this manual.

Table 5-1 contains examples of legal output specifications to the Command Decoder.

Table 5-1
Examples of Output to the Command Decoder

File Format	Meaning
EXPLE.EX	Output to a file named EXPLE.EX on device DSK (the default file storage device).
LPT:	Output to the LPT. This format generally specifies a nonfile-structured device.
DTA2:EXPLE.EX	Output to a file named EXPLE.EX on device DTA2.
DTA2:EXPLE.EX[99]	Output to a file named EXPLE.EX on device DTA2. A maximum output file size of 99 blocks is specified.
null	No output specified.

Table 5-2 contains examples of legal input specifications to the Command Decoder.

Table 5-2
Examples of Input to the Command Decoder

File Format	Meaning
DTA2:INPUT	Input from a file named INPUT.df on device DTA2. "df" is the assumed input file extension specified in the Command Decoder.
DTA2:INPUT.EX	Input from a file named INPUT.EX on device DTA2. In this case .EX overrides the assumed input file extension.
INPUT.EX	Input from a file named INPUT.EX. If there is no previously specified input device in the command line, input is from device DSK, the default file storage device; otherwise, the input device is the same as the last specified input device.
PTR:	Input from device PTR; nonfile-structured devices do not require a file name.
DTA2:	Input from device DTA2 treated as a nonfile structured device, as, for example, in the PIP command line: *TTY:/L<DTA2: No LOOKUP operation is performed in the last two formats because in each case the device is assumed to be nonfile structured.
null	Repeats input from the previous device specified (must not be first in input list, and must refer to a nonfile-structured device). For example: * <PTR:,, (two null files) indicates that three paper tapes are to be loaded.

NOTE

Whenever you omit a file extension from an input file specification, the Command Decoder performs a LOOKUP for the given name to which the system program has appended an assumed extension If the LOOKUP fails, a second LOOKUP is made for the file to which a null (zero) extension has been appended.

5.2 COMMAND DECODER ERROR MESSAGES

If the Command Decoder detects an error in the command line, it prints one of the error messages in Table 5-3. After the error message, the Command Decoder starts a new line, prints an asterisk (*), and waits for another command line.

Table 5-3
Command Decoder Error Messages

Error Message	Meaning
ILLEGAL SYNTAX	The command line is formatted incorrectly.
TOO MANY FILES	More than 3 output files or 9 input files were specified (in special mode, more than 1 output file or more than 5 input files).
device DOES NOT EXIST	The specified device name does not correspond to any permanent device name or to any user-assigned device name.
name NOT FOUND	The specified input file name was not found on the selected device.

5.3 THE CCL AND THE COMMAND DECODER

The CCL uses its own copy of the Command Decoder instead of the copy available from the Monitor. Thus, the CCL Command Decoder has several options not available via standard USR calls to the OS/8 Command Decoder, for example, multiple default extensions.

For complete information on the Command Decoder, see the OS/8 Software Support Manual.

CHAPTER 6

BATCH

6.1 INTRODUCTION

OS/8 BATCH provides PDP-8 users with a batch processing monitor that is integrated into the OS/8 monitor structure. The system is organized in such a way that it may be used in either a keyboard input configuration or as a batch stream processor.

BATCH may be run on any OS/8 system equipped with at least 12K of memory. A line printer, although optional, is highly desirable. BATCH will support up to 32K of memory and any I/O devices that are present in the system.

OS/8 BATCH processing is ideally suited to frequently run production jobs, large and long-running programs, and programs that require little or no interaction with the user. BATCH permits you to prepare a job on punched cards, high-speed paper tape or the OS/8 system device and leave it for the computer operator to start and run. Output is returned to you in the form of line printer and/or teleprinter listings that include program output as well as a comprehensive summary of all action taken by the user program, the monitor system and the computer operator.

BATCH provides optional spooling of output files. This feature serves to increase throughput on any system, but it is particularly valuable when a line printer is not available. BATCH also performs extensive command analysis and error diagnosis, as well as detailed interaction with the user/operator to facilitate initializing the system and establishing system parameters.

Almost any program that runs under interactive OS/8 may also be run under BATCH. Since BATCH is called from the keyboard in the same manner as any other system program, interactive users may use BATCH to execute multiprogram utility routines, even when continuous batch processing is not desired.

With a few exceptions, BATCH uses the standard OS/8 command set. This chapter assumes that you are familiar with the operation and use of OS/8.

6.2 BATCH PROCESSING UNDER OS/8

OS/8 BATCH maintains an input file and an output file. The BATCH input file may be a punched card, high-speed paper tape, disk, or DECtape file consisting of a series of BATCH commands. If the input file is a disk or DECtape file, it must reside on the OS/8 system device or on a device whose handler is coresident with the OS/8 system device (e.g., RKB0 on RK05 systems).

6.2.1 Input Files

Each command in the BATCH input file occupies one file record. If the file is a punched card file, each punched card constitutes one record, which must contain one complete BATCH command. If the file resides on paper tape, disk, or DECtape, each record consists of one logical line or of all the characters between two line terminators, including the second terminator.

6.2.2 Output Files

The BATCH output file is a line printer listing on which BATCH prints job headers, certain messages that result from conditions within the input file, an image of each record in the input file, and certain types of user output. If a line printer is not present in the system, the output file is printed on the terminal.

6.2.3 I/O Devices

BATCH accepts user input files (i.e., program and data files) from any device in the OS/8 system; however, high-speed paper tape input files are not allowed when the BATCH input file also resides on high-speed paper tape. User output files may be directed to any output device in the system.

6.2.4 Spooling

You may optionally spool output files with BATCH. When you request spooling, every output file is assigned a file name from a list of names maintained by BATCH and directed to a file-structured spool device instead of to the user-specified device. Spooling of output files increases BATCH throughput when system resources are scarce and permits you to postpone slow output operations until a more favorable time. For example, you may initialize a batch processing run that generates many output listings so that it reroutes all listings from the terminal or line printer to a specified DECtape unit. You may then dump this DECtape onto the appropriate hard copy device after the run, when more time is available. The spool device may be any file-structured device you select.

To call OS/8 BATCH from the keyboard, type

```
.R BATCH
```

in response to the dot generated by the OS/8 monitor. BATCH then calls the OS/8 Command Decoder to obtain its parameters, input device, and file name (if file-structured). If CCL is enabled, you may also invoke BATCH via the SUBMIT command, in which case the BATCH parameters, input device, and file name (if file-structured) are specified on the same line as the SUBMIT command.

6.2.5 Entering File Specifications

The format for a BATCH command string is:

```
*SPDV:  <DEV:INPUT/option/option
```

where

SPDV: is the device on which to spool nonfile-structured output. If you do not specify SPDV:, no spooling is performed. Note that spooling applies only to nonfile-structured output devices specified to the Command Decoder. BATCH does not spool the output of programs such as FOTP, which use a special mode of the Command Decoder.

DEV:INPUT is the input device and file if the input is from SYS: or a device whose handler is coresident with SYS:.

The default extension for BATCH input files is .BI.

The Run-Time Options are used to specify input from the paper tape reader or the card reader. The Run-Time Options and their meanings are listed in Table 6-1.

Table 6-1
Run-Time Options

Option	Meaning
/C	Read the input file from the card reader (CR8/I or CR8/E).
/E	Treat OS/8 Keyboard Monitor and OS/8 Command Decoder errors as nonfatal errors. If /E is not specified, OS/8 Keyboard Monitor and OS/8 Command Decoder errors cause the current job to be aborted.
/P	Read the input file from the paper tape reader.
/Q	Do not output a BATCH log. $JOB and $MSG are the only line output to the terminal.
/T	Output the BATCH log to the terminal. You need only specify this option when a line printer is available. If a line printer is not available, the BATCH log is automatically output to the terminal.
/H	Process the batch input file without echoing and without sending the $JOB and $END batch monitor commands to either terminal or BATCH log.
/U	BATCH will not pause for operator response to $MSG lines. Any attempt to use TTY:, PTR:, or CDR: as input devices to the Command Decoder in an unattended BATCH stream will cause the current job to be aborted.

(continued on next page)

Table 6-1 (Cont.)
Run-Time Options

Option	Meaning
/V	Print the version number of OS/8 BATCH on the terminal.
/6	Accept card input in DEC 026 format. This option is used only when the /C option is specified. The default card input format is DEC 029. NOTE When running BATCH, do not move the input file. In particular, do not SQUISH the device containing the BATCH input file. Moving the input file while BATCH is running produces unpredictable results. If you must SQUISH SYS under BATCH, place the BATCH input file at the beginning of SYS so it will not move. In addition, avoid moving SYS:BATCH.SV while BATCH is running.

6.3 BATCH MONITOR COMMANDS

A BATCH command is a character or string of characters that begins with the first character of a record in the BATCH input file. If the input file is a disk, DECtape or paper tape file, each BATCH command must be followed by a carriage return/line feed combination. If the input file is a punched card file, each command must begin in the first column of a punched card. Disk and paper tape files may contain form feed characters. Form feed characters are ignored by BATCH on input.

OS/8 BATCH recognizes four monitor level commands. These commands allow routine housekeeping operations in a multi-job, batch processing environment and provide communication between the BATCH programmer and the computer operator. Table 6-2 lists the BATCH monitor commands, which may be considered as an extension of the OS/8 Keyboard Monitor command set. Note that the first character of the $JOB, $MSG, and $END commands is a dollar sign. The BATCH monitor does not recognize the ALTMODE character.

In the current version, any record that begins with a dollar sign character but is not one of the BATCH monitor commands listed in Table 6-2 is copied onto the output file and ignored by BATCH.

6.3.1 Defining a BATCH Job

A BATCH processing job consists of a $JOB command record and all the commands that follow it up to the next $JOB or $END record. Normally, all the commands submitted by one user are processed as a single job, and all output from these commands appears under one job header.

Table 6-2
BATCH Monitor Commands

Command	Meaning
$JOB	Initialize for a new job and print a job header on the output file. The remainder of the $JOB record is included in the job header but is ignored by BATCH. It should be used for identifying jobs, and correlating output from the teletype, line printer, and spool device.
$MSG	Ring the terminal bell and print an image of the record at the teleprinter. If you do not specify the /U option, implying that an operator is present, BATCH pauses until any key is struck at the keyboard. If you do specify the /U option, processing continues uninterrupted.
$END	Terminate batch processing and exit to the OS/8 Keyboard Monitor. A $END command record should be the last record of every BATCH input file.
/	Copy the record onto the output file, then ignore it. BATCH assumes that every record beginning with a slash is a comment.

After BATCH encounters a $JOB command, it scans the input file until it finds a Keyboard Monitor command. Any records that follow the $JOB command and precede the first Keyboard Monitor command are written onto the output file and ignored by BATCH.

NOTE

To abort a BATCH job or a sequence of jobs, use the console HALT switch and manually branch to location 7000 in the highest field. This causes BATCH to scan its input for the next $JOB command.

6.3.2 Using OS/8 Keyboard Commands

The first character of every Keyboard Monitor command record is a dot (.). The rest of the record contains an OS/8 Keyboard Monitor command, which should appear in standard OS/8 format. However, commands that would be terminated with an ALTMODE under interactive OS/8 should be terminated with a dollar sign under BATCH. Every standard OS/8 Keyboard Monitor command is legal input to BATCH; however, the ODT command will go to the terminal for input instead of to the BATCH file. Typing CTRL/C to ODT will terminate BATCH. Type 7600G to ODT to resume the BATCH run.

BATCH executes a Keyboard Monitor command by stripping off the initial dot character and loading the remainder of the record into the Keyboard Monitor buffer. BATCH then passes control to the Keyboard Monitor, which executes the command as though it had been typed at the keyboard.

Keyboard Monitor commands that return control to the monitor level should be followed by a BATCH monitor command or another Keyboard Monitor command. Keyboard Monitor commands that transfer control to the program level should be followed by a Command Decoder file specificationwhenever the running program calls the Command Decoder. All OS/8 V3 CCL commands are legal under BATCH, including the SUBMIT command (which can be used to chain from one BATCH stream to another).

6.3.3 Using the Command Decoder

When a running program calls the Command Decoder, the Command Decoder determines whether batch processing is in progress and, if so, instructs BATCH to read the next record of the BATCH input file. BATCH expects this record to contain a Command Decoder file specification.

The first character of every Command Decoder file specification record is an asterisk (*). The rest of the record contains an OS/8 Command Decoder file (and/or option) specification, which should appear in standard OS/8 format. As with BATCH monitor commands and Keyboard Monitor commands, any Command Decoder specification that would be terminated with an ALTMODE under interactive OS/8 should be terminated with a dollar sign under BATCH.

BATCH executes a Command Decoder file specification by stripping off the initial asterisk character and loading the remainder of the record into the Command Decoder buffer. BATCH then passes control to the Command Decoder, which decodes the file specification as though it had been typed at the keyboard and returns control to the running program.

6.3.4 Additional Features

If BATCH reads a record from the input file, expecting to find a Command Decoder file specification, and finds a Keyboard Monitor command instead, BATCH returns control to the monitor level by recalling the Keyboard Monitor to execute the command. The running program is thus terminated; control remains at the monitor level. If BATCH encounters a BATCH monitor command when it expects to find a Command Decoder specification, it executes the BATCH monitor command and continues processing the input file. As long as a Command Decoder file specification is read before the next Keyboard Monitor command, control will eventually return to the running program, and the file specification will be executed.

A BATCH monitor command is legal at any level of command execution, and the BATCH monitor returns control to the level from which it was entered. Keyboard Monitor commands are also legal at any level (under BATCH, but not under interactive OS/8); however, the Keyboard Monitor terminates any program that may be running when it is called and returns control to the monitor level.

The computer operator may type CTRL/C at any time during a batch processing run. Typing CTRL/C at the program level causes an effective jump to location 07600, which recalls the BATCH monitor. The BATCH monitor then recognizes the CTRL/C and terminates the BATCH run.

6.4 THE BATCH INPUT FILE

Figure 6-1 shows a listing of a BATCH input file. This listing represents the output that you obtain by using PIP to transfer the BATCH input file from disk to the console terminal. Assume that OS/8 BATCH is loaded on a 12K system containing one TU56 dual DECtape transport, a line printer, a Teletype terminal, and a disk as the system device. If you specify the disk file shown in Figure 6-1 as an input file, BATCH will begin processing by printing a job header and executing the DATE command.

```
$JOB OS/8 BATCH PROCESSING EXAMPLE #1
.DATE 3/5/74
.R PIP
/LIST SYSTEM DEVICE DIRECTORY ON TELETYPE
*TTY:<SYS:/F
/NOW LIST THE DIRECTORY OF DECTAPE #3 ON THE LPT
$MSG MOUNT TAPE #3 ON UNIT 1
*LPT:<DTA1:/L
/NOW TRANSFER FORTRAN SOURCE PROGRAM
/FROM DISK TO DECTAPE #3 (UNIT 1)
$MSG WRITE ENABLE UNIT 1
*DTA1:FORTS1.FT<DSK:FORTS1.FT
/COMPILE FORTRAN SOURCE
.R FORT
*DTA1:FORTS1.RL,FORTS1.LS<FORTS1.FT
/THAT CONCLUDES JOB #1
$JOB OS/8 BATCH PROCESSING EXAMPLE #2
$MSG MOUNT TAPE #2 ON UNIT 1, WRITE ENABLED
.R PAL8
*PTP:,DTA1:PROG.LS<DTA1:PROG.PA
.RUN DSK CREF
*DTA1:PROG.LS
/END OF EXAMPLE #2 AND END OF INPUT FILE
$END
```

Figure 6-1 Sample BATCH Input File

Control remains at the monitor level, so BATCH executes the next command by calling and starting the Peripheral Interchange Program. PIP, in turn, calls the Command Decoder, which accepts and decodes the file/option specification that occupies the next executable record (following the comment) of the input file. The Command Decoder passes control to the program level, and PIP lists the short form of the system disk directory at the terminal.

If spooling is active, BATCH intercepts this output and stores it in a temporary file on the spool device. Assuming that DTA0 is the spool device and that this listing is the first nonfile-structured output file intercepted by BATCH, the output is stored in a file named BTCHA1. BATCH then prints the message:

```
$SPOOL TO FILE BTCHA1
```

on both the console terminal and the line printer. The next file rerouted to the spool device is assigned the file name BTCHA2; successive files are named:

```
BTCHA3
BTCHA4
.
.
.
BTCHA9
BTCHB0
BTCHB1
.
.
.
BTCHZ9
```

allowing a total of 260 spool device files, which is more than adequate in view of the maximum size of the OS/8 file directory (about 240 entries). If output to a spool device file is generated by a program that appends a default extension to output file names, the spool device file is assigned a standard default extension. You may then transfer all the spool device files to the terminal or line printer by using the program FOTP, with the input file specification dev:BTCH??.*.

Returning to the example of Figure 6-1, PIP executes the file specification that appears in the fifth record of the input file and recalls the Command Decoder.

The Command Decoder then instructs BATCH to scan the input file for the next file specification record. BATCH processes the comment record by copying it onto the line printer, then processes the $MSG command by ringing the terminal bell, copying the $MSG record onto the terminal, and, assuming that an operator is present, pausing until any key is typed at the terminal.

Once the operator has resumed processing by typing any character, BATCH reads the eighth record in the file, recognizes it as a Command Decoder specification record, and transfers control back to the Command Decoder.

Processing continues in this manner until the third Command Decoder specification record is read. When BATCH searches for the next file specification record, it reads and executes the last $MSG command, then encounters a Keyboard Monitor command. BATCH passes this command to the Keyboard Monitor, which terminates PIP and calls the FORTRAN compiler to load and compile source program FORTS1. Upon completion of these operations, FORTRAN routes its output to the specified files and returns control to the monitor level. BATCH then encounters the second $JOB record, causing it to terminate the current job and print a new header.

The second job calls PAL8 to assemble a source program from disk. The output listing is directed to DECtape #2, mounted on unit 1, while the binary output file is dumped onto high-speed paper tape. The job concludes by running CREF to produce a cross-referenced listing of the assembled program.

This job illustrates how you may use OS/8 BATCH to execute multiprogram utility routines. If user #2 is a programmer who usually follows a PAL8 assembly by running CREF, job #2 could be a utility routine that combines the call to PAL8, the call to CREF, and both file specifications into a single software package that you may run under batch processing or in an interactive environment.

The $END record that appears as the last record in Figure 6-1 serves as a signal that batch processing has concluded; it causes BATCH to recall the Keyboard Monitor and reestablish interactive processing under OS/8. This command is always the last record of the BATCH input file.

6.5 BATCH ERROR MESSAGES

BATCH generates two types of error messages. The first type is a run-time error message that appears in the form:

```
#BATCH ERR
```

The second type of error message is generated when the Keyboard Monitor or the Command Decoder recognizes a command error in the BATCH input file. When this occurs, either the Keyboard Monitor or the Command Decoder transmits a standard OS/8 error message, and BATCH will append a "#" character to the beginning of the message, so that it appears in the form:

```
#SYSTEM ERROR
```

Any occurrence of a Keyboard Monitor or Command Decoder error normally causes BATCH to abort the current job and scan the input file for the next $JOB command. If the /E option was specified, BATCH treats Keyboard Monitor and Command Decoder errors as nonfatal and continues the BATCH run.

Table 6-3 lists the BATCH error messages, their meanings, and the probable cause for the error.

Table 6-3
BATCH Error Messages

Message	Meaning
#MONITOR OVERLAYED	The Command Decoder attempted to call the BATCH monitor to accept and transmit a file specification, but found that a user program had overlayed part or all of the BATCH monitor. Control returns to the monitor level, and BATCH executes the next Keyboard Monitor command.
#BAD LINE. JOB ABORTED	The BATCH monitor detected a record in the input file that did not have one of the characters dot, slash, dollar sign, or asterisk as the first character of the record. The record is ignored, and BATCH scans the input file for the next $JOB record.

(continued on next page)

Table 6-3 (Cont.)
BATCH Error Messages

Message	Meaning
#SPOOL TO FILE BTCHA1	(Where the "A" may be any character of the alphabet and the "1" may be any decimal digit.) BATCH has intercepted a nonfile-structured output file and rerouted it to the spool device. This is not, generally, an error condition. Spool device file names are assigned sequentially, beginning with file BTCHA1. Standard default extensions may be assigned by some system programs.
#MANUAL HELP NEEDED	BATCH is attempting to operate an I/O device, such as PTR or TTY, that will require operator intervention. If the initial dialogue indicated that an operator is not present, this message is suppressed, the current job is aborted, and BATCH scans the input file for the next $JOB command record. If an operator is present, he should have been notified what action to take by a $MSG command.
#ILLEGAL INPUT	A file specification has designated TTY or PTR as an input device although the initial dialogue indicates that an operator is not available. The current job is aborted, and BATCH scans the input file for the next $JOB command record.
#INPUT FAILURE	Either a hardware problem prevented BATCH from reading the next record of the input file, or BATCH read the last record of the input file without encountering a $END command record. If a hardware problem exists, correct the problem and type any character at the Teletype to resume processing.
#SYS ERROR	A hardware problem prevented BATCH from performing an I/O operation. Program execution halts, and the system must be restarted manually. This message often indicates that the system device is not write-enabled.
INSUFFICIENT CORE FOR BATCH RUN	OS/8 BATCH requires 12K of core to run. Control returns to the OS/8 Monitor.
BATCH.SV NOT FOUND ON SYS:	A copy of BATCH.SV must exist on the system device. Control returns to the OS/8 Monitor.

(continued on next page)

Table 6-3 (Cont.)
BATCH Error Messages

Message	Meaning
WRONG OS/8 MONITOR	OS/8 BATCH requires an OS/8 Monitor no older than version 3.
DEV NOT IMPLEMENTED	BATCH cannot accept input from the specified input device because its handler is not permanently resident (SYS: or coresident with SYS:). Control returns to the Command Decoder.
ILLEGAL SPOOL DEVICE	The device specified as a spooling output device must be file structured. Control returns to the Command Decoder.

6.6 RUNNING BATCH FROM PUNCHED CARDS

The carriage return and ALTMODE characters are not defined in the punched card character set. BATCH allows you to omit terminating carriage return characters from punched card input files. Thus, when BATCH reads a punched card input file, it appends a carriage return to the content of each card, immediately following the last character on the card that is not a space character. As with disk, DECtape or paper tape input files, BATCH considers the dollar sign character to be equivalent to an ALTMODE when it appears on a punched card in any column except the first.

When you run BATCH with a punched card input file, it is possible to embed your input files in the BATCH input file. You should insert your input files into the BATCH input file in such a way that BATCH will never attempt to read a record of the files. That is, your files should follow a command record that transfers control to the program level, and the running program must exhaust all records of your file before returning to the monitor level.

Figure 6-2 illustrates how you may modify the second sample job of Figure 6-1 to run from a punched card input file with an embedded user file. In this example, PAL8 reads the punched card user file and assembles the source program, then returns control to the monitor level. BATCH reads the next card of the input file, which should contain the .R CREF command. If PAL8 has not read every record of the user input file, however, BATCH will encounter a record from this file rather than the Keyboard Monitor command record. This results in the message:

```
#RAD LINE. JOB ABORTED
```

and causes BATCH to scan the input file for the next $JOB record.

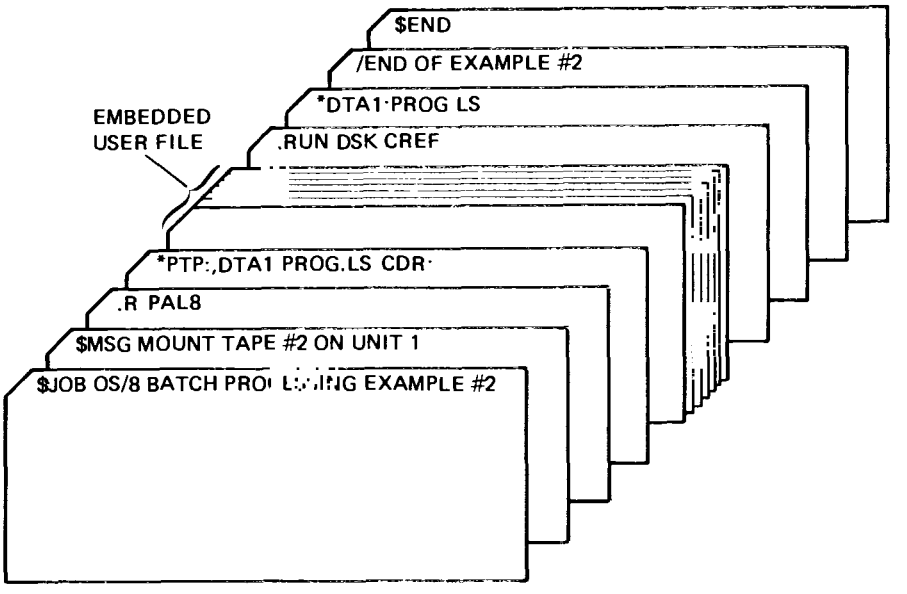


Figure 6-2 Punched Card Input File

6.7 RESTRICTIONS UNDER OS/8 BATCH

OS/8 BATCH is a "friendly" system; that is, one that is largely unprotected from user errors. The BATCH monitor resides in locations 5000 to 7577 in the highest memory field available. BATCH also uses the following locations in field 0 and the memory field in which it resides:

Location	Used as:
07777	Batch processing flag
N7774-N7777	Internal pointers

Both the Keyboard Monitor and the Command Decoder check the batch processing flag whenever they are entered from the program level. Any user program that modifies location 07777 may cause batch processing to be terminated prematurely before the next record of the BATCH input file is read.

When the Keyboard Monitor is entered from the program level (effective JMP to 07600 or 07605), it checks the batch processing flag and reads a new copy of the BATCH monitor into core if batch processing is in progress. The Command Decoder, however, does NOT perform this operation. Thus, the Command Decoder must not be called unless the BATCH monitor is already in core.

This means that you may load large programs over the BATCH monitor as long as they do not modify the last four locations in the highest memory field; however, once a user core load has overwritten the BATCH monitor, execution must remain at the program level until the Keyboard Monitor has been reentered and a new copy of the BATCH monitor is read into core. You must not call the Command Decoder after you have loaded your program over the BATCH monitor.

In general, this restriction applies only to loader programs and only when the loader calls the Command Decoder more than once while building a large core load. You can avoid multiple calls to the Command Decoder when loading large programs during batch processing if you first build the core load in a stand-alone environment and then save it for subsequent execution under BATCH.

In conjunction with this, note that it is impossible to save the core image of any program that overlays the BATCH monitor under BATCH. After the load operation but before the save is executed, the BATCH monitor is read back into core, destroying part of the user program: Thus, the Keyboard Monitor SAVE operation causes part of the BATCH monitor to be saved instead of that part of the user program that originally overlayed the BATCH monitor.

6.8 BATCH DEMONSTRATION PROGRAM

The following listing was produced by running BATCH on a 12K PDP-8/E system containing a disk, DECtape and a line printer. Only the Teletype output is reproduced here, and page breaks were inserted arbitrarily to divide the listing into convenient segments. The same BATCH input file has been processed twice, with two different system configurations.

Notice that the first BATCH processing run begins by listing the BATCH input file, and that the three demonstration programs are listed shortly thereafter.

```
.R BATCH
*SYS:DEMO/U
$JOB OS/8 BATCH DEMO
$MSG BEGIN BY LISTING BATCH INPUT FILE ON TERMINAL:
.R PIP
*TTY:<DEMO.BI
.DATA 3/5/74
$MSG SYSTEM DEVICE ASSIGNED LOGICAL NAME "IN"
.ASSIGN SYS IN
$MSG MOUNT SCRATCH DECTAPE ON UNIT 1
.R PIP
/ZERO DECTAPE DIRECTORY
$MSG WRITE ENABLE UNIT 1
*DTA1:</Z
/LIST SYSTEM DIRECTORY ON LINE PRINTER
*LPT:<IN:/E
/TRANSFER DEMO PROGRAMS TO DECTAPE
*DTA1:DEMO1.PA<DEMO1.PA
*DTA1:DEMO2.FT<DEMO2.FT
*DTA1:DEMO3.FT<DEMO3.FT
/LIST THE FIRST DEMO PROGRAM
*TTY: IN:DEMO1.PA/T
/LIST THE SECOND DEMO PROGRAM
*TTY: IN:DEMO2.FT/T
/LIST THE THIRD DEMO PROGRAM
*TTY:<IN:DEMO3.FT/T
/ASSEMBLE DEMO1.PA
.R PAL8
*IN:DEMO1.BN,DEMO1.LS<IN:DEMO1.PA
/PRINT CROSS REFERENCE LISTING
.R CREF
*LPT: IN:DEMO1.LS
/LOAD ASSEMBLED BINARY INTO CORE
```

```
.R ABLSDR
*DEMO1.BN$
/RUN FIRST DEMO PROGRAM
.START 200
/NOW SAVE CORE IMAGE OF DEMO1.PA, BUT MUST
/RELOAD FIRST, SINCE DEMO1 IS SELF-MODIFYING
.R ABSLDR
*IN:DEMO1.BN$
.SAVE SYS DEMO1 0,200
/RUN DEMO1.SV TO BE SURE THAT IT WAS SAVED CORRECTLY
.RUN SYS DEMO1
/NOW COMPILE FORTRAN MAINLINE PROGRAM
.R FORT
*IN:DEMO2.BN,LPT:<IN:DEMO2.FT$
/COMPILE FORTRAN FUNCTION ROUTINE
.R FORT
*IN:DEMO3.BN,LPT:<IN:DEMO3.FT$
/TRANSFER BOTH BINARY FILES TO DECTAPE
.R PIP
*DTA1:DEMO2.BN<DEMO2.BN/B
*DTA1:DEMO3.BN<DEMO3.BN/B
/LOAD AND EXECUTE FORTRAN PACKAGE
.R LOADER
*DEMO2.BN,DEMO3.BN/G
/RENAME DEMO3.BN FOR FUTURE REFERENCE
.R PIP
*FACT<DEMO3.BN/I
*DEMO3.BN<D
/ADD FORTRAN FUNCTION TO FORTRAN LIBRARY
.R LIBSET
*LIB8.BN/S
*FACT$
/FINALLY, DELETE TEMPORARY FILE "FACT"
.R PIP
*FACT</D
/NOW CLEAN UP DISK AREA
*DEMO1.BN,DEMO1.SV,DEMO2.BN<D
$MSG DEVICE NAMES DEASSIGNED
.DEASSIGN
$END



$MSG SYSTEM DEVICE ASSIGNED LOGICAL NAME "IN"

$MSG MOUNT SCRATCH DECTAPE ON UNIT 1

$MSG WRITE ENABLE UNIT 1

*10
IR1,    300
*200
START,  CLA CLL
        TLS
        TAD I IR1
        JMS TYPE
        JMS TEST
        JMP .-4
```

```
TYPE,   0
        TSF
        JMP .-1
        TLS
        CLA
        JMP I TYPE
TEST,   0
        TAD IR1
        TAD M335
        SZA CLA
        JMP I TEST
        TSF
        JMP .-1
        JMP 7600
M335,   -335
*301
        215;212;212;241;241;241;305;330;305;303
        325;324;311;317;316;240;303;317;315;320
        314;305;324;305;241;241;241;215;212;212
$
C       FORTRAN DEMONSTRATION PROGRAM
        DIMENSION A(35)
        DO 10 N=2,34,2
        A(N)=FACT(N)
10      WRITE (1,60)N,A(N)
        STOP
60      FORMAT (13, '! = ',E14.7)
        END



C       FORTRAN FUNCTION TO COMPUTE FACTORIALS
        FUNCTION FACT(N)
        IF (N-34) 1,5,5
1       IF (N) 2,4,2
2       M=N-2
        FACT=N
        DO 3 K=1,M
        C=N-K
3       FACT=FACT*C
        RETURN
4       FACT=1.
        RETURN
5       WRITE (1,6) N
        FACT=0.
        RETURN
6       FORMAT (15,'! EXCEEDS CAPACITY OF PROGRAM.')
        END


'''EXECUTION COMPLETE'''


!!!EXECUTION COMPLETE!!!

  2! =  0.2000000E+01
  4! =  0.2400000E+02
  6! =  0.7200000E+03
  8! =  0.4032000E+05
 10! =  0.3628800E+07
 12! =  0.4790016E+09
 14! =  0.8717829E+11
 16! =  0.2092279E+14
 18! =  0.6402374E+16
```

```
 20! =  0.2432902E+19
 22! =  0.1124001E+22
 24! =  0.6204484E+24
 26! =  0.4032915E+27
 28! =  0.3048883E+30
 30! =  0.2652529E+33
 32! =  0.2631308E+36
  34! EXCEEDS CAPACITY OF PROGRAM.
 34! =  0.0000000E+00
$MSG DEVICE NAMES DEASSIGNED


 END BATCH
```

The next run is initiated via the SUBMIT command.

```
.SUBMIT SYS:<SYS:DEMO/U/T
$JOB OS/8 BATCH DEMO


$MSG BEGIN BY LISTING BATCH INPUT FILE ON TELETYPE:
.R PIP
*TTY:<DEMO.BI

 SPOOL TO FILE BTCHA1
.DATE 8/3/72
$MSG SYSTEM DEVICE ASSIGNED LOGICAL NAME "IN"
.ASSIGN SYS IN
$MSG MOUNT SCRATCH DECTAPE ON UNIT 1
.R PIP
/ZERO DECTAPE DIRECTORY
$MSG WRITE ENABLE UNIT 1
*DTA1:</Z
/LIST SYSTEM DIRECTORY ON LINE PRINTER
*LPT:<IN:/E

 SPOOL TO FILE BTCHA2
/TRANSFER DEMO PROGRAMS TO DECTAPE
*DTA1:DEMO1.PA<DEMO1.PA
*DTA1:DEMO2.FT<DEMO2.FT
*DTA1:DEMO3.FT<DEMO3.FT
/LIST FIRST DEMO PROGRAM
*TTY:<IN:DEMO1.PA/T

 SPOOL TO FILE BTCHA3
/LIST SECOND DEMO PROGRAM
*TTY:<IN:DEMO2.FT/T

 SPOOL TO FILE BTCHA4
/LIST THIRD DEMO PROGRAM
*TTY:<IN:DEMO3.FT/T

 SPOOL TO FILE BTCHA5
/ASSEMBLE DEMO1.PA
.R PAL8
*IN:DEMO1.BN,DEMO1.LS  IN:DEMO1.PA
/PRINT CROSS REFERENCE LISTING
.R CREF
*LPT:<IN:DEMO1.LS
```

```
*SPOOL TO FILE BTCHA6
/LOAD ASSEMBLED BINARY INTO CORE
.R ABSLDR
*DEMO1.BN$
/RUN FIRST DEMO PROGRAM
.START 200


!!!EXECUTION COMPLETE!!!
/NOW SAVE CORE IMAGE OF DEMO1.PA, BUT MUST
/RELOAD FIRST, SINCE DEMO1 IS SELF-MODIFYING
.R ABSLDR
*IN:DEMO1.BN$
.SAVE SYS DEMO1 0,200
/RUN DEMO1.SV TO ENSURE THAT IT WAS SAVED CORRECTLY
.RUN SYS DEMO1


!!!EXECUTION COMPLETE!!!
/NOW COMPILE FORTRAN MAINLINE PROGRAM
.R FORT
*IN:DEMO2.BN,LPT:  IN:DEMO2.FT$
 SPOOL TO FILE BTCHA7
/COMPILE FORTRAN FUNCTION ROUTINE
.R FORT
*IN:DEMO3.RN,LPT:  IN:DEMO3.FT$

 SPOOL TO FILE BTCHA8
/TRANSFER BOTH BINARY FILES TO DECTAPE
.R PIP
*DTA1:DEMO2.BN<DEMO2.BN/B
*DTA1:DEMO3.BN<DEMO3.BN/B
/LOAD AND EXECUTE FORTRAN PACKAGE
.R LOADER
*DEMO2.BN,DEMO3.BN/G

  2! =  0.2000000E+01
  4! =  0.2400000E+02
  6! =  0.7200000E+03
  8! =  0.4032000E+05
 10! =  0.3628800E+07
 12! =  0.4790016E+09
 14! =  0.8717829E+11
 16! =  0.2092279E+14
 18! =  0.6402374E+16
 20! =  0.2432902E+19
 22! =  0.1124001E+22
 24! =  0.6204484E+24
 26! =  0.4032915E+27
 28! =  0.3048883E+30
 30! =  0.2652529E+33
 32! =  0.2631308E+36
   34! EXCEEDS CAPACITY OF PROGRAM.
 34! =  0.0000000E+00
/RENAME DEMO3.BN FOR FUTURE REFERENCE
.R PIP
*FACT<DEMO3.BN/I
*DEMO3.BN<0/D
/ADD FORTRAN FUNCTION TO FORTRAN LIBRARY
.R LIBSET
*LIB8.BN/S
*FACT$
/FINALLY, DELETE TEMPORARY FILE "FACT"
```

```
.R PIP
*FACT</D
/NOW CLEAN UP DISK AREA
*DEMO1.BN,DEMO1.SV,DEMO2.BN</D
$MSG DEVICE NAMES DEASSIGNED
.DEASSIGN
$END

$END BATCH
```

6.9 LOADING AND SAVING BATCH

You may load and save the paper tape binary version of OS/8 BATCH on the OS/8 system device by typing the following commands in response to the prompt signs generated by the OS/8 monitor:

```
.R ABSLDR
*PTR:(9P)$^
.SAVE SYS BATCH
```

Once the ALTMODE ($) has been entered, the system will print an uparrow and pause. Load the binary paper tape into the high-speed reader, turn the reader on, and type any character at the keyboard to continue.

6.10 LOADING AND SAVING PROGRAMS FOR USE UNDER BATCH

A program that never uses more than 8K of core can never destroy the BATCH monitor. When you are loading this sort of program from a DECtape system, you can save considerable time through the use of the /P option.

The /P option is a new ABSLDR option designed for use under OS/8 BATCH. It causes the 400 bit of the job status word (location 07746) to be set and prevents the Keyboard Monitor from reading a fresh version of the BATCH monitor into core every time the monitor level is reentered from the program level.

For example, OS/8 PIP never uses more than 8K of core. Thus, the best method of loading PIP would be:

```
.R ABSLDR
*PTR:(89P)=13000$
.
```

The /P option is not really necessary on a disk system, because very little time is required to refresh the BATCH monitor from disk. You should not use the /P option with any program that occupies or modifies core above field 1.

6.11 TRANSFERRING THE SYSTEM SOFTWARE FROM CASSETTE TO THE SYSTEM DEVICE

The following BATCH file can be used to transfer the OS/8 System Software from cassette to the system device.

```
$JOB JOB TO LOAD SYSTEM CASSETTE #2 TO SYSTEM DEVICE
.R MCPIP
*SYS:CCL.SV=CSA0:CCL.SV
*SYS:DIRECT.SV=CSA0:DIRECT.SV
*SYS:FOTP.SV=CSA0:FOTP.SV
*SYS:PIP.SV=CSA0:PIP.SV
*SYS:LIB8.RL=CSA0:LIB8.RL
*SYS:EDIT.SV=CSA0:EDIT.SV
*SYS:PAL8.SV=CSA0:PAL8.SV
*SYS:CREF.SV=CSA0:CREF.SV
*SYS:BITMAP.SV=CSA0:BITMAP.SV
*SYS:BOOT.SV=CSA0:BOOT.SV
*SYS:CAMP.SV=CSA0:CAMP.SV
*SYS:RK8FMT.SV=CSA0:RK8FMT.SV
*SYS:RKEFMT.SV=CSA0:RKEFMT.SV
$END
```

```
$JOB JOB TO LOAD SYSTEM CASSETTE #3 TO SYSTEM DEVICE
.R MCPIP
*SYS:FORT.SV=CSA1:FORT.SV
*SYS:SABR.SV=CSA1:SABR.SV
*SYS:LOADER.SV=CSA1:LOADER.SV
*SYS:SRCCOM.SV=CSA1:SRCCOM.SV
*SYS:EPIC.SV=CSA1:EPIC.SV
*SYS:PIP10.SV=CSA1:PIP10.SV
*SYS:RESORC.SV=CSA1:RESORC.SV
*SYS:DTCOPY.SV=CSA1:DTCOPY.SV
*SYS:TDCOPY.SV=CSA1:TDCOPY.SV
*SYS:TDFRMT.SV=CSA1:TDFRMT.SV
*SYS:DTFRMT.SV=CSA1:DTFRMT.SV
$END
```

```
$JOB JOB TO LOAD SYSTEM #4 TO SYSTEM DEVICE
.R MCPIP
*SYS:TC08SY.BN=CSA0:TC08SY.BN
*SYS:TD8ESY.BN=CSA0:TD8ESY.BN
*SYS:LINCSY.BN=CSA0:LINCSY.BN
*SYS:DF32SY.BN=CSA0:DF32SY.BN
*SYS:RF08SY.BN=CSA0:RF08SY.BN
*SYS:RK08SY.BN=CSA0:RK08SY.BN
*SYS:RK8ESY.BN=CSA0:RK8ESY.BN
*SYS:ROMMSY.BN=CSA0:ROMMSY.BN
*SYS:LINCNS.BN=CSA0:LINCNS.BN
*SYS:TC08NS.BN=CSA0:TC08NS.BN
*SYS:RK8ENS.BN=CSA0:RK8ENS.BN
*SYS:PT8E.BN=CSA0:PT8E.BN
*SYS:LSPT.BN=CSA0:LSPT.BN
*SYS:L645.BN=CSA0:L645.BN
*SYS:ASR33.BN=CSA0:ASR33.BN
*SYS:RK08NS.BN=CSA0:RK08NS.BN
```

```
*SYS:CR8E.BN<CSA0:CR8E.BN
*SYS:BAT.BN<CSA0:BAT.BN
*SYS:TD8EA.BN<CSA0:TD8EA.BN
*SYS:TD8EB.BN<CSA0:TD8EB.BN
*SYS:TD8EC.BN<CSA0:TD8EC.BN
*SYS:TD8ED.BN<CSA0:TD8ED.BN
*SYS:VR12.BN<CSA0:VR12.BN
*SYS:RF08NS.BN<CSA0:RF08NS.BN
*SYS:DF32NS.BN<CSA0:DF32NS.BN
*SYS:KL8E.BN<CSA0:KL8E.BN
*SYS:LPSV.BN<CSA0:LPSV.BN
*SYS:TM8E.BN<CSA0:TM8E.BN
*SYS:CSA.BN<CSA0:CSA.BN
*SYS:CSB.BN<CSA0:CSB.BN
*SYS:CSC.BN<CSA0:CSC.BN
*SYS:CSD.BN<CSA0:CSD.BN
*SYS:DIRECT.HL<CSA0:DIRECT.HL
*SYS:BATCH.HL<CSA0:BATCH.HL
*SYS:SABR.HL<CSA0:SABR.HL
*SYS:PIP.HL<CSA0:PIP.HL
*SYS:FOTP.HL<CSA0:FOTP.HL
*SYS:ABSLDR.HL<CSA0:ABSLDR.HL
*SYS:PIP10.HL<CSA0:PIP10.HL
*SYS:BOOT.HL<CSA0:BOOT.HL
*SYS:LOADER.HL<CSA0:LOADER.HL
*SYS:BITMAP.HL<CSA0:BITMAP.HL
*SYS:EDIT.HL<CSA0:EDIT.HL
*SYS:CREF.HL<CSA0:CREF.HL
*SYS:BUILD.HL<CSA0:BUILD.HL
*SYS:PAL8.HL<CSA0:PAL8.HL
*SYS:ODT.HL<CSA0:ODT.HL
*SYS:SRCCOM.HL<CSA0:SRCCOM.HL
*SYS:CCL.HL<CSA0:CCL.HL
*SYS:TECO.HL<CSA0:TECO.HL
*SYS:FORT.HL<CSA0:FORT.HL
*SYS:LOAD.HL<CSA0:LO.HL
*SYS:LIBRA.HL<CSA0:LIBRA.HL
*SYS:EPIC.HL<CSA0:EPIC.HL
$END



$JOB JOB TO LOAD SYSTEM CASSETTE #5 TO SYSTEM DEVICE
.R MCPIP
*SYS:LIB8.RL<CSA1:LIB8.RL
*SYS:GENIOX.RL<CSA1:GENIOX.RL
*SYS:IOH.RL<CSA1:IOH.RL
*SYS:FLOAT.RL<CSA1:FLOAT.RL
*SYS:INTEGR.RL<CSA1:INTEGR.RL
*SYS:UTILTY.RL<CSA1:UTILTY.RL
*SYS:POWERS.RL<CSA1:POWERS.RL
*SYS:IPOWRS.RL<CSA1:IPOWRS.RL
*SYS:SQRT.RL<CSA1:SQRT.RL
*SYS:TRIG.RL<CSA1:TRIG.RL
*SYS:ATAN.RL<CSA1:ATAN.RL
*SYS:RWTAPE.RL<CSA1:RWTAPE.RL
*SYS:IOPEN.RL<CSA1:IOPEN.RL
*SYS:LIBSET.SV<CSA1:LIBSET.SV
*SYS:KL8E.PA<CSA1:KL8E.PA
$END
```

```
$JOB JOB TO LOAD SYSTEM CASSETTE #6 TO SYSTEM DEVICE
.R MCPIP
*SYS:CCL.PA<CSA0:CCL.PA
$END
```

```
$JOB JOB TO LOAD OS/8 EXTENSION CASSETTE TO SYSTEM DEVICE
.R MCPIP
*SYS:BATCH.SV<CSA1:BATCH.SV
*SYS:BASIC.SV<CSA1:BASIC.SV
*SYS:BCOMP.SV<CSA1:BCOMP.SV
*SYS:BLOAD.SV<CSA1:BLOAD.SV
*SYS:BRTS.SV<CSA1:BRTS.SV
*SYS:BASIC.AF<CSA1:BASIC.AF
*SYS:BASIC.SF<CSA1:BASIC.SF
*SYS:BASIC.FF<CSA1:BASIC.FF
*SYS:BASIC.UF<CSA1:BASIC.UF
*SYS:EAEOVR.BN<CSA1:EAEOVR.BN
*SYS:RESEQ.BA<CSA1:RESEQ.BA
*SYS:TECO.SV<CSA1:TECO.SV
*SYS:MSBAT.SV<CSA1:MSBAT.SV
*SYS:GENIOX.RL<CSA1:GENIOX.RL
$END
```

6.12 RUNNING FORTRAN IV UNDER BATCH IN 32K

To run FORTRAN IV V3D under BATCH V3D causes a problem because BATCH allows the use of 32K words of memory, but the FRTS loader restricts memory to 28K. If you do not have a TD83 ROM and would like to run FORTRAN IV under BATCH in 32K, install the following patch to FRTS.SV.

```
.GET SYS FRTS
.ODT
12713/  5326  7000
^C
.SAVE  SYS  FRTS
```

CHAPTER 7

BITMAP

The BITMAP program constructs a table (map) showing the memory locations used by given binary files.

BITMAP uses 8K of core to map programs using up to 16K of core, but it requires 12K of core to map programs using more than 16K of core.

7.1 FILE AND DEVICE SPECIFICATIONS

To call BITMAP from the system device, type

```
.R BITMAP
```

The system responds by printing an asterisk (*) in the left margin. Type the input line to BITMAP, specifying input devices and file name (if input is from a mass storage device), any options desired, and an output device and file name (if output is to a mass storage device).

The standard input devices for BITMAP are PTR, DTAn, DSK, and SYS. Any other device can serve as an input device if a device handler exists in the system. Do not use TTY because the binary code may appear to the TTY handler as control characters.

BITMAP accepts only absolute binary files; you may not use relocatable and core image files. If you do not type an extension to the input file name, BITMAP defaults to the .BN extension. If more than one program is present in a file, only the first program is bitmapped. (This feature allows BITMAP to ignore any noise characters caused by reading over the end of a paper tape.) The /S switch can override this feature.

Type the RETURN key at the end of an input specification line to signal that you wish to continue to input on the next line. When there is no more input, use the ALT MODE key as a line terminator. The Command Decoder is not recalled, and control returns to the Keyboard Monitor. The last line typed specifies the output device on which the bit map is to be produced. You may specify any legal OS/8 output device, but if you don't specify one, output goes to the console terminal. For example:

```
.R BITMAP
*DTA1:FILE1,FILE2,FILE3,FILE4
*LPT:<PTR:$^
```

If an output file is specified without an extension, BITMAP inserts a .MP extension. The preceding lines cause FILE1, FILE2, FILE3, and FILE4 from DECtape 1 to be considered. Then a file is read from the high-speed paper tape reader. When you press the ALT MODE key, the $

character is printed, which indicates a return to the Keyboard Monitor. A bit map, which combines all the files read, is produced on the line printer.

The various options BITMAP accepts are listed in Table 7-1.

Table 7-1
Bitmap Options

Option	Meaning
/R	Reset internal bit map of BITMAP to look as though nothing has been input.
/S	Consider all binary programs in the specified input file(s) (instead of only the first program in each file, which is normally done).
/n	Where n is an integer, forces mapping of all files specified on this input line as if it were initially in field n.
/T	This is used to change the style of output - i.e., put teletype-style output on non-teletype or non-teletype-style output on teletypes.

Consider the following examples of command lines to BITMAP:

```
.R BITMAP
*SYS:PROG.01
*DTA1:MAP<DTA5:PATCH.BN
```

The preceding commands first create a bit map of the combined files PROG.01 (on the system device) and PATCH.BN (on DECtape 5) and then store the output in file MAP.MP on DECtape 1.

```
.R BITMAP
*LPT:<A,B,C$
```

This example combines three binary files (A, B, and C) on device DSK: to produce a bit map on the line printer.

```
.R BITMAP
*TTY:<PTR:/S$^
```

The preceding example reads a binary tape from the high-speed paper tape reader, combines all binary files on the paper tape, and produces a bit map of these files on the terminal.

7.2 BITMAP OUTPUT

BITMAP outputs a series of lines, each comprised of a string of digits. Each digit represents a single core location and has the value 0, 1, 2, or 3. The value is assigned as follows:

0 means that the location was not loaded into.

1 means that the location was loaded into once.

2 means that the location was loaded into twice.

3 means that the location was loaded into three or more times.

Occurrence of a 2 or 3 may imply a programming error (e.g., two separate routines are trying to load values into the same location).

Each line of digits represents 100(8) core locations, and lines are blocked in pairs to represent pages. On teletype output, a set of octal coordinates that associates one core location to each digit borders the bit map. Adding the horizontal and vertical coordinates that lie directly to the left and above the entry determines the corresponding core location for any given entry in the map.

7.3 BITMAP ERROR MESSAGES

After each error message, control returns to the Command Decoder. You can repeat the procedure, or reset the program (using the /R option) and repeat using different inputs.

Message	Meaning
I/O ERROR FILE #n	An I/O error occurred in input file number n.
BAD INPUT, FILE #n	A physical end-of-file has been reached before a logical end-of-file, or extraneous characters have been found in binary file n.
BAD CHECKSUM, FILE #n	File number n of the input file list had a checksum error.
NO INPUT	No binary file was found on the designated device.
ERROR ON OUTPUT DEVICE	Error occurred while writing on output device, i.e., output error on DECtape write.
NO /I	Cannot produce a bit map of an image file.

7.4 ASSEMBLY INSTRUCTIONS

Use PAL8 to make BITMAP.BN from BITMAP.PA as follows:

```
.R PAL8
*DEV:BITMAP=DEV:BITMAP
```

Use ABSLDR to make BITMAP.SV from BITMAP.BN on a DECtape file:

```
.R ABSLDR
*DEV:BITMAP=12000/9$
.SAVE DEV BITMAP
```

To load and save the binary paper tape (DEC-S8-OSYSB-A-PB15):

```
.R ABSLDR
*PTR:=12000/9$^
.SAVE DEV BITMAP
```

CHAPTER 8

BOOT

BOOT is an OS/8 program that you use to bootstrap from one PDP-8 system to another and to bootstrap from one device to another by typing commands on the keyboard. BOOT can run conveniently from OS/8 and COS 300, and can also run from any other PDP-8 monitor system (e.g., CAPS-8).

8.1 BOOTING WITH BOOT

To run BOOT from COS 300, see Chapter 9 in the COS 300 System Reference Manual (DEC-08-OCOSA-E-D).

To run BOOT from OS/8, type:

```
.R BOOT/dv
```

or

```
.RUN DEV:BOOT/dv
```

where dv is a two-character mnemonic that must immediately follow a slash. This mnemonic represents the device type and the system to be bootstrapped. Do not attempt to bootstrap onto a device that is not ready or does not exist.

To run BOOT from an OS/8 device with CCL enabled, type:

```
.BOOT/dv
```

If you use this form of call, BOOT.SV must be present on the system device.

If you type the following:

```
.R BOOT
```

the system responds with a slash, and you can enter the dv mnemonic.

If an illegal mnemonic is typed, the system prints:

```
NO
/
```

to allow you to enter a new mnemonic. Type RUBOUT to erase the line, then enter the correct command.

If a period follows the device mnemonic, the program loads the correct bootstrap into core and then halts. Press CONT to branch to the bootstrap.

Table 8-1 lists the legal mnemonics for BOOT.

Table 8-1
BOOT Mnemonics

Mnemonic	Device	System or Comments
CA	TA8E cassette	CAPS-8
DK	Any disk (RF08, DF32, RK8E, RK8)	OS/8, COS-300
DL	LINCtape	DIAL-V2, DIAL-MS
DM	RF08 or DF32	Disk Monitor
DT	Any tape (TC08, TD8E, LINCtape)	OS/8, COS 300
LT	LINCtape	OS/8, COS 300
PT	PT8E paper tape	Loads BIN/loader into field 0
RE	RK8E disk	OS/8, COS 300
RF	RF08, DF32 disks	OS/8, COS 300
RK	RK8 disk	OS/8, COS 300
TC	TC08 DECtape	OS/8, COS 300, Disk Monitor, DEC library system, and others
TD	TD8E DECtape	OS/8, COS 300
TY	TC08 DECtape unit 4	Typeset bootstrap
VE		Types BOOT's version number
ZE		Zeroes core (field 0)
RX	RX01 Diskette	OS/8

8.2 BOOT PRIORITIES

More than one type of device (e.g., disk, DECtape) may be present on the OS/8 system. When you use the DK or DT mnemonic, BOOT assumes the following priorities:

Disk	DECtape
1. RF08 or DF32	1. TC08
2. RK8E	2. TD8E
3. RK8	3. LINCtape

CHAPTER 9

BUILD

BUILD is the system generation program for OS/8 that allows you to:

- Create an OS/8 monitor system from cassettes or paper tapes.
- Maintain and update device handlers in an existing OS/8 system.
- Add device handlers supplied by DIGITAL to a new or existing system.
- Add your own device handlers to a new or existing system.

With BUILD, you use simple keyboard commands to manipulate the device handlers that make up the OS/8 peripheral configuration. BUILD allows you to quickly and easily insert devices not standard on the system.

9.1 OS/8 DEVICE HANDLERS

Each OS/8 configuration has certain device handlers available within BUILD when the system is supplied by DIGITAL. The handlers supplied with BUILD depend on the distribution media of OS/8 software, i.e., DECtape (LINCtape), cassettes, or paper tape. These device handlers are detailed for specific distribution media in Tables 9-1, 9-2, and 9-3 (see Appendix (G) for more information).

You must activate the device handlers included with BUILD before the OS/8 system can use them. The BUILD commands INSERT, REPLACE, and SYSTEM activate the device handlers. A maximum of 15 handlers, including the system device (SYS) and the default mass storage device (DSK), can be made active.

Inactive devices, although included with BUILD, cannot be used on the system until they are made active by the INSERT command. Thus, several system handlers may be supplied with BUILD, but only one may be marked active.

All other OS/8-supported device handlers are supplied with every configuration. But if they are not included in the original BUILD, you must load them into BUILD before you can use them. The BUILD command LOAD accomplishes this. See Table 9-4 for a complete list of the device handlers available with OS/8.

Two names identify handlers in BUILD. The first is the group name, assigned to an entire group of handlers of the same type. For example, the nonsystem TC08 DECtape handler supplied with a DECtape system, which has four separate internal handlers, has the group name TC.

The second name is the permanent device name. This is the name by which OS/8 identifies the physical device. For example, TC08 DECtape unit 3 has the group name TC and the permanent name DTA3.

When OS/8 software is supplied on DECtape or LINCtape, BUILD includes the device handlers shown in Table 9-1.

Table 9-1
Standard DECtape System Device Handlers

Handler	Group Name	Permanent Name(s)
TC08 DECtape system handler	TC08	SYS
TC08 nonsystem DECtape drives 0-3	TC	DTA0-DTA3
12K TD8E DECtape system handler and drives 0 and 1	TD8E	SYS, DTA0, DTA1
8K ROM TD8E DECtape system handler and drives 0 and 1	ROM	SYS, DTA0, DTA1
TD8E nonsystem DECtape drives 0 and 1	TD8A	DTA0, DTA1
TD8E nonsystem DECtape drives 2 and 3	TD8B	DTA2, DTA3
RK8E disk system handler	RK8E	SYS,RKB0
RK8E disk nonsystem handler	RK05	RKA0, RKA1, RKB0, RKB1
RK8 disk system handler	RK8	SYS, RKA1
RK8 disk nonsystem handler	RK01	RKA0, RKA1
LINCtape system handler	LINC	SYS
LINCtape nonsystem handler	LNC	LTA0-LTA3
RF08 disk system handler	RF08	SYS
Console terminal (2-page handler)	KL8E	TTY
High-speed I/O simulated on ASR-33 Teletype	KS33	PTR, PTP
High-speed reader/punch	PT8E	PTR, PTP
LP08, LS8E, LV8E line printers	LPSV	LPT
TA8E cassette drives 0 and 1	TA8A	CSA0, CSA1
PDP-12 scope	VR12	TV

The handlers supplied with a DECtape or LINCtape system are on the System Tape #2 (AL-4712C-BA). To include extra handlers in BUILD, mount this tape and use the LOAD command.

9.1.1 Cassette Systems

When OS/8 software is supplied on cassettes, the device handlers shown in Table 9-2 are included in BUILD.

Table 9-2
Standard Cassette System Device Handlers

Handler	Group Name	Permanent Name(s)
RK8E disk system handler	RK8E	SYS, RKB0
RK8 disk system handler	RK8	SYS, RKA1
RF08 disk system handler	RF08	SYS
DF32 disk system handler	DF32	SYS
Console terminal (2-page handler)	KL8E	TTY
High-speed I/O simulated on ASR-33 Teletype	KS33	PTR, PTP
High-speed reader/punch	PT8E	PTR, PTP
TA8E cassette drives 0 and 1	TA8A	CSA0, CSA1
LP08, LS8E, LV8E line printers	LPSV	LPT

These handlers are present on the system cassette AR-4588C-BA. To include extra handlers in BUILD, build an OS/8 system, use MCPIP to move specific device handlers onto the system device, then use the BUILD command LOAD. MCPIP is discussed in detail in Chapter 18.

9.1.2 Paper Tape Systems

When OS/8 software is supplied on paper tape, the device handlers shown in Table 9-3 are included in BUILD.

Table 9-3
Standard Paper Tape System Device Handlers

Handler	Group Name	Permanent Name(s)
RK8E disk system handler	RK8E	SYS, RKB0
RK8 disk system handler	RK8	SYS, RKA1
RF08 disk system handler	RF08	SYS
DF32 disk system handler	DF32	SYS

(continued on next page)

Table 9-3 (Cont.)
Standard Paper Tape System Device Handlers

Handler	Group Name	Permanent Name(s)
Console terminal (2-page handler)	KL8E	TTY
High-speed I/O simulated on ASR-33 Teletype	KS33	PTR, PTP
High-speed reader/punch	PT8E	PTR, PTP
TA8E cassette drives 0 and 1	TA8A	CSA0, CSA1
LP08, LS8E, LV8E line printers	LPSV	LPT

Two binary paper tapes provide other OS/8 handlers not included in BUILD: AK-4660C-BA contains the file-structured handlers; AK-4671C-BA contains character-oriented handlers. These tapes contain handlers that you can load into core using the BUILD command LOAD.

The BUILD device handler tapes are composed of separate segments, with a short length of leader/trailer code between them. (All of these handlers are in the special format described in BUILD Device Handler Format in this section.) Table 9-4 contains a list of the handlers that are included on the tapes. The handlers are listed in the order in which they appear on the tapes. The TC08 handler is the first segment on handler tape #1, and the KL8E terminal handler is the first segment on handler tape #2. The segments should be either labeled or separated for easier use.

To utilize a binary handler file, place the desired segment into the paper tape reader. Use the BUILD command LOAD to load that segment as follows:

```
$LOAD PTR[:]
^
$
```

Type a colon (:) after the device name if you loaded BUILD from an OS/8 system device. The ^ allows time to place the tape in the reader. Type any keyboard character to load the tape. When the $ reappears, the handler has been loaded into BUILD's table. Type the BUILD command PRINT to verify that the handler has been loaded.

Table 9-4
OS/8 Device Handlers

Handler	Group Name	Permanent Name(s)	File Name on DECtape, LINCtape, or Cassette
TC08 DECtape system handler	TC08	SYS, DTA0	TC08SY.BN
12K TD8E DECtape system handler	TD8E	SYS, DTA0, DTA1	TD8ESY.BN
8K ROM TD8E DECtape system handler	ROM	SYS, DTA0, DTA1	ROMMSY.BN

(continued on next page)

Table 9-4 (Cont.)
OS/8 Device Handlers

Handler	Group Name	Permanent Name(s)	File Name on DECtape, LINCtape, or Cassette
LINCtape system handler	LINC	SYS, LTA0	LINCSY.BN
RK8E disk system handler	RK8E	SYS, RKA0, RKB0	RK8ESY.BN
RK8 disk system handler	RK8	SYS, RKA0, RKA1	RK08SY.BN
RF08 disk system handler	RF08	SYS	RF08SY.BN
DF32 disk system handler	DF32	SYS	DF32SY.BN
TD8E DECtape drives 0 and 1	TD8A	DTA0, DTA1	TD8EA.BN
TD8E DECtape drives 2 and 3	TD8B	DTA2, DTA3	TD8EB.BN
TD8E DECtape drives 4 and 5	TD8C	DTA4, DTA5	TD8EC.BN
TD8E DECtape drives 6 and 7	TD8D	DTA6, DTA7	TD8ED.BN
TC08 DECtape drives 0-7	TC	DTA0-DTA7	TC08NS.BN
LINCtape drives 0-7	LNC	LTA0-LTA7	LINCNS.BN
RK8E disk nonsystem handler	RK05	RKA0-3, RKB0-3	RK8ENS.BN
RK8 disk nonsystem handler	RK01	RKA0-RKA3	RK08NS.BN
RF08 disk nonsystem handler	RF	RF, NULL	RK08NS.BN
DF32 disk nonsystem handler	DF	DF	DF32NS.BN
RX01SY disk system handler	RX8E	SYS	RX01SY.BN
RX01NS disk nonsystem handler	RX01	RXA0.RXA1	RX01NS.BN
VT50 VT-50 input handler	VT50	LST	VT50.BN
LQP line printer handler	LQP	LPT	LQP.BN
Octal block DUMP handler	DUMP	DUMP	DUMP.BN
RX78B disk nonsystem handler (for VT-78 only)	RX01	RXA2, RXA3	RX78B.BN
Console terminal (2-page handler)	KL8E	TTY	KL8E.BN
Console terminal (1-page handler)	AS33	TTY	ASR33.BN

(continued on next page)

Table 9-4 (Cont.)
OS/8 Device Handlers

Handler	Group Name	Permanent Name(s)	File Name on DECtape, LINCtape, or Cassette
High-speed I/O simulated on ASR-33 Teletype	KS33	PTR, PTP	LSPT.BN
High-speed reader/punch	PT8E	PTR, PTP	PT8E.BN
LP08, LS8E, LV8E line printers	LPSV	LPT	LPSV.BN
Anelex 645 line printer	L645	LPT	L645.BN
Card reader	CR8E	CDR	CR8E.BN
BATCH handler	BAT	BAT	BAT.BN
PDP-12 scope	VR12	TV	VR12.BN
TU10 magnetic tape drives 0-7	TM8E	MTA0-MTA7	TM8E.BN
TA8E cassette drives 0 and 1	TA8A	CSA0, CSA1	CSA.BN
TA8E cassette drives 2 and 3	TA8B	CSA2, CSA3	CSB.BN
TA8E cassette drives 4 and 5	TA8C	CSA4, CSA5	CSC.BN
TA8E cassette drives 6 and 7	TA8D	CSA6, CSA7	CSD.BN

9.2 CALLING AND USING BUILD

BUILD is distributed as both a binary paper tape or cassette and as a core image file (BUILD.SV) on the system DECtape or LINCtape. You should load and save the binary BUILD file on the system device when you build the initial system (see OS/8 System Generation Notes). To use the BUILD.SV file on the system device, type the following command in response to the dot the OS/8 Keyboard Monitor prints:

```
.RUN SYS BUILD
```

NOTE

It is important that you specify the RUN command, rather than the R command, when loading BUILD into core. This will allow the use of the SAV command without specifying SAVE arguments.

BUILD responds by printing a $, signaling that it is ready to accept commands.

BUILD uses a keyboard monitor similar to the one contained in the OS/8 system. Text is input from the terminal and interpreted by BUILD. Table 9-5 lists the special characters that are available for editing.

Table 9-5
BUILD Editing Characters

Character	Function
ALT MODE key	Terminate command; begin command execution. No carriage return/line feed is generated.
CARRIAGE RETURN	Terminate command; begin command execution. Also generate carriage return/line feed combination.
CTRL/C	Terminate command; return immediately to the OS/8 Keyboard Monitor.
CTRL/O	Terminate printing; return control to BUILD.
CTRL/U	Ignore line; the line may be typed again.
LINE FEED key	Examine contents of the command line.
RUBOUT key	Delete the last typed character from the command.

The standard characters permitted in a BUILD command line are:

A-Z, 0-9, SPACE, PERIOD, =, COMMA, COLON, HYPHEN

Typing any other character causes the error message:

SYNTAX ERROR

9.3 BUILD COMMANDS

The commands available in BUILD are:

ALTER	INSERT
BOOT	LOAD
BUILD	NAME
CORE	PRINT
CTL	QLIST
DCB	REPLACE
DELETE	SYSTEM
DSK	UNLOAD
EXAMINE	VERSION

The general format of the command string is:

$command args

where command represents a legal command from the list and args represents a file name, device, group name, or other argument associated with the command. You can type the command in full or abbreviate it to the first two characters. For example:

```
$PRINT
```

and

```
$PR
```

are the same. If you attempt to issue an illegal command, BUILD replies by printing the illegal command preceded by a ?. Thus the illegal command ERASE would appear:

```
$ERASE
?ERASE
$
```

9.3.1 The Hyphen Construction

Certain BUILD commands (DELETE, INSERT, REPLACE) allow the use of the hyphen construction to specify more than one permanent name. These permanent names must be four characters long and must differ only in the last character. You can insert permanent names that meet this restriction with the hyphen construction, so long as the last characters form a sequence of consecutive ASCII characters.

For example, if you wish to delete DECtape handlers DTA0, DTA1, DTA2, and DTA3, type:

```
$DELETE DTA0,DTA1,DTA2,DTA3
```

or you can use the hyphen construction and type:

```
$DELETE DTA0-3
```

9.3.2 PRINT

Syntax:

```
$PRINT or $PR
```

Function:

Prints detailed list of the BUILD devices tables. The following example shows five handlers.

```
RF08:     SYS
RK8E:     *SYS     *RKB0
KL8E:     *TTY
PTR:      PTR      *PTP
LPSV:     LPT
```

Group names are printed first in each line, followed by a colon. Following the group name is the list of permanent names available with each group. If one of the permanent names in a group is SYS, then this handler can be a system handler. An OS/8 system must have just one system handler. Some system handlers have other coresident handlers.

Any handler that is active is marked with an asterisk to the left of its permanent name (RKB0, TTY, PTP in the printout), and the devices will be included in the new OS/8 system (i.e., these handlers were inserted with the INSERT, SYS, or REPLACE commands. Other commands are available for removing, loading, and deactivating handlers). The preceding printout indicates that RK8E is the system device. The handler RK8E:RKB0 is also marked as being active.

After printing the list of available handlers, the PRINT command might also print some additional information. If, for example, you specified RK8E:RKB0 with the DSK command, the following is printed:

```
DSK=RK8E:RKB0
```

If you specified the core command to restrict the core to 12K, the message:

```
CORE=2
```

is printed, indicating that field 2 is to be the highest core field available to the OS/8 system.

9.3.3 QLIST

Syntax:

```
$QLIST or $QL
```

Function:

List the active permanent names on the system. No * is printed and the system device is the only group name printed. For example:

```
$QLIST
PTR DTA3 RK08:SYS LPT DTA4
```

9.3.4 LOAD

Syntax:

```
$LOAD activename or $LOAD dev:filename
```

Function:

Use LOAD to load a new device handler into BUILD. This handler can be one supplied by DIGITAL or one you have written. See the OS/8 Software Support Manual (DEC-S8-OSSMB-A-D) for instructions on writing device handlers. This handler is input into BUILD as a binary file or image.

If you are running BUILD stand-alone, e.g., to create an initial OS/8 system, the LOAD command has the form:

```
$LOAD activename
```

where activename is the permanent name of an input device handler that the INSERT, REPLACE, or SYSTEM command has made active. It must be a handler for a non-file structured device. For example, to load a new handler from a binary paper tape with the PTR handler already in BUILD, type:

```
$LOAD PTR
```

If you are running BUILD under control of OS/8, the LOAD command has the form:

```
$LOAD dev:filename
```

where dev is an input device handler that exists in the current OS/8 system. (These are not the same as the handlers that BUILD marks active.) If no dev: is specified, DSK: is assumed.

If dev: is non-file structured (i.e., paper tape), you may omit the filename. The filename has the form:

```
name.extension
```

Filename is the binary file of the new handler to be loaded. The default extension is .BN. If you use no extension, you may omit the dot (.).

Example:

```
$LOAD DTA3:HANDLR.03      Load a file named HANDLR, with an
                          extension of 03, from DTA3.
```

You may specify several files that you are loading on one line. Separate the files by commas. You must specify a device for each file, or DSK will be assumed. If multiple files are specified, each file must contain a separate handler to be loaded. For example:

```
$LOAD DTA3:FILE1,DTA5:FILE2
```

Once you have successfully issued the LOAD command, the new device handlers become available for further manipulation. The new handlers will appear in the PRINT output, but will not be marked as active.

9.3.5 INSERT

Syntax:

```
$INSERT gname,pname
```

Function:

After a LOAD command has made a handler or group of handlers available for insertion into the OS/8 system, use the INSERT command to make particular entry points active. The INSERT command uses two arguments; gname and pname. Gname is the group name of the handler, for example, the gname for TC08 DECtape is TC. Pname is the permanent name by which the device is currently known to BUILD. See Table 9-4 for a complete list of permanent device names. TC08 DECtape thus has the group name TC and the permanent names DTA0-DTA7.

Examples:

```
$IN KL8E,TTY
$IN TC08,SYS
```

If you specified no permanent name (and no :), the first name in the device group is assumed. For example:

```
$INSERT TC
```

would assign DTA0 as the permanent name.

You can insert several handlers in the same group into the same command by separating the permanent names with commas. For example:

```
$IN TC,DTA0,DTA3,DTA7
```

If several permanent names (each four characters long) differ only in the last character, you can insert them simultaneously with the hyphen construction so long as the last characters form a sequence of consecutive ASCII characters.

Example:

```
$INSERT TC,DTA2-5
```

is the same as

```
$INSERT TC,DTA2,DTA3,DTA4,DTA5
```

and

```
$INSERT RK01,RKA0-2
```

is the same as

```
$INSERT RK01,RKA0,RKA1,RKA2
```

If the permanent name specified is not part of the group name specified, or if the group name does not exist, the following message is printed:

```
name NOT FOUND
```

If disk is the device you are inserting, you can follow the group name with a construction of the form:

```
pname=n
```

Where n is a digit in the range 1 to 7 and represents the number of platters available. Use this option for the RF08 and DF32 disks. For example:

```
$IN RF,RF=2
```

If you specify no such option, =1 is assumed. If n is too large for the device specified, the following message is printed:

```
?PLAT
```

9.3.6 DELETE

Syntax:

```
$DELETE aname
```

Function:

DELETE takes a device that is currently marked as active and makes it inactive. (Active devices are marked with an * in the PRINT command output and are printed by the QLIST command.)

The argument for DELETE is the permanent name of the device. You can obtain the current permanent name from the PRINT or QLIST output. The major function of DELETE is to make device slots available to BUILD.

For example, assume that the QLIST command output is:

```
DTA0 DTA1 RK8E:SYS RKB0 TTY LPT CSA0 CSA1 CSA2 CSA3
```

If the following command is issued to BUILD:

```
$DELETE CSA0,CSA1,CSA2,CSA3
```

CSA0, CSA1, CSA2, and CSA3 will no longer be permanent devices, and the slots used by the TA8A and TA8B device groups will be made available to BUILD. The QLIST output after the above command will be:

```
DTA0 DTA1 RK8E:SYS RKB0 TTY LPT
```

Note, as previously explained, that you can use the hyphen construction in DELETE to remove a sequence of devices. Therefore, you can type the command to make the cassette handlers inactive as follows:

```
$DELETE CSA0-3
```

9.3.7 REPLACE

Syntax:

```
$REPLACE pname=gname, pname1
```

Function:

REPLACE combines the functions of DELETE and INSERT to delete one device and activate another in a single step. The arguments for REPLACE are:

pname	The permanent name of the device to be deleted. (Same as the argument of the DELETE command.)
gname, pname2	The group name and permanent name of the particular device to be inserted into the system (see INSERT for more details).

Example:

Assume the PRINT output is:

```
PTR8E:   *PTP   *PTR
CR8E:    *CDR
RK05:     RKA0   RKB0   RKA1    RKB1
```

Use REPLACE to delete the card reader (CDR) and to insert the RK05 group handler for RKA0:

```
$REPLACE CDR=RK05,RKA0
```

The output of PRINT after this REPLACE is:

```
PTR8E:   *PTP   *PTR
CR8E:     CDR
RK05:    *RKA0   RKB0   RKA1    RKB1
```

Use the hyphen construction with REPLACE to delete and insert more than one device handler. For example, assume you are replacing the LINCtape handlers LTA0, LTA1, LTA2, and LTA5 with DECtape handlers DTA0, DTA1, DTA2, and DTA5. The following command accomplishes this:

```
$REPLACE LTA0-2,LTA5=TC,DTA0-2,DTA5
```

9.3.8 UNLOAD

Syntax:

```
$UNLOAD gname, or $UNLOAD gname, pname
```

Function:

Use UNLOAD to delete a handler group (gname) or a permanent name (pname) from the BUILD system. (This differs from DELETE, which does not physically eliminate a device.) Use UNLOAD primarily when the NO ROOM error occurs during a LOAD command.

For example, assume you are removing the entire group of LINCtape handlers. Type the command:

```
$UNLOAD LNC
```

This command unloads the LINCtape handler LNC and all permanent names (LTA0, LTA1, LTA2, LTA3, etc.) associated with it.

To remove a particular permanent name from BUILD, e.g., DTA3, type:

```
$UNLOAD TC:DTA3
```

This command unloads only the entry point name.

To remove several permanent names, but not the entire group, use the UNLOAD command, with commas separating the permanent names. For example:

```
$UNLOAD TC:DTA0,DTA2
```

You cannot use the hyphen construction with the UNLOAD command.

9.3.9 NAME

Syntax:

```
$NAME pname=pname1
```

Function:

The NAME command allows you to alter the device name that will be used by OS/8. The first argument, pname, must be the current name of a device marked active in the PRINT output. Pname2 is the name you wish to call this device. You may use only 4-character device names in the NAME command. If you enter longer names, all characters beyond the first four are ignored. After you use the NAME command, pname2 becomes the current permanent name; pname is unknown to BUILD.

Example:

Assume that the PRINT output is:

```
TC    :   *DTA0   *DTA1    DTA2     DTA3
RK8E  :   *SYS    *RK0
KL8E  :   *TTY
PT8E  :   *PTP    *PTR
```

To change the paper tape reader so that it is recognized by the permanent name READ, use the following command:

```
$NAME PTR=READ
```

The output from PRINT would then be:

```
TC    :   *DTA0   *DTA1    DTA2     DTA3
RK8E  :   *SYS    *RK0
KK8E  :   *TTY
PT8E  :   *PIP    *READ
```

If the permanent name specified as pname is not a currently active device, the message:

```
pname NOT FOUND
```

is printed. If this message appears, check the PRINT output to determine the correct permanent name.

9.3.10 ALTER

Syntax:

```
$ALTER gname, loc=newvalue
```

Function:

The ALTER command allows you to change locations in device handlers. The arguments are:

gname	Group name of the handler.
loc	Alter relative octal location. If the handler is a 1-page handler, loc must be an octal number in the range 0-0177. If it is a 2-page handler, loc must be an octal number in the range 0-0377.
newvalue	An octal number specifying the new contents of the location specified by loc. If you do not enter =newvalue, BUILD prints the old value of loc followed by a slash. You can then enter newvalue or type a carriage return to retain the old value.

9.3.11 EXAMINE

Syntax:

```
$EXAMINE gname, loc
```

Function:

EXAMINE allows you to examine, but not modify, a location within a device handler. See the ALTER command.

9.3.12 DSK

Syntax:

```
$DSK=gname,pname or $DSK=aname
```

Function:

Use the DSK command to specify which device you are designating as DSK, the default storage device for OS/8. If you use the first form of the command, i.e.,

```
$DSK=gname,pname
```

the gname is the group name of the device, and pname is the permanent name. For example:

```
$DSK=TC08:DTA0
```

assigns DTA0 as the device called DSK.

When you issue the DSK command, you need not enter the permanent name. However, you must enter the permanent name, via an INSERT, REPLACE, or SYSTEM command before you issue the BOOT command.

If you use the second form of the command, i.e.,

```
$DSK=aname
```

aname must be a permanent name BUILD marks as active. For example, the following command specifies the already active device RKA0 as the default device DSK:

```
$DSK=RKA0
```

If you enter no DSK command, or if you issue the command without an argument, i.e.,

```
$DSK=
```

or

```
$DSK
```

BUILD specifies SYS as DSK when you issue a BOOT command.

9.3.13 CORE

Syntax:

```
$CORE n
```

Function:

You use the CORE command to specify the highest core field available to the OS/8 system being built. The n is an octal number in the range 0 to 7. If n is 0 or omitted, or if you do not use the CORE command, the built system will use all of the available core. If n specifies more core than is available, the following message is printed:

```
?CORE
```

The value of n for the available core sizes is as follows:

Value of n	Core
0	all available core
1	8K
2	12K
3	16K
4	20K
5	24K
6	28K
7	32K

For example, a system that is to use only 24K of a 32K system requires the following CORE command:

```
$CORE 5
```

9.3.14 DCB

Syntax:

```
$DCB aname or $DCBaname=newvalue
```

Function:

The DCB command allows you to examine or modify the DCB word associated with a permanent name (see Section 9.5 for information on DCB words).

The DCB word is the first word that appears after the permanent name in a description (from the handler header information words). Aname must be the permanent name of a device currently marked as active in the PRINT output.

Example:

```
$DCB DTA4=6160
```

changes the DCB of DTA4 so that this handler becomes a read-only device. You could also type this command as:

```
$DCB DTA4
4160/6160
```

9.3.15 CTL

Syntax:

```
$CTL aname=loc
```

Function:

The CTL command allows you to modify the control word that appears after the DCB word in the handler header block. For example:

```
$CTL LTA3=24
```

changes the entry point of the LTA3 handler to relative location 24.

9.3.16 VERSION

Syntax:

```
$VERSION or $VE
```

Function:

The VERSION command prints BUILD's version number on the terminal.

9.3.17 SIZE

Syntax:

```
$SIZE aname or $SIZE aname=new value
```

aname must be the permanent name of a device currently marked as active.

Example:

```
$SIZE RF08=1777
```

changes the length of the RF08 handler to 1777.

Function:

The SIZE command modifies word ten of a handler header block. Word ten specifies the size, in blocks, of a single platter on a system device.

9.3.18 SYSTEM

Syntax:

```
$SYSTEM sname=n
```

Function:

The SYSTEM command specifies devices that are system handlers or that are coresident with system handlers. The number n reflects the number of platters included in the system device (valid only for multiple platter RF08 and DF32 disks). (Table 1-6 lists the available system

handlers and their associated values for n.) The argument sname must be one of the legal device system names. If it is not, BUILD prints:

```
?SYS
```

thereby requesting a new system specification.

Action is not taken on the SYSTEM command until you give the BOOTSTRAP command, so you may respecify a device with SYS. The system device used is the last one issued prior to the BOOT command. Specifying a new system device is not always necessary. For example, if you wish to insert new peripheral handlers, then this command is not needed. If you do not issue it, the OS/8 resident system is not affected beyond having altered device tables.

BUILD includes the SYSTEM command only so it will be compatible with older versions of BUILD. You can specify the system device with the INSERT command. For example, the command:

```
$SYS RF08=2
```

is the same as the command:

```
$INSERT RF08,SYS=2
```

If the device specified in the SYS command is not the current system device, you will have an opportunity to have a zero directory placed on your new system device. If the system device is the same as the current system device, no new directory will result.

9.3.19 BUILD

Syntax:

```
$BUILD or $BU
```

Function:

You use the BUILD command only when building an initial OS/8 system from cassettes or paper tape. When you type the BUILD command, BUILD prints:

```
LOAD OS/8:
```

to which you must respond by typing the device that contains the new OS/8 monitor, e.g.,

```
LOAD OS/8: CSA0
```

BUILD then loads and writes the various parts of OS/8 onto the system device. After writing OS/8, BUILD prints:

```
LOAD CD:
```

to which you respond with the appropriate device, or with a carriage return to specify that the device is the same as the one specified in the LOAD OS/8: message. BUILD loads the Command Decoder and writes it onto the system device.

Do not use the BUILD command at any time other than while building an initial OS/8 system. When you type this command, OS/8 assumes that you are building a new OS/8 system. It automatically zeroes the system device directory. Refer to the OS/8 System Generation Notes for instructions on building an initial system.

9.3.20 BOOTSTRAP

Syntax:

$BOOTSTRAP or $BO

Function:

BOOTSTRAP is the command that finally implements all the changes that you made with BUILD. BOOT rewrites all relevant Monitor tables and device handlers to reflect the updated system status. The devices BUILD had marked active now become device handlers in the system.

Before you type a BOOTSTRAP command, you must specify the system device with either the SYSTEM or INSERT command. If no SYS is specified, the message:

SYS NOT FOUND

is printed.

If the system device specified is different from the current system device, BUILD copies the system from the current system device to the new system device. After the copy is complete, BUILD asks:

WRITE ZERO DIRECT?

to determine whether a new (zero) directory is to be written on the new system device. If the reply is YES, the system will place a zero directory on the device. Any other reply causes the system to retain the old directory.

NOTE

Exercise care if you want the old directory retained. The directory must be one of an OS/8 system device.

After you answer this question, BUILD updates the system and prints:

```
SYS BUILT
.
```

Control returns to the Keyboard Monitor. When the BOOTSTRAP command has performed its functions and the Keyboard Monitor is once again active, save the copy of BUILD just used. This way, an image of the current system status is preserved, and you can use the saved copy of BUILD again. When it is used again, the devices initially marked active remain marked active. To save BUILD, type:

```
.SAVE SYS BUILD
```

in response to the dot printed by the Keyboard Monitor. This assumes that you originally loaded BUILD into core with a RU or RUN command.

9.4 BUILD ERROR MESSAGES

The following is a list of error messages that may appear when using BUILD. These messages usually indicate a syntax or user error.

Table 9-6
BUILD Error Messages

Message	Explanation
?BAD ARG	No device name was included in the LOAD command.
?BAD INPUT	An error was detected in the binary file; it is not a proper input for the LOAD command.
?BAD LOAD	An attempt was made to load a binary handler that is not in the correct format.
?BAD ORIGIN	The origin in a binary file is not in the range 200-577.
?CORE	A CORE command specified more memory than is physically available, or the BOOT command was issued on an 8K system with a 2-page system handler active. Two-page system handlers require at least 12K of core to be present on the OS/8 system.
?DSK	The device specified in a DSK command is not a file structured device.
?HANDLERS	More than 15 handlers, including SYS and DSK, were active when a BOOT command was issued.
I/O ERR	An error occurred while reading from an input device during a LOAD command.
?NAME	A device or file name was not designated in a command that requires one to be present.
NO ROOM	Too many device handlers were present on the system when a LOAD or BUILD command was typed. The UNLOAD command must be used to remove a handler before another can be loaded.
name NOT FOUND	The device or file name designated in the command was not found.
?PLAT	The =n in a SYS command is too large for the device specified, e.g., RF08=5.
?SLOTS	More than eight groups of non-system handlers were inserted. Each slot may have more than one entry point. To correct, delete PNAMES until there are eight or fewer nonsystem handlers.

(continued on next page)

Table 9-6 (Cont.)
BUILD Error Messages

Message	Explanation
?SYNTAX	An illegal character was typed in a BUILD command line. The line must be retyped.
?SYS	One of the following conditions exists: a. A permanent name in a SYS command was not a system handler or coresident with one. b. A BOOT command was issued when two or more system handlers were active. c. A BOOT command was issued when an active handler that must be coresident with a SYS handler did not have the system handler active.
SYS ERR	An I/O error occurred with a system handler. The computer halts. Press CONT to retry or restart the BUILD procedure from the beginning. Do not assume that a valid OS/8 system remains in core.
SYS NOT FOUND	No active handler with the name SYS was present when a BOOTSTRAP command was issued.

9.5 BUILD DEVICE HANDLER FORMAT

Use the BUILD command LOAD to load device handlers not provided by BUILD into core. They can then be inserted into the OS/8 system. The format of the input to LOAD is a binary file containing the handler as well as a header block that contains information pertaining to the devices included in that file. You should code the handler in PAL8 machine language. The structure of the source for a BUILD device handler is:

```
        *0
HEADER BLOCK
        *200
BODY OF DEVICE
    HANDLER
```

The origins at 0 and 200 are vital to BUILD. The *0 is an important part of the header block. If you omit this, you cannot load. The *200 is also necessary for loading. If the handler contains an origin outside the range 200-577, BUILD generates an error message and then aborts the load.

9.5.1 Header Block

The header block contains the following information:

Word 1: -X, where X is the number of separate handlers contained in this file. Thus a handler for TC08 has the first word equal to -10(octal).

Words 2-9: Descriptor block for the first handler in the group.

Words 10-17: Descriptor block for second handler in the group.

. .
. .
. .

Descriptor block for the last handler in the group. If the handler is a system handler, the length of the bootstrap and the bootstrap itself follows.

Thus, each handler in the group must have an eight-word block describing its characteristics. If more than twelve handlers are in a group, an error generates during the LOAD.

9.5.2 Descriptor Block

Each eight-word descriptor block contains the following information:

Words 1,2: Device type name. This name is the group name, or type, of all the handlers in this group. It is usually designated by the DEVICE pseudo-op.

Example: DEVICE RK8

Words 3,4: OS/8 device name. This is the name (permanent name) by which the particular device will be recognized in the OS/8 system. The NAME command can alter it.

Example: DEVICE RKA0

Word 5: Device Control Block. This word reflects the type of device, in accordance with Table 9-7. Bits 9-11 specify the maximum number of platters on the device (0=1).

Example: 4050

Word 6: Entry point word. This word must contain the entry point offset in bits 5-11 (see Section 9.5.4). Bit 0 should be a 1 if the handler is a two-page handler. Bit 1 should be a 1 if the entry point is SYS. Bit 2 should be a 1 if the entry point is coresident with SYS.

Example: 0020

Word 7: Must be 0.

Word 8: Must be 0, except for a system handler that uses it to specify the block length of the device as a negative number.

As an example, consider the handler for the nonsystem RK05 handlers. This file contains four separate handlers; the source code would appear as follows:

```
        *0
        -4                  /4 DEVICES

DEVICE RK05; DEVICE RKA0; 4050; 0020; ZBLOCK 2
DEVICE RK05; DEVICE RKB0; 4050; 0021; ZBLOCK 2
DEVICE RK05; DEVICE RKA1; 4050; 0022; ZBLOCK 2
DEVICE RK05; DEVICE RKB1; 4050; 0023; ZBLOCK 2

        *200

              (HANDLER BODY)
```

The device type of the group is RK05 (Words 1-2). The permanent device names are RKA0, RKB0, RKA1, RKB1. Since each device is RK05, the device control block (DCB) word for each is identical.

The entry point word indicates where the entry point for that particular device occurs relative to the top of the page. Thus, in the above example, RKA0 enters at the 20th location from the top of the page, RKB0 at the 21st, etc.

It is vital that this information is accurate. If errors exist in this data, unpredictable results occur when you generate the system.

9.5.3 Breakdown of DCB Word

The DCB word for a device provides specific information that is used in the OS/8 Monitor. Table 9-7 details its structure.

Table 9-7
DCB Word

Bit	Meaning
0	1 if file structured device
1	1 if read-only device (e.g., PTR)
2	1 if write-only device (e.g., LPT)
3-8	Device Type 00 = console terminal 01 = high-speed paper tape reader 02 = high-speed paper tape punch 03 = card reader 04 = line printer

(continued on next page)

Table 9-7 (Cont.)
DCB Word

Bit	Meaning
	Device Type
3-8 (Cont.)	05 = RK8 Disk 06 = RF08 (1 platter) 07 = RF08 (2 platter) 10 = RF08 (3 platter) 11 = RF08 (4 platter) 12 = DF32 (1 platter) 13 = DF32 (2 platter) 14 = DF32 (3 platter) 15 = DF32 (4 platter) 16 = TC08 DECtape 17 = LINCtape 20 = TM8E magnetic tape 21 = TD8E DECtape 22 = BAT - BATCH handler 23 = RK8E disk 24 = NULL - NULL handler 25 = RX01 diskette 26 = Unused 27 = TA8E cassettes 30 = PDP-12 scope 31-35 = Unused by DIGITAL 36 = Dump Handler 37 = Unused by DIGITAL 40-77 = Reserved for user-written handlers
9-11	Used only by OS/8 Monitor

Whenever you insert a device into OS/8, follow this structure to obtain correct results.

9.5.4 Entry Point Offset

Word 6 of each device descriptor block specifies the relative entry point of that particular handler. Devices supplied by DIGITAL have a fixed set of entry points, described below.

Use care when coding new device handlers for insertion into the system. The entry point offset for the new handler must not be the same for any other file structured device in the system. For example, OS/8 currently uses relative entry points 7-24 for file structured devices. No new handler should have entry points at 7 to 24 of the page. If this occurs, the system may perform incorrectly.

Current file device and entry point offsets appear below:

Device	Entry Relative to Top of Page
TC08 DECtape	10-17
TD8E DECtape	10-17
LINCtape	10-17
System device	7
RK8/RK8E disk	20-23
RF/DF disk	24
RXA0	30
RXA1	34

Thus, the user-coded file devices should use entry points other than 7-24, 30, 34.

If you add a new file structured user device to the system, alter the device-length table in PIP to permit zeroing of the device directory. To do this, use ODT as follows:

```
.GET SYS PIP
.ODT
136nn/0000 xxxx
^C                (user types CTRL/C)
.SAVE SYS PIP
```

The nn represents the two-digit device Table 9-12 indicates. The xxxx is the negative of the last block number on the device. Both nn and xxxx are octal numbers.

For example, if you assign the new device a code of 40 (currently the first unused entry), and the last OS/8 block on the device was block 1000, PIP would change as follows:

```
.GET SYS PIP
.ODT
13640/0000 7000
^C
.SAVE SYS PIP
```

9.6 CREATING A SYSTEM HANDLER

When you create a new system handler, observe the following restrictions:

- The length of a bootstrap must be greater than or equal to 21 (octal) locations. You must pad a bootstrap shorter than 21 locations, otherwise BUILD results are unpredictable.
- The length of the bootstrap must be less than or equal to 177 (octal) locations.
- If the system handler is a one-page handler, only the first 47 (octal) locations of the bootstrap are significant. The remaining locations are ignored and not written on the system device. Also, no handler may have more than 20 (octal) entry points.
- If a system handler is two pages long, relative location 12 of the first page must contain a 3. The second page loads into location 27600 and is stored on block 66 of SYS:.

CHAPTER 10

CASSETTE AND MAGNETIC TAPE POSITIONER (CAMP)

The CAMP program positions cassettes, magnetic tapes, and certain other devices. To call CAMP from the system device, type:

```
.R CAMP
```

in response to the Keyboard Monitor dot. CAMP prints a # to indicate it is ready to receive a command. You may terminate the command line you enter with a carriage return (CAMP retains control) or an ALTMODE (control returns to the Keyboard Monitor).

10.1 CAMP COMMANDS

Each CAMP command begins with a keyword consisting of two or more letters. You need not type the full CAMP command; however, each command has letters that are required. The CAMP commands appear below in alphabetical order. Letters not required are underlined.

BACKSPACE
EOF
HELP
REWIND
SKIP
UNLOAD
VERSION

10.1.1 BACKSPACE Command

The BACKSPACE command spaces a magnetic tape or cassette backward a specified number of files or records. You may also issue this command indirectly with the CCL BACKSPACE command.

The BACKSPACE command has the form:

```
                  Records
BA dev: nnnn
                  Files
```

Where "dev:" is the permanent name of a cassette or magnetic tape drive. The "nnnn" is an unsigned decimal number representing the number of records or files to backspace. This number must be in the range 0-4095. If you enter no number, nnnn=1 is assumed. A keyword beginning with either an R, indicating records, or an F, indicating files, follows this number. If neither F nor R is entered, F is assumed.

Examples:

```
*BA CSA0: 2 F
```

positions the cassette mounted on CSA0 backward two files.

```
*BA MTA1:
```

positions the magnetic tape mounted on MTA1 backward one file.

If a file mark is read before the tape has spaced over the proper number of records, the message:

```
% CAN'T - AT BOF
```

appears and the device is moved forward one record. This leaves the device positioned at the beginning of the file (just before a data record).

The file where the device is currently positioned is not counted when you make an attempt to backspace a number of files. For example, the command:

```
*BA MTA1: 3 F
```

moves the tape backward over four file marks and then moves it forward one record. The tape is then positioned at the beginning of the file. If nnnn=0, the tape backspaces to the beginning of the file where it is currently positioned.

10.1.2 EOF Command

The EOF command writes a single file mark (file gap) on the magnetic tape or cassette you have specified. You may also issue this command indirectly with the CCL EOF command.

The EOF command has the form:

```
EOF dev:
```

where "dev:" is the permanent name of a cassette or magnetic tape drive.

Example:

```
*EOF CSA1:
```

10.1.3 HELP Command

The HELP command prints a short message on the console terminal, reminding you of the CAMP command syntax. The form of this command is:

```
*HELP
```

10.1.4 REWIND Command

The REWIND command issues a rewind command to one of the following OS/8 device controllers: cassette, magnetic tape, or TC08 DECtape.

The REWIND command form is:

```
REWIND dev:
```

where "dev:" can be any OS/8 file structured device. If "dev:" is a cassette, control returns to CAMP while the cassette is rewinding: CAMP prints another #, indicating it is ready to receive another command. If "dev:" is magnetic tape or TC08 DECtape, the device rewinds immediately, and control returns to the OS/8 Keyboard Monitor while the device is rewinding. If you issue a REWIND command to any other OS/8 device (e.g., LINCtape), control returns to CAMP after the device is rewound.

Example:

```
#RE DTA1:
```

10.1.5 SKIP Command

The SKIP command advances over the number of files or records you specified on a magnetic tape. You may also issue this command indirectly with the CCL SKIP command. You do not implement the SKIP command for cassettes.

The SKIP command has the form:

```
                nnnn Records
#SKIP MTAn:          Files
                EOD
```

where MTAn: may be any magnetic tape drive, depending upon the number of magnetic tape drives on the OS/8 system. The "nnnn" is an unsigned decimal number representing the number of files or records you are advancing over. This number must be in the range 0-4095. EOD indicates that the tape is to advance to the end of data. The end of data on a magnetic tape is a point between two file marks. If the tape is already past the end of data, rewind it before you issue the EOD command. If you have specified neither "nnnn" nor EOD, nnnn=1 is assumed.

If you have specified a number, a keyword beginning with R (for records) or F (for files) may follow. If neither F nor R is entered, F is assumed.

Examples:

```
#SKIP MTA0: 2 RECORDS
```

advances the magnetic tape on MTA0: forward two records.

```
#SKIP MTA1: 6 F
```

advances the magnetic tape on MTA1: forward six files.

If a file mark is read before the tape has advanced over the proper number of records, the warning message:

```
% CAN'T - AT EOF
```

appears and the tape moves backward one record to the end of the file (just after the last data record but before the file mark). If nnnn=0, nnnn=1 is assumed when skipping records.

The file where the tape is currently positioned is counted when you attempt to advance over a number of files. Thus nnnn=1 means to advance to the beginning of the next file. If nnnn is greater than 0, the tape is positioned at the beginning of a file (just after a file mark but before any data records). If nnnn=0, the tape advances to the end of the file where currently positioned (before a file mark, but after all data records).

If you encounter the end of data before you have skipped the specified number of files, the warning message:

```
% CAN'T - AT EOD
```

appears and the tape is positioned at the end of data. If a tape is already positioned at the end of data, the SKIP command produces meaningless results.

10.1.6 UNLOAD Command

The UNLOAD command rewinds a magnetic tape controller and turns it off line. As the tape is rewinding, control returns to CAMP for another command. You will have to manually turn the magnetic tape on line for use after you have issued an UNLOAD command.

You may also use the UNLOAD command to unload TC08 and TD8E DECtapes from their reels. When used on DECtapes, the UNLOAD command rewinds the DECtape on the unit specified, selects a different unit, and returns control to CAMP for another command. This DECtape unit cannot be used until another legal command, e.g., the Keyboard Monitor ASSIGN command, is issued to the DECtape controller.

You can also use the UNLOAD command to write-lock an RK8E disk.

The UNLOAD command form is:

```
#UNLOAD dev:
```

where "dev:" may be any one of the following:

```
magnetic tape
TC08 DECtape
TD8E DECtape
RK8E disk
```

10.1.7 VERSION Command

The VERSION command prints the version number of CAMP on the terminal. This command form is:

```
#VERSION
```

10.2 CAMP ERROR MESSAGE SUMMARY

The error messages listed in Table 10-1 may appear during a CAMP operation.

Table 10-1
CAMP Error Messages

Messages	Explanation
% CAN'T - AT BOF	A file mark was read before the specified number of records were read over in a BACKSPACE command. The device is moved forward so that it is positioned at the beginning of the file.
? CAN'T - AT BOT	A BACKSPACE command cannot move the device backward the specified number of files because the device is positioned at the beginning of the first file.
% CAN'T - AT EOD	The specified number of files cannot be advanced over because the end of data was encountered. The tape is positioned at the end of data.
% CAN'T - AT EOF	A file mark was read before the specified number of records were advanced over in a SKIP command. The tape is moved backward one record to leave it positioned at the end of the file.
? CAN'T - DEVICE DOESN'T EXIST	The device specified in a CAMP command is not present on the OS/8 system.
? CAN'T - DEVICE IS READ-ONLY	The device specified in a CAMP command is a read-only device, e.g., PTR.
? CAN'T - DEVICE IS WRITE-ONLY	The device specified in a CAMP command is a write-only device, e.g., TTY.
? CAN'T FOR THIS DEVICE	The operation specified does not make sense for the device specified, e.g., REWIND LPT:.
? CAN'T I/O ERROR	An input/output error has occurred, and a brief explanation will follow.
? NUMBER TOO BIG	The "nnnn" specified in a BACKSPACE or SKIP command is greater than 4095.
? SYNTAX ERROR	An illegal character was typed in a CAMP command or a command was formatted incorrectly. The command must be retyped.

CHAPTER 11

CROSS-REFERENCE PROGRAM (CREF)

CREF aids you in writing, debugging, and maintaining assembly language programs by pinpointing all references to a particular symbol. CREF operates on output from either the PAL8, SABR, or RALF assembler.

11.1 CALLING AND USING CREF

To call CREF from the system device, type

```
.R CREF
```

in response to the Keyboard Monitor dot. This loads the Command Decoder, which replies by printing an asterisk in the left margin. Enter one output file specification and one input file specification.

NOTE

The input to CREF must be the listing pass output from either the PAL8, SABR, or RALF assembler. If this is not the case, CREF will not operate properly. RALF is not fully supported by CREF.

If you specify no output file, CREF assumes you are sending the output to the line printer. If you specify no input or output file extension, the extension .LS is assumed. If you specify no input file, control returns to the Command Decoder until an input file is specified. The CREF version number is printed at the end of the CREF table in the form Vn, where n is the current version number.

11.1.1 CREF Options

The following options are available to you. The option and the file specification are placed in the command string.

Table 11-1
CREF Options

Option Code	Meaning
/P	Disable pass 1 listing output. The output is reenabled when $ (or END for SABRE) is encountered. Thus the $ (END) and symbol table are printed if you use the /P option. Inoperable for RALF output.
/U	Disable pass 1 listing output and the symbol table. Inoperable for RALF output.
/R	Interpret input as RALF code.
/Q	Interpret input as SABR code. Signal CREF to accept special SABR characters. If you use the /Q option, the /X option is forced on.
/X	Do not process literals. For programs with too many symbols and literals for CREF, this option may create enough space for CREF to operate.
/A	Do not eliminate the file CREFLS.TM. If you do not specify the /A option, and if CREF was chained to from PAL8, the file CREFLS.TM is eliminated.
/M	Cross-reference mammoth files in two major passes. Pass 1 processes the symbols from A through LGnnnn; pass 2 processes the symbols from LHnnnn through Z and literals. This permits significantly large files to be cross-referenced. If the /M option is used, the file CREF.SV must be on the system device.

11.1.2 Examples of CREF Usage

Examples of calling and using CREF are given below.

Example 1:

```
.R CREF
*PTEMP
```

The Command Decoder prints an *, CREF assigns LPT: as the output device. The input file is PTEMP, assumed to be on device SYS, with the extension .LS. If you do not find the file SYS:PTEMP.LS, a search for SYS:PTEMP is attempted.

Example 2:

```
.R CREF
*SBRLS/R
```

Given to the Command Decoder, this command string causes output to be sent to the line printer. The input is expected to be a SABR listing file named SBRLS.LS or SBRLS from device SYS:.

Example 3:

```
.R CREF
*DTA1:LIST<DTA3:PALIST/X
```

This command string causes output to be sent to DECtape unit 1, as a file named LIST.LS. Input is expected to be a PAL8 listing file called PALIST.LS or PALIST. No literals appear in the CREF output table.

Example 4:

```
.R CREF
*DTA2:LIST<SYS:BIGLST
```

The source listing, symbol table, and cross-reference of symbols in the file BIGLST or BIGLST.LS on SYS is in the file LIST.LS on DTA2. To list the CREF output you may now run PIP.SV as follows:

```
.R PIP
*LPT:<DTA2:LIST.LS
```

11.2 PSEUDO-OP HANDLING

The PAL8 and SABR assemblers have certain pseudo-ops that cause CREF to perform actions similar to those taken by the assembler whose output is being processed. These pseudo-ops are described below:

PAL8 Pseudo-Op	Action Taken by CREF
EXPUNGE	CREF purges its current symbol table of all permanent and user-defined symbols. If any literals were in the symbol table, they are not deleted.
FIXTAB	Causes all symbols (except literals) to be marked as permanent symbols. After a FIXTAB, no references will be reported by CREF.
TEXT	Ignores characters between delimiters.
$	End-of-input signal.

11.3 INTERPRETING CREF OUTPUT

The output of CREF consists of two parts. On the first pass through the input file CREF generates a sequentially numbered listing file. The sequence numbers are decimal. The /P and /U options disable this part of the output.

The cross-reference table appears after the listing. This table contains every user-defined symbol and literal, sorted alphabetically. An underline (or back-arrow on most DEC terminals) indicates each literal, and it is followed by the field and address where the literal occurs. For each symbol and literal there appears a list of numbers specifying the line in which each is referenced.

If CREF finds too many references to fit into core at one time, multiple passes are required to process all symbols. The minimum number of passes is two. The maximum number of passes depends on the size of the input file and on the amount of core available. CREF calculates the number of core fields available and uses all available space for reference tables. If there is not enough core available, three or more passes are required. For example, the current OS/8 SABR assembler (5518 source lines, 849 symbols) requires four passes through CREF on an 8K machine.

The following example illustrates a program that PAL8 has assembled and CREF has listed. Form feeds on the terminal have been converted to a series of carriage return/line feed combinations followed by a dotted tear line. Notice the line in the CREF table where a # follows the defined symbol. All literals and symbols defined by OPDEF or SKPDF in SABR do not have a # following them.

```
/EXAMPLE PROGRAM

/   EXAMPLE PROGRAM                          PAL8-V9B  03/05/74  PAGE 1

   1                    /   EXAMPLE PROGRAM
   2                    /   ILLUSTRATING DETAILS OF LISTING FORMAT
   3                    /   USING PAL8 AND CREF
   4            0200    *200
   5   00200    7300    START,  CLA CLL
   6   00201    1207            TAD A           /CURRENT PAGE SYMBOL
   7   00202    1777'           TAD B           /OFF-PAGE SYMBOL, LINK GENERATED
   8   00203    1177            TAD C2          /PAGE ZERO LITERAL
   9   00204    1376            TAD (3          /CURRENT PAGE LITERAL
  10   00205    3777'           DCA LINK        /OFF-PAGE SYMBOL, LINK GENERATED
  11   00206    5610            JMP I ADDRP2    /USER CREATED LINK
  12   00207    0011    A,      0011
  13   00210    0400    ADDRP2, P2              /INDIRECT ADDRESS
  14   00376    0003
  15   00377    0407
  16            0400    *400
  17   00400    1207    P2,     TAD LINK        /PAGE 2 START
  18   00401    1377            TAD (3          /NOTE THAT THIS IS A NEW LITERAL
  19   00402    1177            TAD C2          /NOTE THAT THIS IS SAME OLD LITERAL
  20   00403    1377            TAD (3          /SAME AS CURRENT PAGE LITERAL
  21   00404    3207            DCA B           /CURRENT PAGE SYMBOL
  22   00405    6213            CDF CIF 10      /CHANGE FIELDS
  23   00406    5776'           JMP FLD1        /OFF PAGE SYMBOL, LINK GENERATED
  24   00407    0000    LINK,   0
  25            0407            B-LINK
  26   00576    0200
  27   00577    0003
  28   00177    0002
  29            0001    FIELD 1
  30   10200    1377    FLD1,   TAD (3          /FIELD 1, DEFAULT TO PAGE 1 *200
  31   10201    1177            TAD C2          /NEW LITERAL, BECAUSE IN PAGE 0 OF NEW FIELD
  32   10202    6203            CIF CDF 0       /CHANGE FIELDS AGAIN
  33   10203    5200            JMP START       /NO LINK GENERATED, SAME PAGE, OTHER FIELD
  34   10377    0003
  35                    $
  36   10177    0002
```

```
/   EXAMPLE PROGRAM                          PAL8-V9B  03/05/74  PAGE 2

A       0207
ADDRP2  0210
B       0407
FLD1    0200
LINK    0407
P2      0400
START   0200

ERRORS DETECTED: 0
LINKS GENERATED: 3


A             6    12#
ADDRP2       11    13#
B             7    21     25#
FLD1         23    30#
LINK         10    17     24#   25
P2           13    17#
START         5#   33
_00177        8    19
_00377        9
_00577       18    20
_10177       31
_10377       30

V3
```

11.4 RESTRICTIONS

CREF has the following restrictions:

- CREF can handle a maximum of 896 (decimal) symbols in one major pass. (In 8K, PAL8 is limited to 897 symbols while SABR is limited to fewer than 800 symbols.) If CREF finds more than 896 symbols, it generates an error message.
- If any symbol in the input file has more than 2044 (decimal) references, an error message appears.
- If more than 8192 (decimal) source lines are input, sequence numbers return to 4096, not 0.
- If you use the /D option in PAL8 (to generate a DDT-compatible symbol table) and you put the output listing through CREF, no symbol table listing will appear.
- Use of semicolons - This is a restriction that, when not observed, could cause errors in the CREF table. You should follow these suggestions when preparing source files in order to insure a proper CREF listing. Do not use semicolons on lines with pseudo-ops. In particular, do not use a combination such as the following:

```
*3000
TEST   %ERROR% ;  TAD [42

EXPR=0
```

In this case, CREF does not process the page zero literal properly. It generates a literal derived from the expanded TEXT message. No error message generates, but the literal table entry is meaningless. As a general rule, do not use semicolons as line terminators inside conditional assembly brackets (<>). For example:

```
EXOR=0
IFNZRO EXOR<CLA;TAD B;  HLT \ERROR>
\THIS IS THE NEXT LINE PAST IFNZRO
```

The conditional code is not assembled; but because CREF does not realize this, it tries to process the bracketed instructions. As a result of the semicolons, extra symbols may be processed and some valid references missed. However, if the code had been assembled CREF would operate properly. There are two ways around this:

a. Write straight-line code:

```
EXOR=0
IFNZRO EXOR <
CLA
TAD B
HLT ERROR
>
```

b. Use XLIST around conditional code, in the preceding example:

```
IFZERO EXOR <XLIST>
IFNZRO EXOR <CLA;TAD B;  HLT\ERROR>
IFZERO EXOR <XLIST>
```

XLIST turns off the listing, if the code does not assemble, and turns it back on after the conditional code.

- Formats - There are several output formats you can use in generating a PAL8 listing file:

  /T Form feeds converted to carriage return/line feeds.
  /H No heading or form feeds generated.
  /D DDT-compatible symbol table is generated.

  For best results with CREF, use none of these switches. This generates a heading and form feed in the output. CREF automatically converts form feeds to carriage return/line feeds if output is to the terminal.

- PAL8-generated links are not recognized by CREF. CREF processes only literals specifically generated with (and [.

11.5 CREF ERROR MESSAGES

CREF errors are nonrecoverable errors, and control returns to the Keyboard Monitor through location 07605 (no core saved). Table 11-2 lists the error messages printed by CREF.

Table 11-2
CREF Error Messages

Error Message	Meaning
SYM OVERFLOW	More than 896 (decimal) symbols and literals were encountered during a major pass.
ENTER FAILED	Entering an output file was unsuccessful - possibly output was specified to a read-only device.
OUT DEV FULL	The output device is full (directory devices only).
CLOSE FAILED	CLOSE on output file failed.
INPUT ERROR	A read from the input device failed.
DEV LPT BAD	The default output device, LPT, is not available on this system.
2045 REFS	More than 2044 (decimal) references to one symbol were made.
HANDLER FAIL	This is a fatal error on output; it can occur if either the system device or the selected output device is WRITE-LOCKed.

CHAPTER 12

DIRECT

DIRECT is an OS/8 program that produces listings of OS/8 device directories. The directories produced vary depending upon the options you specify in the DIRECT command line. The standard directory listing consists of the following columns: file name, file name extension, length (decimal) in blocks written, and creation date.

DIRECT supports the wild card construction, accepting * in place of the file name or extension, or ? in place of a character. See the FOTP chapter for a description of wild card construction.

12.1 CALLING AND USING DIRECT

To call DIRECT from the system device, type:

```
.R DIRECT
```

in response to the Keyboard Monitor dot. You may also call DIRECT via the CCL command DIR. The Command Decoder prints an asterisk in the left margin, indicating it is ready to accept a line of I/O files and options. You can enter one output specification, and one to five input specifications in a DIRECT command line. You may terminate the I/O command line with a carriage return (DIRECT retains control) or with an ALTMODE (control returns to the Keyboard Monitor).

The output specification consists of a device upon which you can produce the directory, a file name, and a file name extension. All parts of the output specification are optional, as is the output specification itself. You should specify a file name and extension if you desire to save the directory for listing at a later time. If you specify no output device, TTY is assumed. If you give a file name without an extension, the extension .DI is assumed. The wild card ? and * are not permitted in DIRECT output file names and extensions.

A DIRECT input specification consists of a device, an optional file name, and an optional extension. The wild cards * and ? are permitted in input specifications. If you specify an input device with no file name or extension, *.* is assumed. DIRECT determines which files have the form specified and prints a directory listing of just those files.

NOTE

If you want to include the date in your directory listing, you must enter it first with the DATE command.

12.1.1 DIRECT Options

The following table lists the options you can use in a DIRECT I/O specification line. Examples of the use of these options are shown after Table 12-1.

Table 12-1
DIRECT Options

Option	Meaning
/B	Include the starting block numbers (octal) for each file in the directory.
/C	List only files with the current date, i.e., the date entered with the most recent DATE command.
/E	Include empty file spaces in the directory listing.
/F	List a short form of the directory, omitting file lengths and dates.
/I	List additional information words in octal, other than the first that is listed as the date.
/L	List the standard form of the directory, including file name, extension, length in blocks, and creation date. The /L option is assumed if none is specified.
/M	List only the empty spaces in the directory.
=n	Use n columns in the directory listing. This option allows you to specify the number of directory entries per line of output. The "n" must be in the range 0 to 7. The =n option is useful when a wide column printer, e.g., 132 columns, is being used.
/O	List only files with other than the current date.
/R	List the remainder of the files after the first one found. This option causes DIRECT to find the first file that matches the specifications given and then list a directory that includes the first matching file and all files that follow it on the device. The /C and /O options are still considered when listing these remaining files. If /R and /V are used in the same command, only the first file of the form specified is listed.
/U	Treat each input specification separately. The /U option creates a separate directory listing for each input specification.
/V	List files not of the form specified.
/W	Print the version number of DIRECT.

12.2 DIRECT EXAMPLES

The following are legal command strings to DIRECT and the resultant DIRECT output. To facilitate understanding of the DIRECT options, the same device (DTA0) is used for each of the examples. The current date is 21-JAN-74.

When DIRECT has completed an operation, control returns to the Command Decoder for additional input.

Example 1:

This example shows a directory of all the files on DTA0, listed in two columns on the terminal (TTY).

```
.R DIRECT
*DTA0:=2

 21-JAN-74

MTPALA.PA    1 18-JAN-74     WNTSTA.BA    1 18-JAN-74
MTPALB.PA    1 18-JAN-74     WNTSTB.BA    1 19-JAN-74
WNTSTC.BA    1 19-JAN-74     WNPALA.PA    1 19-JAN-74
WNPPPA.PA    1 19-JAN-74     WNTSTD.BA    1 21-JAN-74
WNPALB.PA    1 21-JAN-74     MTPALC.PA    1 21-JAN-74
WNXX  .BA    1 21-JAN-74     WNXY  .BA    1 21-JAN-74

 718 FREE BLOCKS
```

Example 2:

This example shows all files that have a file name beginning with WN, have any file extension, and do not have the current date. The directory is listed in two columns on TTY.

```
*DTA0:WN????.*/O=2

 21-JAN-74

WNTSTA.BA    1 18-JAN-74     WNTSTB.BA    1 19-JAN-74
WNTSTC.BA    1 19-JAN-74     WNPALA.PA    1 19-JAN-74
WNPPPA.PA    1 19-JAN-74

 718 FREE BLOCKS
```

Example 3:

This example shows files that have any file name, have a .BA extension, and have the current date. TTY lists the directory in a single column.

```
*DTA0:*.BA/C

 21-JAN-74

WNTSTD.BA    1 21-JAN-74
WNXX  .BA    1 21-JAN-74
WNXY  .BA    1 21-JAN-74

 718 FREE BLOCKS
```

Example 4:

This example demonstrates the use of the /U option to produce separate directories for each input specification. The command specifies that all files beginning with WN and having .BA extensions appear first, and that all files beginning with WN and having .PA extensions appear next. The short form of the directory is to be listed on the line printer (LPT) in three columns.

```
*LPT:<DTA0:WN????.BA,WN????.PA/F/U=3

 21-JAN-74

WNTSTA.BA   WNTSTB.BA       WNTSTC.BA
WNTSTD.BA   WNXX  .BA       WNXY  .BA

718 FREE BLOCKS

 21-JAN-74

WNPALA.PA   WNPPPA.PA       WNPALB.PA

 718 FREE BLOCKS
```

Example 5:

This example demonstrates the use of the /V option to print files not of the specified form and the use of the /O option to exclude files with the current date. TTY is to print in a single column all files except those beginning with WN.

```
*DTA0:WN????.*/O/V

 21-JAN-74

MTPALA.PA   1  18-JAN-74
MTPALB.PA   1  18-JAN-74

 718 FREE BLOCKS
```

Example 6:

This example demonstrates the use of the /R option to list part of the directory. DIRECT finds the first file that begins with WN and has a .PA extension; that file and all files that follow are listed. TTY lists the directory in two columns.

```
*DTA0:WN????.PA/R=2

 21-JAN-74

WNPALA.PA   1  19-JAN-74    WNPPPA.PA   1  19-JAN-74
WNTSTD.BA   1  21-JAN-74    WNPALB.PA   1  21-JAN-74
MTPALC.PA   1  21-JAN-74    WNXX  .BA   1  21-JAN-74
WNXY  .BA   1  21-JAN-74

 718 FREE BLOCKS
```

12.3 DIRECT ERROR MESSAGES

The following error messages may appear when running the DIRECT program.

Table 12-2
DIRECT Error Messages

Message	Meaning
BAD INPUT DIRECTORY	This message occurs when the input device has a bad directory, e.g., the device is not an OS/8 device, or a DECtape has not been zeroed.
DEVICE DOES NOT HAVE A DIRECTORY	The input device is a non-directory device, e.g., PTR. DIRECT can only read directories from file structured devices.
EQUALS OPTION BAD	The =n option is not in the range 0-7.
ERROR CLOSING FILE	System error.
ERROR READING INPUT DIRECTORY	An error occurred while reading the directory.
ERROR WRITING FILE	An error occurred while writing the output file.
ILLEGAL *	An asterisk (*) was included in the output file specification or an illegal * was included in the input file name.
ILLEGAL ?	A question mark (?) was included in the output file specification.
NO ROOM FOR OUTPUT FILE	The output device does not have sufficient space for the directory to be written.
THERE IS NO HOPE - THERE IS NO TTY HANDLER IN YOUR SYSTEM!	A command was issued to print a directory on the terminal when no TTY handler is present on the OS/8 system. Use BUILD to insert a TTY handler in the system.

CHAPTER 13

DECTAPE COPY AND FORMAT PROGRAMS

The following programs enable you to format and copy DECtapes.

13.1 DTFRMT

This program records the required timing and mark tracks on a DECtape mounted on the TC01-TU55 unit or a TC08-TU56 DECtape unit.

The program interacts with you via the terminal to obtain the necessary data for each set of DECtapes to be formatted. As soon as one set of tapes is formatted, the program is ready to format another set.

Two full passes are required to completely format each DECtape, and up to eight DECtapes may be formatted at a time (assuming that you have eight tape transports). With a minimum of operator-program communication, you can mount and format new tapes in the same fashion upon completion of a cycle.

13.1.1 Loading Procedure

Load the program into core using the standard Binary Loader.

13.1.2 Using the Program

To start the program from the console, key 1000 into the SWITCH REGISTER. Depress LOAD ADDRESS and depress START. DTA? is printed on the terminal.

Mount the DECtapes to be marked onto the tape transports, with just enough turns of tape on the right-hand reel of each transport to provide a grip. Make sure that no two tape units are set to the same unit number. Set the RDMK-WRTM-NORMAL switch located on the TC01 maintenance control panel to the WRTM position. For each transport to be used, set the WRITE ENABLED-WRITE LOCK switch to WRITE ENABLED, and the REMOTE-OFF-LOCAL switch to REMOTE.

To run the program from the terminal, type:

```
.R DTFRMT
```

in response to the Keyboard Monitor dot. You now converse with the program. The printout:

```
DTA?
```

asks which DECtape units you are using. Type a unit number or series of unit numbers, corresponding to the DECtape units with mounted tapes. For instance, if you have mounted tapes on units 2, 5, 7, and 8, type 2 5 7 8 followed by a carriage return. Spaces are ignored, so it makes no difference if you type spaces between the unit numbers. Only one specification of a unit is significant, i.e., typing 2 2 5 7 7 5 8 2 8 has the same effect as typing 2 5 7 8.

Once you have specified the units you wish to use, the program types:

DIRECT?

Respond by typing:

MARK or MARK XXXX

If you type:

MARK

the program assumes 201(8) words, 2702(8) blocks (standard PDP-8 format). Otherwise, XXXX is accepted as a decimal number of words per block, and must be divisible by 3. Note that typing MARK 384 will cause the program to generate a standard PDP-10 format DECtape (1102(8) blocks of 600(8) words, which is equivalent to 1102(8) blocks of 200(8) words, where each word is 36 bits rather than 12 bits).

The program now types:

XXXX WORDS, YYYY BLOCKS OK? (YES OR NO)

This serves as a final check for block count. XXXX and YYYY are octal values representing the final outcome of a formula solved by the program. They determine the number of blocks you may write on a DECtape, given a specified number of words. If you give a NO answer, the program reverts to DIRECT?. If YES, the tape on the first unit specified begins to move.

Once all of the tapes specified have been marked, the printout:

SET SWITCH TO NORMAL

appears. Return the RDMK-WRTM-NORMAL switch to NORMAL, and strike the RETURN key on the terminal, starting the second pass. Note that during the second pass with multiple DECtape units, as soon as one tape stops and the next tape starts, the first tape is completed and may be replaced with a fresh tape in preparation for recycling.

The program continues by itself until completed. Now the DIRECT? printout occurs. Typing:

SAME<

repeats the entire process with the original constants. The new DECtapes must be mounted and ready to write timing and mark tracks before you type SAME. Also, in response to DIRECT?, typing RDR causes the printout of the unit numbers of the DECtapes and the last twelve block numbers. RDF causes the printout of the unit numbers and the first twelve block numbers. RESTART returns the program to DTA?. Unit numbers are printed as N000, where N is the unit number (0 means DECtape unit 8). Once formatting begins, control C will cause the program to restart at DTA?. If you wish to return to the monitor, type control C.

Following are several examples of successful operation. The program prints the underlined portions. A carriage return should follow all responses.

1. Create a standard tape on unit 4.

```
DTA? 4
DIRECT? MARK
0201 WORDS, 2702 BLOCKS OK? YES OR NO
YES
SET SWITCH TO NORMAL
DIRECT?
```

2. Create 16 standard PDP-10 format tapes - eight at a time, on units 1-8.

```
DTA? 12345678
DIRECT? MARK 384
0600 WORDS, 1102 BLOCKS OK? YES OR NO
YES
SET SWITCH TO NORMAL (TYPE <CR>)
DIRECT? SAME
SET SWITCH TO NORMAL (TYPE <CR>)
DIRECT?
```

13.1.3 Error Messages

Errors typed to DTA? and DIRECT? revert back to DTA? or DIRECT?

Error messages for response to MARK XXXX:

NOT DECIMAL	A character in XXXX is not 0-9.
NOT DIVISIBLE BY 3	XXXX cannot be divided evenly by 3.
TOO MANY WORDS	The number of words plus 15 exceeds 7777(8).
TOO MANY BLOCKS	The number of blocks generated by XXXX exceeds 7777(8).

Error messages for response to YES (after message - revert back DTA?):

SETUP?	Indicates an error in the DECtape setup: unit in WRITE-LOCK nonselectable unit switch not in WRTM position

Error messages for marking and verifying a tape:

```
XXXX SHOULD BE YYYY BLK ERROR PHASE X
XXXX SHOULD BE YYYY DATA ERROR PHASE X
END TAPE ERROR PHASE X
MARK TRACK ERROR PHASE X
PARITY ERROR PHASE X
SELECT ERROR PHASE X
TIMING ERROR PHASE X
LAST INT NOT END ZONE
```

Although an error message should cause doubt concerning the entire process, you can restart by phases (except when in phase 0) by typing RETRY<. Type RESTART< to return to DTA?.

```
PHASE 0:  MARK TRACK WRITE
PHASE 1:  WRITING LAST REVERSE BLOCK NUMBER FORWARD
PHASE 2:  WRITING BLOCK NUMBERS AND DATA IN REVERSE
PHASE 3:  READING AND CHECKING BLOCK NUMBERS AND DATA
```

The error message LAST INT NOT END ZONE indicates an interrupt occurred between the first or last block number and the end zone.

You can restart the entire program at 1000(8) any time.

13.1.4 Details of DTFRMT Operation and Storage

The program writes timing and mark tracks on a DECtape. It then inserts block numbers and parity correct information, checking the results of all operations.

The number of block frames the program writes is a function of the number of words per block. The formula

$$\text{blocks per tape} = \frac{212080}{NW+15} + 2$$

(where NW equals the number of words the program writes) is used by the program to compute the number of blocks. But this number is adjusted by the program to provide the standard PDP-8 format of 129 (12-bit) words, 1744 blocks, and standard PDP-10 format of 128 (36-bit) words, 578 blocks.

Two full passes are required to mark and verify a tape.

Pass 1 — Marks the tape forward, inserts block numbers and parity correct date in reverse.

Pass 2 — Reads and checks block numbers and data forward and reverse.

During the forward direction of the first pass, the TC01 switches into WRITE TIMING AND MARK TRACKS, CONTINUOUS MODE, FORWARD. The program manipulates data it is writing by monitoring the word count register and the DTF, (DECtape flag). Initially, the program writes ten feet of end-zone code, and abutting the end zone are about two standard block lengths of interblock sync. To the TC01, this interblock sync acts as no operation, but guarantees that at turn-around time, block 0 is read first (or 2701 if turning out of the forward end zone). Now the program writes the remainder of the tape, creating block frames. The formula above determines the number of such frames. Upon completion of the block framing, another extended interblock sync zone is written, as well as ten feet of end zone.

Pass 1 forward is now complete (timing and mark tracks are written). The program orders the tape to MOVE in reverse for three seconds, thus moving it out of the end zone and onto the marked section. The tape once again moves forward, and the program writes the last REVERSE BLOCK NUMBER until it senses the forward end zone. The tape now turns out of the end zone in SEARCH, and the program waits for a block interrupt (first reverse block number). When the DTF rises, the TC01 switches into WRITE ALL, CONTINUOUS, REVERSE. Thus DTFRMT synchronizes the system and writes all block numbers and data, until the forward end zone is sensed. This completes the marking and blocking of the tape. Pass 2 in CONTINUOUS MODE checks the data and block numbers to be certain they are correct. When you specify multiple DECtape units, Pass 1 forward ends for each tape before Pass 1 reverse begins.

13.2 TDFRMT

The TD8-E DECtape formatter program records the timing and mark tracks on a DECtape mounted on the TU56 DECtape transport.

TDFRMT interacts with you via the terminal to obtain the necessary data for each set of DECtapes the program will format. As soon as one set of tapes is formatted, the program is ready to format another set.

The program requires three full passes to completely format each DECtape, and it can format up to two DECtapes at a time (units 0 and 1). Upon completion of a cycle, the program can mount and format new tapes in the same way, with a minimum of operator-program communication. Excluding tape setup time, the program formats one tape in three minutes from start to finish.

Mount the DECtapes to be marked onto the tape transports with just enough turns of tape on the right-hand reel of each transport to provide a grip. Make sure that no two tape units are set to the same unit number. Set the switch on the TD8-E to WTM position. For each transport you are using, set the WRITE ENABLED-WRITE-LOCK switch to WRITE ENABLED, and the REMOTE-OFF-LOCAL switch to REMOTE.

13.2.1 Operating Procedures

Type:

```
.R TDFRMT
```

in response to the Keyboard Monitor dot. You are now set to converse with the program. The printout:

```
UNIT?
```

asks which DECtape units you are using. Type one or two unit numbers corresponding to the DECtape units with mounted tapes. For instance, if you have mounted tapes on units 0 and 1, type 0 1. Spaces are ignored, so it makes no difference if you type spaces between the unit numbers. Only one specification of a unit is significant, i.e., typing 000111 has the same effect as typing 01.

Once you have specified the unit(s) you wish to use, the program types:

```
FORMAT?
```

Respond by typing:

```
MARK or MARK XXXX
```

If you type:

```
MARK
```

the program assumes 201 words, 2702 blocks (standard PDP-8 format). Otherwise the program accepts XXXX as a decimal number of words per block that must be divisible by 3. Note that typing MARK 384 will cause the program to generate standard PDP-10 format DECtapes (1102(8) blocks of 600 words, which is equivalent to 1102(8) blocks of 200 words, where each word is 36 bits rather than 12 bits).

The program now types:

```
XXXX WORDS, YYYY BLOCKS OK?  (YES OR NO)
```

This serves as a final check for block count. XXXX and YYYY are octal values representing the final outcome of a formula solved by the program. This determines the number of blocks the program can write on DECtape, given a specified number of words. If you give a NO answer, the program reverts to FORMAT?. If YES, the program types out SET SWITCH TO WTM. Hit carriage return on the teletype and the tape on the first unit specified begins to move, if you have set the switch.

Once the program has marked all of the specified tapes, the printout SET SWITCH TO OFF appears. Reset the WTM switch to off, and strike the return key on the terminal, starting the second pass. Note that during the second pass with multiple DECtape units, as soon as one tape stops and the next tape starts, the first tape is completed. You may replace it with a fresh tape in preparation for recycling.

The program continues by itself until it is completed. At this point the FORMAT printout occurs. Typing:

```
SAME<
```

repeats the entire process with the original constants. Make sure you have mounted the new DECtapes before you type a carriage return in response to the message SET SWITCH TO WTM. The tapes should be ready to write timing and mark tracks. Also, in response to DIRECT?, typing RDR causes the printout of the unit number of the DECtape and the last 22 block numbers. RDF< causes the printout of the unit number and the first 22 block numbers. RESTART< returns the program to UNIT?. Unit numbers are printed as 000N, where N is the unit number.

Following are several examples of successful operations. The program prints the underlined portions. A carriage return should follow all responses.

1. Create a standard PDP-8 tape on unit 1.

```
UNIT? 1
FORMAT? MARK
0201 WORDS, 2702 BLOCKS, OK? (YES OR NO)
YES
SET SWITCH TO WTM
SET SWITCH TO OFF
FORMAT?
```

2. Create four standard PDP-10 format tapes, two at a time on units 011.

```
UNIT? 01
FORMAT? MARK 384
0600 WORDS, 1102 BLOCKS OK? (YES OR NO)
YES
SET SWITCH TO WTM
SET SWITCH TO OFF
FORMAT? SAME
SET SWITCH TO WTM
SET SWITCH TO OFF
FORMAT?
```

13.2.2 Error Messages

Errors typed to UNIT and FORMAT revert back to UNIT? or FORMAT?.

Error messages for response to MARK XXXX:

NOT DECIMAL	A character in XXXX is not 0-9
NOT DIVISIBLE BY 3	XXXX cannot be divided evenly by 3
TOO MANY WORDS	The number of words plus 15 exceeds 7777(8)
TOO MANY BLOCKS	The number of blocks generated by XXXX exceeds 7777

Error messages for response to SET SWITCH TO WTM:

SETUP?

Indicates an error in the DECtape setup. One of the units specified is in write-lock position, not selected, or the write flip-flop is unable to be set, or there may be a timing error. (After message revert back to UNIT.)

Switch not set to WTM or single-line flag failed to set.
Set switch to WTM.

This typeout says that either the switch on the M868 modules is not set to the WTM position or the timing generator for writing the mark and timing tracks is not setting the single-line flag.

RECOVERY:

If you did not set the switch to WTM position, set the switch and hit carriage return on the teletype.

If the switch was set to WTM position and this type out occurred, try again or examine the timing generator circuit.

Error messages for marking and verifying a tape:

```
PC   XXXX  MARK TRACK ERROR PHASE Y
PC   XXXX  BLOCK NUMBER ERROR PHASE Y
PC   XXXX  DATA ERROR PHASE Y
PC   XXXX  CHECKSUM ERROR PHASE Y
PC   XXXX  TIMING ERROR PHASE Y
PC   XXXX  WRITE ERROR PHASE Y
```

XXXX equals the program counter at time of the failure. Y equals the number of the pass involved.

Although an error message should cause doubt concerning the entire process, you can restart the phase (except in phase 0) by typing RETRY<. Type RESTART< to return to UNIT?.

PHASE 0: WRITE TIMING AND MARK TRACK FORWARD

PHASE 1: READS MARK TRACK REVERSE

PHASE 2: WRITE DATA, FORWARD BLOCK AND REVERSE BLOCK NUMBERS FORWARD AND WRITES THE CHECKSUMS

PHASE 3: DISPLAYS BLOCK NUMBERS IN AC REVERSE

PHASE 4: READS DATA, FORWARD BLOCK AND REVERSE BLOCK NUMBERS FORWARD AND CALCULATES THE CHECKSUM

PHASE 5: READS REVERSE BLOCK NUMBERS IN REVERSE

You can restart the entire program at 0200 any time.

13.2.3 Details of TDFRMT Operation and Storage

The program writes timing and mark track in a forward direction on a DECtape with the WTM switch set. Then it reads the mark track in the reverse direction with the switch set to off. The program checks all of the mark track once it is in sync. When it finishes reading the mark track reverse, it bounces off the end zone and starts writing zeroes to the first block mark. The program is now in sync.

The program now continues writing forward block numbers, reverse checksum, data, checksum, and reverse block numbers for the rest of the tape. When it sees the end zone, it turns around. It starts displaying the reverse block number in the accumulator until it hits the end zone again.

At this point the tape turns around and starts reading and comparing all forward block numbers; reverse checksum; all data, checksum, and reverse block numbers that were written in Phase 2. This comparison is made on all blocks until the end zone is reached. The tape turns around in the end zone and starts looking for reverse block numbers and comparing them all the way down the tape to the end zone. The formatting is now complete, the tape stops, and FORMAT is typed out waiting for new directions.

The number of block frames to be written is a function of the number of words per block.

The program uses the formula

BLOCKS PER TAPE [(212080)/(NW+15)]+2

(where NW equals the number of words to be written) to compute the number of blocks, but it is adjusted by the program to provide the standard PDP-8 format of 129(10) (12-bit) words, 1474(10) blocks, and standard PDP-10 format of 128(10) (36-bit) words, 578(10) blocks.

The writing of the mark track is done through AC bits 0, 3, 6, and 9. The following description shows how the program writes the mark track.

1. Install the tape with enough turns to create a pull. The reverse end zone requires a sequence of three data words for its pattern.

   4044
   0440
   4404

   In the mark track the words appear at 101101101101101 (5555(8)). The reverse end zone should cover about 10 feet of tape. Write the above three words 4096(10) times.

2. Write the three words in point 3, or expand code 99 times.

3. Expand code, three words of expand code should immediately follow each block.

```
0404
0404
0404
```

   In the mark track the words appear as 010101010101 (2525(8)).

4. The forward block mark and reverse guard require three words.

```
0404
4004
4040
```

   Which appear on the mark track as 010110011010 (2632(8)).

5. The lock mark, reverse checksum, reverse final, reverse prefinal consist of six PDP-8 memory words.

```
0040
0000
4000
0040
0000
4000
```

   These words appear on the mark track as 001000001000001000001000 (10101010(8)).

6. Mark track code for data is generated by

```
4440
0044
4000
```

   These three words appear as 111000111000 (7070(8)) and are repeated 41(10) times for a 129-word block.

7. The prefinal, final, checksum, and reverse lock consist of six PDP-8 words.

```
4440
4444
4044
4440
4444
4044
```

   These words appear on the mark track as 111011111011111011111011 (73737373(8)).

8. The guard and reverse block mark consist of three words

```
4040
0440
0404
```

   which appear as 101001100101 (5145(8)).

9. Generate 2702(8) block patterns. Repeat 3 through 8 2702(8) times.

10. 100 expand codes (see 3).

11. The end zone pattern consists of three words,

    0400
    4004
    0040

    which appears on the mark track as 010010010010 (2222(8)). Repeat these three words 4096(10) times.

13.3 DTCOPY

A dialog on the terminal controls the TC01, TC08, and TU-55 Copy Program. Your responses to the questions are in the form of octal numbers followed by a carriage return. Separate the answers with semicolons when more than one is required. Alphabetic or other illegal characters will cause an error message to be generated and the question to be repeated. If you type too many digits for the expected response, only the last ones typed will be used. If the response was to be either 0 or 1 (YES or NO), a non-zero final digit will be assumed to be 1.

Before answering the dialog's questions, you should make sure that all the DECtapes involved are mounted on their respective drives and all drives set to REMOTE. You may set the input drive to WRITE-LOCK or WRITE ENABLE; all output drives must be set to WRITE ENABLE. No two drives may have the same unit number.

Type:

```
.R DTCOPY
```

in response to the Keyboard Monitor dot. The program prints:

```
DECtape COPY V10A
```

For each set of copies, the dialog is as follows:

```
DECTAPE COPY V10A
FROM UNIT 0

TO UNIT 2

FIRST BLOCK TO COPY (OCTAL) 0

FINAL BLOCK TO COPY (OCTAL) 700

PDP-8 WORDS PER BLOCK 0201

VERIFY OUTPUT? (0=YES, 1=NO): 0
```

When all specified copies have been finished, the tapes are rewound and the dialog continues:

```
DONE
DECTAPE COPY V10A
FROM UNIT
```

You may return to the monitor by typing CTRL/C at any time. (Control characters are not echo printed.)

13.3.1 Error Messages

DTCOPY produces the following error messages:

ILLEGAL RESPONSE

Your response to the dialog was not correct; for example, an alphabetic character was typed or carriage return was typed before an octal number was given. The question will be restated and any previous answer ignored. Type nothing until the terminal has stopped printing.

SELECT ERROR UNIT n

During attempted data transfer, unit n was not found. The program waits for you to correct the cause of the error. You should check to see that:

1. when unit n is an output drive, it is set to WRITE ENABLE.
2. unit n is set to REMOTE.
3. there is only one unit n.
4. all units are set to numbers appropriate to their TD8E internal wiring.

When you have corrected the cause of the error, you may type CTRL/R to resume transfer or type CTRL/S to restart the dialog.

TAPE ERROR BLOCK x UNIT n

During attempted transfer, a parity error or timing error was detected, or too great a block number was requested near block x on the tape on unit n. The tapes are rewound and the dialog is automatically restarted at DONE, REPEAT (YES=1, NO=0).

VERIFY ERROR BLOCK x UNIT n

The data on the input tape does not match the data that was written on block x of the output tape on unit n. You may type CTRL/R to ignore the error and continue with the transfer, CTRL/T to try the last transfer again and continue if the error does not recur, or CTRL/S to restart the dialog.

ILLEGAL FORMAT UNIT n

Indicates one of two situations: Either the number of words per block on unit n does not agree with the number of words per block on the input unit; or, when the number of blocks on the tape was calculated from the block length of the input tape, the length was found to be illegal. The number of blocks is only calculated if you request the whole tape copy option. In either case, when the error has been corrected, you may type CTRL/R to check the formats of all tapes again and continue, or CTRL/S to restart the dialog.

13.4 TDCOPY

A dialog on the terminal controls TD83 Copy. Your responses to the questions are in the form of octal numbers followed by a carriage return. Separate the answers with semicolons when more than one is required. Alphabetic or other illegal characters will cause an error message to be generated and the question to be repeated. If you type too many digits for the expected response, only the last ones typed will be used. If the response was to be either 0 or 1 (YES or NO), a non-zero final digit will be assumed to be 1.

Before answering the dialog's questions, make sure that all the DECtapes involved are mounted on their respective drives and all the drives set to REMOTE. You may set the input drive to WRITE-LOCK or WRITE ENABLE; all output drives must be set to WRITE ENABLE. No two drives may have the same unit number.

Type:

```
.R TDCOPY
```

in response to the Keyboard Monitor dot. The program prints:

```
TD8E COPY V4A
HIGHEST FIELD AVAILABLE:
```

Respond with the number of the highest field you want used for buffer space. This response may allow data to be preserved in any higher field or may make full use of the available memory. This question is asked only once: immediately after you have loaded the program. To change the response, you must execute the program again. If you want to use 4K of memory, respond with a 0; if 8K, respond with a 1, and so forth.

For each set of copies, the dialog is as follows:

Dialog	Comments
FROM UNIT: 0	You may specify one unit number.
TO UNITS: 1;2;3;4;5;6;7	You may specify up to seven unit numbers, separated by semicolons.
FIRST INPUT BLOCK: 100	You may supply any legal DECtape block number.
FIRST OUTPUT BLOCK: 200	You may supply any legal DECtape block number.
NUMBER OF BLOCKS TO COPY: 50	You may supply appropriate number of blocks.
VERIFY OUTPUT (YES=1, NO=0): 1	
0201 12-BIT WORDS PER BLOCK	Determined by program from tape on input unit.

The program checks the block length of all the specified tapes. If any are found to be different from the input tape, the ILLEGAL FORMAT UNIT n error message is generated.

When the program has finished all specified copies, it rewinds the tapes and the dialog continues:

```
DONE

REPEAT (YES=1, NO=0):
```

If you want to copy more tapes with the same set of specifications, you should place them on the drives before typing 1 to repeat the previous operation. If you desire a different set of specifications, type 0 to restart the dialog.

Occasionally a TD8E drive will not stop fast enough after the tapes have rewound, causing the end of the tape to spin off the reel. If this should happen, you can set the drive to OFF and stop the reel by hand. This will not affect the validity of the copy. If the dialog does not continue properly after one or more tapes have spun off, you can restart the program.

In response to any question in the dialog, type CTRL/S to restart the dialog at REPEAT (YES=1, NO=0) or CTRL/C to exit the monitor. You may also type either CTRL/S or CTRL/C during a small amount of further motion. If you type CTRL/S during the dialog, you answer NO to the REPEAT question; this option is mainly for cases where a complete set of specifications is already available.

A unique component in the dialog allows you to copy the entire input tape onto the output tape with a minimum of effort. This feature eliminates the need to specify the starting block numbers and number of blocks to copy. In this case, the answer to FIRST INPUT BLOCK: is only a carriage return. The shortened dialog is as follows:

```
TD8E COPY
FROM UNIT: 0
TO UNITS: 1;2;3;4;5;6;7
FIRST INPUT BLOCK:
VERIFY OUTPUT (YES=1, NO=0): 1
0201 12-BIT WORDS PER BLOCK
```

The preceding sample dialog will cause the entire tape on unit 0 to be copied onto the other seven tapes and verified.

13.4.1 Error Messages

TDCOPY produces the following error messages:

ILLEGAL RESPONSE

Your response to the dialog was improper; for example, you typed an alphabetic character or carriage return before a required octal number. The questions will be restated and any previous answer ignored. Type nothing until the terminal has stopped printing.

SELECT ERROR UNIT n

During attempted data transfer, unit n was not found. The program waits for you to correct the cause of the error. Check to see that:

1. when unit n is an output drive, it is set to WRITE ENABLE.
2. unit n is set to REMOTE.
3. there is only one unit n.
4. all units are set to numbers appropriate to their TD8E internal wiring.

When you have corrected the cause of the error, type CTRL/R to resume transfer or CTRL/S to restart the dialog.

TAPE ERROR BLOCK x UNIT n

During attempted transfer, a parity error or timing error was detected, or too great a block number was requested near block x on the tape on unit n. The tapes are rewound and the dialog automatically restarts at DONE, REPEAT (YES=1, NO=0).

VERIFY ERROR BLOCK x UNIT n

The data on the input tape does not match the data written on the block x of the output tape on unit n.

Type CTRL/R to ignore the error and continue with the transfer, CTRL/T to try the last transfer again and continue if the error does not recur, or CTRL/S to restart the dialog.

ILLEGAL FORMAT UNIT n

Indicates one of two situations: Either the number of words per block on unit n does not agree with the number of words per block on the input unit; or when the program calculated the number of blocks on the tape from the block length of the input tape, the length was found to be illegal. The number of blocks is calculated only if you request the whole tape copy option. In either case, when the error has been corrected, type CTRL/R to check the formats of all tapes again and continue, or CTRL/S to restart the dialog.

13.4.2 Details of Operation

After the answers to the dialog have been stored, use the following procedure:

1. The number of words per block is determined from the input tape. All output tapes are checked to see if they have the same format as the input tape. If you used the shortened dialog option, the number of blocks on the tape is determined using the formula:

   # of blocks = (636,160/(words per block + 17)) + 2 (octal)

   or

   # of blocks = (212,080/(words per block + 15)) + 2 (decimal)

2. The response to the VERIFY question is checked. The copying loop is set up to verify or not, as was requested.

3. The loop that copies the input tape is entered, using the same set of specifications for each output tape.

   a. The buffers are filled from the input tape.
   b. All output tapes are written with the contents of the buffers.
   c. If verification was requested, a separate set of buffers is filled from the output tape and the two sets of buffers are compared. If there are any discrepancies a VERIFY ERROR has occurred.
   d. If more blocks remain to be copied, the loop is entered again.

4. When all the specified blocks have been copied onto the output tapes, all the tapes are rewound.

5. The REPEAT option is offered.

The number of fields to be used for buffer space is determined immediately after loading. As soon as you have answered the question, it is removed from the program.

If you verify the output tape, each available field, including that part of field 0 not occupied by the program, is divided in half. The lower half is used as the input and output buffer; the upper half is used for verification. The output tape is read back into the upper half and the two halves are compared. If they are not identical, a VERIFY ERROR has occurred.

CHAPTER 14

DUMP

The DUMP handler is a new OS/8 2-page handler that obtains blocks of binary data on file-structured devices and sends them to the LP08 line printer to produce a listing. This listing is called a DUMP.

Format:

```
.COPY DUMP:<dev:filename.ex
```

or

```
.R PIP
*DUMP:<dev:filename.ex/I
```

Example:

```
.COPY DUMP:<SYS:FL2
```

After typing the command line, followed by a carriage return, type the initial block number of the area in the specified file you want dumped. This automatically dumps block number 0000 of the file. In addition, the DUMP routine skips to the block number specified and dumps it and any block numbers greater than it.

Because the DUMP handler contains a routine that interacts with the keyboard monitor, you can change the block number previously entered by typing a new block number on the keyboard. When you type a new block number, the current block number is dumped before the new block number takes effect.

If you enter a carriage return after the command line and do not supply a block number, the DUMP handler starts at block number 0000 of the file and dumps all the remaining block numbers in the specified file.

Each block of data (2 memory pages) sent to the LP08 line printer results in a printed page of data followed by a form feed. If an uneven number of pages is sent to the line printer during the DUMP operation, the odd numbered page printed on the line printer will show only half a block (one memory page) of data.

If you type an illegal character (excluding 0-7, carriage return, and CTRL/C) while entering the block number, a question mark (?) echoes on the terminal. Any digits typed before it are ignored, and you can type in a new block number. If you type CTRL/C while the DUMP handler is running, control returns to the keyboard monitor.

In addition to the CCL format shown using the COPY command, there is a -D option. When specified, this option forces the output device to be DUMP:. You can use this option with any CCL command.

14.1 FORM FEEDS

A form feed on the LP08 line printer occurs before block 0 data is sent to the handler and after the handler is called to do a close (page count of 0).

14.2 ADDING THE DUMP HANDLER TO YOUR SYSTEM

You can add the DUMP handler to your system through the BUILD program. Its group name, as well as its entry point name, is DUMP; and the current version of the handler is A. This handler does not directly interact with the keyboard monitor, but rather contains a routine that performs that function. It is a 2-page handler and it has no coresident handler. The keyboard monitor runs completely overlapped with the LP08 handler.

14.3 FORMAT OF THE DUMP

The top left of every printed page in the DUMP listing has a 4-digit octal number. This number is the relative file block number of the data that is printed on that page. The first column of 4-digit octal numbers represents line numbers. Each line number is followed by a slash (/), which distinguishes the line number from the remaining eight columns. The remaining eight columns represent the actual data words located within a specific block in a file. The next column containing 16 characters is a representation of the eight data words on that line. Two 6-bit characters are packed in one word (that is, two ASCII characters represent each data word).

The last column containing 12 characters is another representation of the eight data words on that line. Three 8-bit characters are packed in two words; that is, every two data words are represented by three ASCII characters. Some of the spaces in this column could represent non-printable characters. Any character that is not on the line printer can be referred to as a non-printable character.

The following listing is an example of a single printed page from a DUMP listing.

```
0004
0000/  7733  2213  0000  1720  7777  1322  0506  6300  ?[KK@@OP??KREF3@   [    P   R F@
0001/  1501  6264  7653  1322  0506  6300  2202  6264  MA24>+KREF3@R824   A4<+R F@  4L
0002/  7767  1420  2326  0000  0216  8304  7777  2213  ?7LPSV@@BN3D??RK      V @ D
0003/  0502  1404  2326  6354  7737  0425  1520  0000  EBLDSV3,?.DUMP@@   B   V L   P 0
0004/  0215  6314  7776  0681  0000  0000  1501  6314  BN3L?>F1@@@@MA3L    L   1     AL<
0005/  7777  0662  0000  0000  1501  6314  7777  0664  ??F2@@@@MA3L??F4    2     AL< 4
0006/  0000  0000  1501  6314  7777  0425  1520  0000  @@@@MA3L??DUMP@@      AL<   P 0
0007/  2001  6304  7761  2425  1520  6200  2001  6314  PA3D?1DUMP2@PA3L    DL    P  < LL
0010/  7762  0663  0000  0000  1501  6314  7777  0665  ?2F3@@@@MA3L??F5    3     AL< 5
0011/  0000  0000  1521  6314  7777  0363  0000  0000  @@@@MA3L??C3@@@@      AL<
0012/  1601  6314  7777  0361  0000  0000  1501  6314  MA3L??C1@@@@MA3L   AL<        AL<
0013/  7777  0362  0000  0000  1501  6314  7777  0364  ??C2@@@@MA3L??C4          AL<
0014/  0000  0000  1501  6314  7777  2262  0000  0000  @@@@MA3L??R2@@@@      AL< 2
0015/  1501  6314  7777  2261  0000  0000  1501  6314  MA3L??R1@@@@MA3L   AL<  1     AL<
0016/  7777  2263  0000  0000  1501  6314  7777  0365  ??K3@@@@MA3L??C5    3     AL<
0017/  0000  0000  1501  6314  7777  0425  1520  6300  @@@@MA3L??DUMP3@      AL<   P@<
0020/  2001  6324  7762  0425  1520  6300  0216  6324  PA3T?2DUMP3@BN3T    TL    P@<  T
0021/  7776  2266  8070  1623  0216  6324  7777  0425  ?>RF08NSBN3T??DU      8 C T
0022/  1520  6400  0216  6324  7776  0425  1520  0000  MP4@BN3T?>DUMP@@   P  =  T     P 0
0023/  2320  6334  7774  0425  1520  6500  2001  6334  SV3\?<DUMP5@PA3\    V\L    P@=  \ L
0024/  7762  0425  1520  6600  2001  6354  7761  0425  ?2DUMP6@PA3,?1DU      P =   L
0025/  1520  6500  0216  6354  7776  0425  1520  6400  MP6@BN3,?>DUMP4@   P  =        P  =
0026/  2001  6324  7762  0425  1520  6500  0216  6334  PA3T?2DUMP5@BN3\    TL    P@=  \
0027/  7776  1501  0363  6500  0216  6354  7735  1501  ?>MAC35@BN3,?]MA    A   @       ]A
0030/  0363  6500  2001  6354  7400  0617  1700  0000  C35@PA3,<@FOO@@@    @    L    @ 0
0031/  1320  6354  7775  1501  0363  6500  1520  6354  MP3,?=MAC35@MP3,    P  < A  @  P  <
0032/  7744  0000  7252  1501  0363  6500  1423  6354  ?$@@ *MAC35@LS3,       *A  @    <
0033/  5654  0000  7271  6300  0001  6264  7653  1322  2,@@ 93@PA24>+KR    ,  @9@   4 L+R
0034/  0300  6300  0216  6264  7767  3000  0000  0000  EF3@BN24?7X@@@@@    F@   4
0035/  1501  6264  7777  7252  7277  7304  7440  5300  MA24??.* ?;D< +@    A4<  *  ?D    @
0036/  7666  1234  4000  7235  7241  2422  7666  1234  >6J\@.]·!TR>6J\     6       !    6
0037/  4000  7235  7241  2422  4000  7235  7241  2422  @:] 'TR@·]:!TR          !         '
```

CHAPTER 15

EPIC

EPIC, the Edit, Punch and Compare utility program for OS/8, performs the following functions:

- Read and punch tape files and patches
- Edit arbitrary files
- Compare files in any format

These functions are discussed in the next few pages. The discussion assumes you have an elementary knowledge of OS/8.

15.1 LOADING EPIC

To load the EPIC program, type:

```
.R EPIC
```

in response to the Keyboard Monitor dot. Specify the EPIC function you want by including one of the following numeric options in the file command line:

```
.R EPIC                    0 paper tape
*TRANS.AS</0$              1 edit
                           2 compare
```

punch the file TRANS stored on SYS.

```
.R EPIC
*DTA1:FILEA.SV</1$
```

fetch FILEA from DTA1 for editing

```
.R EPIC
*DSK:ABC.SV<DTA1:XYZ.SV/2$
```

compare file ABC on the disk with file XYZ on DTA1 and output block numbers and locations of each non-match on the Teletype.

After you have included one of these numeric options in a command, you do not have to specify it again in subsequent sequential commands requiring the same option. Specifying the number puts EPIC in a mode in which it remains until another number is specified. Initially, EPIC is set to option 0. You use the character ALTMODE, which appears as a $ on the terminal, to end a command that includes a numeric option.

15.2 RESTART PROCEDURE

You can restart EPIC at location 0200. Default options remain active. The default options are discussed later in this section.

15.3 PAPER TAPE FACILITY

EPIC's paper tape option (/0) punches OS/8 files and file patches onto paper tape, and creates OS/8 files from paper tapes. Whole files or patches (blocks) of files can be read or punched. Parity checks are punched to ensure accurate reads. Because of the paper tape format used, tapes must be both punched and read by EPIC. A file punched by PIP, for example, is not acceptable to EPIC.

15.4 COMMAND FORMAT

To request the paper tape facility, specify option 0. Your response to the command decoder's * determines whether a tape is to be punched or read. In both cases, no input files or devices are specified. To punch a tape, specify the file name; to read a tape, you need not enter a file name (that information is encoded on the paper tape). The command line specifying the mode of EPIC is terminated by ALTMODE.

To punch a tape, respond with:

```
*dev:name</0/other options$
```

To read a tape, respond with:

```
*dev:</0/other options$
```

If a file name is specified, EPIC looks up the name on the specified device and punches the file (including the file name) onto paper tape. If no file name is specified, EPIC reads in a paper tape and enters it onto the output device under the name it read in from the tape.

The other options for handling paper tape are:

/L	Use low speed paper tape reader or punch
/E	Do not punch end of tape upon completion
/P	Punch or read a patch (instead of the whole file)
/Z	Set relative block to 0
/=n	Punch relative block n
/Y	Clear default name

You can combine these options to achieve the desired results:

/L	If the /L option is not specified, EPIC assumes a high-speed paper tape device. Thus, SYS:</0 means read a tape from the high-speed reader to device SYS, but SYS:</0/L means read it from the low-speed device.

/E — The /E option can be used to punch a series of patches to a file for all patches except the last one. With the /E option the end of tape mark is not punched. The end of tape must have the "end of tape" punch, a 377 punch and a length of leader/trailer tape.

/P — The /P option is required to indicate the tape to be read or punched is a patch, not an entire file. Generally, the command required to read in a patch is simply dev:</P. File name and block specification are already punched on the tape.

Option /Z or =n must be used with the /P option to indicate punching block 0 or some other block (relative block n), respectively. The patch is read on top of an existing file on the specified output device, that is, modifying an old file, not creating a new one.

/Y — The /Y option is used to clear the default file name when switching from punching to reading paper tape and when reading more than one paper tape.

15.5 DEFAULT OPTIONS

Throughout EPIC, if you do not specify options, files, or devices, the program defaults to the last such item specified. There is an initial default device: SYS is assumed if no output device is specified. No options are assumed initially, except for relative block 0. Note that device and file name options carry between EPIC modes 0, 1 and 2. Specifying an option (that is, L, P, E, Z, etc.) in a command string disables default to any options from the previous command (except 0, 1, 2).

For example, to punch blocks 0, 1 and 30 of the file TRANS on the SYS device and read them back onto that file on DTA3, the commands are:

Punch block 0 of TRANS on high-speed punch with no end of tape punch. Because EPIC defaults to the paper-tape option initially, 0 is not required in this case.

Punch block 1 of file TRANS with no end-of-tape character on high-speed device.

```
.R EPIC
*TRANS</P/E/Z$
*=1
*=30/P
*DTA3:/Y
```

Punch block 30 of the file TRANS on high-speed punch. Punch end of tape (P disables E).

Read the tape from the high speed device and put out to file whose name is encoded in the patch on device DTA3 until end of tape is reached. File name and relative block are punched on the tape so this information is not necessary. Y clears the default name (TRANS).

15.6 ERROR CONDITIONS

If an error occurs while reading a block of paper tape, EPIC outputs an error message (the error messages are listed at the end of this section), and halts. You should reposition the paper tape to the leader/trailer just in front of the block just read before continuing (refer to Section 15.12, Paper Tape Format). Three consecutive read errors terminate the command. When EPIC is reading in a non-patch file it checks the initial block read of every tape and block that is reread because of error. This is done to determine if the read was accurate up to name and block number. If the wrong block number or file name is read, EPIC outputs an appropriate message indicating the type of error. It then halts with AC=7777 to allow you to reposition the tape over the correct block or to enter the correct tape before continuing.

15.7 LOW SPEED I/O

The execution of EPIC differs for low-speed I/O. Before starting a low-speed punch, EPIC halts with 7777 in the AC to allow you to turn on the low-speed punch and then press the CONT key on the computer console. Upon completion of a punch command, EPIC halts with the AC=0 to allow you to turn off the punch. When you press the CONT key, EPIC recalls the command decoder. For low-speed input EPIC halts only upon completion of the read.

If a file or a series of files to be punched exceeds 32 blocks, EPIC segments it by punching end of tape after 32 blocks. This end-of-tape punch, done automatically and independently of the E option, keeps tapes short enough to fit into a paper tape tray. Upon physical end of tape, EPIC halts with the AC=0 if the low-speed punch is being used. This is done to allow you to turn off the punch before continuing. As soon as the punch is turned off, EPIC outputs the message END OF TAPE ENTER NEXT and then halts with the AC=7777 to allow both high- and low-speed users to remove the paper tape. Note that low-speed users get both halts, but high-speed users get only the 7777 halt. In general, a halt with AC=0 means to turn paper tape device off, and a halt with AC=7777 means to turn device on. All halts are terminated by depressing the console CONTinue key. If EPIC encounters end of tape while reading a non-patch file, it outputs the message END OF TAPE ENTER NEXT and halts with AC=7777. This indicates that the file is segmented across a number of tapes and that you should enter the next tape.

15.8 DEVICE CODES

Most of the execution time is spent waiting for paper tape devices. During I/O wait, EPIC holds the device code and version number in the AC. The device code is in bits 3-5 and the version number is in bits 6-11. The codes are as follows:

1 high-speed reader
2 high-speed punch
3 low-speed reader (console TTY)
4 low-speed punch (console TTY)

If you forget to turn on the high-speed reader, EPIC hangs with 1xx in the AC. You can always restart EPIC at 0200. The OS/8 CTRL/C is normally in effect; the exceptions are when EPIC is waiting for a paper tape device or when input is from the low-speed reader.

NOTE

When input is from the low-speed reader, EPIC forces the output device to be SYS because it is the only OS/8 I/O handler that does not check for CTRL/C.

Thus, if you were to enter the command:

```
.DTA2:</L
```

EPIC would force it to be

```
.SYS:</L
```

15.9 EDITING CAPABILITY

Option 1 of EPIC is the file editing and searching facility. With this feature, you can add patches directly to the file by specifying relative blocks and locations in the file.

15.9.1 Initial Command Format

The general format of a command for the editing option is:

```
.R EPIC
*DEV:NAME</OPTIONS/1$
```

The /1$ specifies edit mode for EPIC.

As with the paper tape option, default conditions apply. If you have not specified a device and/or file name, the last one mentioned is used. When editing, the only option available in the initial command is:

/Y Clear default name (if one exists)

Editing is performed one block at a time. The relative block you are currently processing is known as the current block; the location you are currently processing is known as the current location (0-377). Relative block 0 is the first block of the file if a file name is specified or block 0 of the device if no file name is specified.

15.9.2 Editing Commands

After the initial (file specification) command, you use a series of keyboard commands to perform the editing. The general format of an editing command is:

x

or

x,n1,n2

where x is a command letter and n1,n2 are octal numeric arguments. If you use a numeric argument, the letter is followed by a comma. You

can type up to 32^10 characters on a line. Default conditions apply to these commands as well. If carriage return is the only character typed as an editing command, the last command specified is executed. The commands are listed and explained in Table 15-1.

Table 15-1
EPIC Commands

Command	Meaning
E	Exit to command decoder; write out current block of file if it has been modified.
R, n	Read relative block n (octal) of file and set current location to 0. Do not write current block. If n is not specified, the current block is read. If the relative block is out of range, a ? is printed. There are 1341 blocks per OS/8 tape and 6260 per RK8 disk platter.
W	Write the current block of file if it has been modified and read in the next sequential block of the file. If the current block is the last block of the file, a ? is printed and the current location is unmodified.
S, n1, n2	Search the current block for the value n1 with the mask n2. If either n1 or n2 or both are omitted, the last value specified is used. The initial mask is 7777. Masking is performed in a logical AND fashion. If the S command is terminated by the RETURN key, the search is for the current block only. If the S command is terminated by the LINE FEED key, the search continues to the end of the file. If the search fails (either in the block or to the end of the file), EPIC prints a ?. If the search is successful, EPIC prints: m1 m2 m3 / where m1 is the relative block, m2 is the relative location within the block, and m3 is the contents of the location (m1 is omitted if a previous match was found in the same block). To change the contents, type the new contents (in octal) after the slash. To continue the search, type the LINE FEED key; to terminate the search, type the RETURN key. (If the contents are not to be changed, type one of the terminators.)

(continued on next page)

Table 15-1 (Cont.)
EPIC Commands

Command	Meaning
O, n	Open location n of the current block. If n is not specified, the last opened location is the default. If there is no default, location 0 is opened. EPIC responds with m1 / which is the contents of location n. This location may be modified as in search. Terminating with the LINE FEED key closes the current location and opens the next. If the current location is the last one in the block, location 0 of the next block is opened, and the current block is written out as if it had been modified.
C	Print current status, as: m1 (F or B) m2 m3 m4 where m1 is the current block, m2 is the current location, m3 is the search word, and m4 is the mask word. If you type F, the file has been modified since option 1 was requested; B indicates the current block has been modified. Once a modified block has been written to the file, the F is the only code output.

Thus a reasonable sequence is:

```
.R EPIC                          Call EPIC.
*DSK:ISOMER</1$                  Edit file ISOMER on DSK.
R,2                              Read block 2.
S,3126,7770                      Search for a 312x in that block.
?                                Not there.
>                                Search for it throughout the file.
0004 0110                        Found at block 4, location 110.
3124 /3121                       Change contents to 3121.
,,7777                           Search for 31xx throughout the rest
                                 of the block (locations 110-377).
0004 0132                        Found at location 132 of block 4.
3126 / 3127                      Contains 3126. Change to 3127.
C                                Check status.
0004 B 0132 3126 7777            At location 132 of block 4 which
                                 has been modified; the current
                                 search word is 3126 and mask is
                                 7777.
W                                Write block 4.
?                                Block 4 written but file is only
                                 four blocks long, no block 5 to
                                 read.
R,2                              Read block 2.
O, 10                            Open location 10.
1367 /1364                       Contains 1367. Change to 1364.
3324 .                           Check next location. No
                                 modifications.
E                                Exit editing option.

*
```

15.10 COMPARE CAPABILITY

A third feature of EPIC is file compare (/2). Because EPIC uses an absolute compare technique, there are no limitations in the data format or the length of the file. The files you are comparing must reside on the system device.

COMMAND FORMAT

Option 2 of EPIC requires only one command, specified as follows:

```
SYS:file1<SYS:file2/options/2$
```

Specify the first file to be compared to the left of the angle bracket, the second file to the right. The options are:

/A Abort when the first non-match is found.

/B List physical block number for each file where a non-match exists.

If you specify no options, the block numbers and locations of each non-match are listed on the terminal.

For example, to compare files PYTHG1 and PYTHG2 and find all unequal locations, the sequence is as follows:

```
*SYS:PYTHG1<SYS:PYTHG2/2$
SYS:0174  SYS:0631
0152 7450 3421
0153 5741 2021
0154 3421 3022
*
```

To compare them and list unequal blocks, the command is:

```
*SYS:PYTHG1SYS:PYTHG2/B/2$
```

If this block match followed the preceding locations match command, a sufficient command and its results are:

```
*/B
SYS:0174  SYS:0631
```

To abort after the first non-match, use the sequence:

```
*/A
SYS:0174  SYS:0631
```

15.11 ERROR MESSAGES

EPIC can print certain error messages when performing paper tape (option 0) operations. (See Table 15-2.)

Table 15-2
EPIC Error Messages

Message	Explanation
BAD =BLK	When EPIC is punching a patch, it checks the block specified by "=n" to see if it is within range. If the block is out of range, EPIC outputs this error message and returns to the command decoder. For example, if a file JOE were two blocks long and you requested: SYS:JOE</P=3 the error message would be printed.
END OF TAPE	EPIC was expecting a block of tape and found end of tape instead. EPIC halts with AC=7777 to allow you to reposition the tape. When you depress the CONT key, EPIC attempts to read the block.
END OF TAPE ENTER NEXT	When EPIC is reading a file that is segmented across a number of paper tapes and encounters the end of a segment, it outputs this message and halts with AC=7777 to allow you to enter the next segment of paper tape. Press the CONT key to continue reading.
I/O ERROR	If EPIC encounters an error while reading or writing a mass storage device, or if a paper tape read fails three consecutive times, EPIC outputs this error message, deletes the output file if one exists, and returns to the command decoder.
L/T ERROR	EPIC was expecting leader trailer and found non-leader trailer while attempting to read a block. The program prints this error message and halts with AC=7777 to allow you to reposition the tape, then press the CONT key.
NEED:name1 FOUND name2	EPIC read a block of tape for the file NAME2 when it was expecting a block of the file NAME1. This error typically occurs when you come to the end of a segment for NAME1 and enter some segment of NAME2 instead of the next segment for NAME1. EPIC halts with AC=7777 to allow you to enter the correct paper tape.
NEED:n1FOUND:n2	EPIC read block n2 of the file when it was expecting block n1 of the file. EPIC halts with AC=7777 to allow you to reposition the paper tape. This error typically occurs when you reposition the tape to the wrong block after a read error.

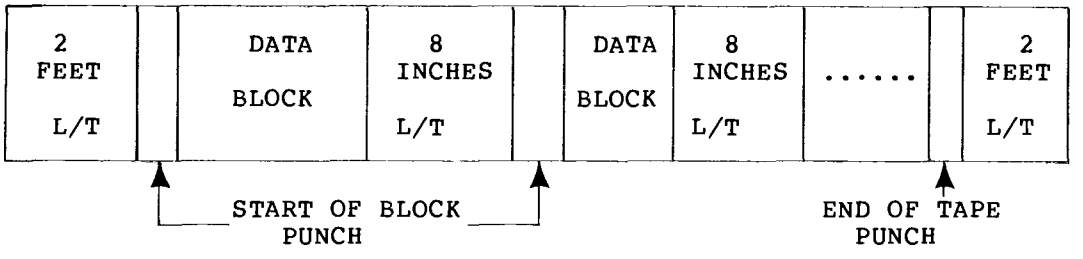
(continued on next page)

Table 15-2 (Cont.)
EPIC Error Messages

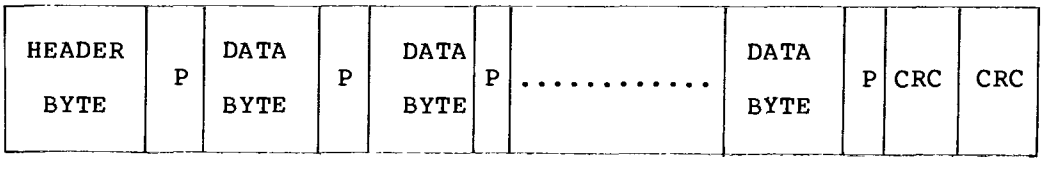
Message	Explanation
PARITY ERROR	EPIC failed to read a block correctly, for example, the reader dropped some bits. EPIC halts with AC=7777 to allow you to reposition the tape so that it can try the read again.
PTR:NAME IS TOO BIG FOR dev:	The paper tape file NAME will not fit on the specified output device DEV:. EPIC aborts the command and returns to the command decoder. EPIC makes the check for size before writing on the output device.
USR n dev:name	The USR encountered an error while attempting to perform a fetch, lookup, enter, or close on the file NAME on device DEV. n=1 is a fetch, n=2 is lookup, n=3 is enter, n=4 is close. EPIC aborts the command and returns to the command decoder. For example, if you request EPIC to punch a file on SYS that does not exist: SYS:NILL EPIC outputs the message USR 0002 SYS:NILL indicating that it could not find the file NILL on the device SYS.

15.12 PAPER TAPE FORMAT

Paper tapes punched by EPIC have the following format:



Leader trailer is any string of 0 or 200 punches; usually it is just 200 punches; leader trailer is terminated by a 201 punch, which indicates the start of a data block. The first punch after the last data block is 377, which is end of tape. Each data block has the following format:



Each byte is 12 punches (96 bits) and corresponds to eight 12-bit words; each byte is followed by an even-odd parity punch of the eight words in the byte. Each block is terminated by two CRC punches of longitudinal parity.

The header byte contains information about the file for example, file name and relative block number. The data bytes constitute the actual data of the block; each 256-word block has 32 data bytes.

15.13 LOADING EPIC FROM PAPER TAPE

If you use receive EPIC on paper-tape, use the following procedure to load the tape and save it on a mass storage device.

```
.R ABSLDR                          Use ABSLDR
*PTR:$^                            Read from reader; after ^ is
                                   output, type any key to start
                                   reader
.SA SYS EPIC 0-7577;0200=0         Save on mass storage with starting
                                   address of 200
```

15.14 EPIC ASSEMBLY INSTRUCTIONS

Use the PAL8 (version 9) assembler to assemble EPIC as follows:

```
.R PAL8
*DEV:EPIC.BN,DEV:EPIC.LS<DEV:EPIC.PA
```

To create the save file, use ABSLDR:

```
.R ABSLDR                          Call ABSLDR.
*DEV:EPIC.BN$                      Load EPIC.BN on device specified.
.SA DEV EPIC 0-7577;0200=0         Save EPIC on device specified.
                                   0-7577 = area in core used during
                                   execution. 0200 = restart address.
```

CHAPTER 16

FILE-ORIENTED TRANSFER PROGRAM (FOTP)

FOTP transfers files from one device to another, deletes files from a device, and renames files. FOTP is significantly faster than PIP and performs certain functions not available with PIP. For example, FOTP can transfer files longer than 256 blocks and can perform multiple file transfers and deletions without requiring multiple accesses of the directory.

FOTP copies files in the image mode, that is, it copies the file word for word, character for character, without making any changes in the file. (This corresponds to the /I option in PIP.) Thus you may use FOTP to copy core image and binary files as well as ASCII files, without specifying options to identify the type of file.

16.1 CALLING FOTP

To call FOTP from the system device, type:

```
.R FOTP
```

in response to the Keyboard Monitor dot. (You can also call FOTP indirectly with several CCL commands. See the CCL section of Chapter 1.) The Command Decoder prints an asterisk in the left margin and waits to receive a line of I/O files and options. FOTP accepts one output specification and up to five input specifications. The I/O specification line may be terminated with a carriage return (FOTP retains control) or with an ALTMODE (control returns to the Keyboard Monitor).

16.1.1 Input Specifications

FOTP input specifications consist of a device, a file name, and a file name extension. Input specifications are optional but must be present if you do not include an output specification.

Within the input specification, FOTP allows you to use a wild card construction. This means that the file name or the extension may be replaced totally with an asterisk or partially with a question mark to designate certain file names or extensions. You can use the asterisk as a wild field to designate the entire file name or extension. For example:

TEST1.*	All files with the name TEST1 and any extension.
*.BN	All files with a BN extension and any file name.
.	All files.

The question mark serves as a wild character to designate part of the file name or extension. Use a question mark for each character you want to match; for example, PR?? matches on four characters or fewer. Some examples follow.

TEST2.B?	All files with the name TEST2 and any extension beginning with B.
TES??.PA	All files with a PA extension and any file name up to five characters beginning with TES.
??.??	All files with file names of two characters or less.

You can specify the asterisk and the question mark together in the same command line.

???.*	All files with file names of three characters or less.

The following are examples of legal FOTP input specifications:

```
DSK:
SYS:A
LTA3:TEST1A
DTA7:A.BN
FILE
FILE3.DA
4
NAME?.TX,NAM??.BN
N?ME.
??????.D?
*
*.BN
PRN:*.??
?W?B?Z.?A
```

A specification may not contain embedded asterisks; for example, A*B.* is an illegal specification. The following are illegal input specifications:

```
A,B,C
A:B:C
A?*.B
.AB
DAT:A.*B
A?B:C
*:BIN
```

If no device is explicitly given for an input specification, the device associated with the previous specification is assumed. If no device is explicitly given for the first specification, the DSK: is assumed. Thus, the following input specifications are equivalent:

```
DSK:B                    B
SYS:B.*,C.*,D.*          SYS:B.*,SYS:C.*,SYS:D.*
B.*,DTA0:,SYS:*.BN       DK:B.*,DTA0:,SYS:*.BN
```

You can include as many as five input specifications in a single command line. If all the files are on the same device, the input device need be specified only once. For example:

```
DTA0:*.BN,*.SV,*.RL
```

refers to files on DTA0 that have .BN, .SV, or .RL extensions with any file name.

16.1.2 Output Specifications

FOTP output specifications consist of a device, a file name, and a file extension. Output specifications are optional. You can use the wild card asterisk in output specifications, but use of the question mark is illegal.

If no output device is specified but a file name is given, then DSK: is assumed. If no file name is specified, then *.* is assumed. Thus the following output specifications are equivalent:

A	DSK:A
A.*	DSK:A.*
DTA3:	DTA3:*.*

16.2 USING FOTP

Since FOTP performs file transfers in a different manner than other OS/8 transfer programs, the following is a detailed description of the way FOTP works. One of the main uses of FOTP is to copy files from one device to another. The following examples show how FOTP examines each aspect of a command to determine what operation will take place.

Example 1:

To copy the file SMILE.PA from DTA3 to DTA5, changing its name to FROWN.PA, type the following command in response to the Command Decoder *:

```
*DTA5:FROWN.PA<DTA3:SMILE.PA
```

1. If FOTP does not find the file SMILE.PA on DTA3, the following message appears and no transfer is made:

   ```
   *NO FILES OF THE FORM SMILE.PA
   ```

2. FOTP examines DTA5 to determine whether it already contains a file FROWN.PA. If FROWN.PA is already on DTA5, FOTP deletes it before beginning the transfer. This process is known as predeletion.

3. The /N option specifies that no predeletion is desired. Thus the command:

   ```
   *DTA5:FROWN.PA<DTA3:SMILE.PA/N
   ```

   begins to copy SMILE.PA to DTA5 without deleting the old FROWN.PA. FOTP does this by opening a tentative file named FROWN.PA on DTA5. When the command completes the transfer operation, it closes the tentative file. Closing this tentative file makes it a permanent file and deletes any old files of the same name. This process is called postdeletion.

4. FOTP assigns the creation date of SMILE.PA to FROWN.PA. This is an advantage over PIP, which would assign the current date to the new file. If you always transfer files with FOTP, you preserve the original creation date of the file. Thus this feature of FOTP allows you to differentiate between versions of a file since the more recent version should have a later date.

5. Use the /T option of FOTP to assign the current date to a file. For example, if SMILE.PA is undated, FOTP assigns the current date to the newly created FROWN.PA.

```
*DTA5:FROWN.PA<DTA3:SMILE.PA/T
```

6. You may be using the additional information words feature of OS/8. This feature allows you to associate additional information (other than the creation date) with each file entry in a device directory. FOTP transfers such additional information words from SMILE.PA to FROWN.PA. (PIP does not perform this function.)

   If the file structure on DTA5 has space for more information words than with SMILE.PA, then those extra words are set to 0.

   If the file structure on DTA5 does not have enough space for all the additional information words associated with SMILE.PA, then FROWN.PA is given as many as can fit (from the left). Excess information words (on the right) are not transferred.

Example 2:

Normally, you copy files from one device to another without changing the file name. For example, to copy the file TEST.PA from DTA1 to DTA2, type:

```
*DTA2:TEST.PA<DTA1:TEST.PA
```

in response to the asterisk printed by the Command Decoder. Since this transfer operation is so common, FOTP allows the output file name to be abbreviated to *.*. The *.* means that you use the input file name as the output file name. Thus you could type the preceding command as:

```
*DTA2:*.*<DTA1:TEST.PA
```

Since the *.* specification is so frequently used, it is the default, that is, if no output file name is specified, *.* is assumed. So you can further simplify the preceding command to:

```
*DTA2:<DTA1:TEST.PA
```

Example 3:

One feature of FOTP allows you to use the same command line to transfer multiple files from one device to another. For example, to transfer five FORTRAN source files from SYS to RKA2, type:

```
*RKA2:*.*<SYS:DATA1.FT,DATA2.FT,DATA3.FT,DATA4.FT,DATA5.FT
```

The wild card characters * and ?, explained previously, are particularly useful when doing multiple file transfers. For example, to transfer all FORTRAN II source files from SYS to RKA2, type:

```
*RKA2:*.*<SYS:*.FT
```

The specification *.FT stands for files with any name that have the .FT extension.

To copy all files from DTA1 to DSK, type:

```
*DSK:*.*<DTA1:*.*
```

Note that the *.* specification has a different meaning when it appears on the left side of the < than it does when it appears on the right. When used on the output (left) side, *.* means that the output file name is the same as the input file name. When used on the input (right) side, *.* means transfer or consider all files on this device. For example:

```
*RKA2:<SYS:TEST1.PA,TEST2.PA,TEST3.PA
```

copies three files from SYS to RKA2. PIP would require three commands to perform the same operation. Each command transfers one file.

In the preceding example, no output file name is specified, so *.* is assumed. No device is specified for the files TEST2.PA and TEST3.PA, so the device specified as the previous input device (SYS) is assumed.

Frequently, you will copy several files with similar names (as above) from one device to another. In many cases, you can reference these files by a single file specification, using the ? wild character. For example, the command:

```
*DTA2:*.*<DTA1:TEST?.PA
```

transfers all files on DTA1 that have the extension .PA and that have names beginning with TEST followed by one other character.

16.2.1 Additional FOTP Commands

Here are some additional FOTP commands that you may find useful.

To transfer the file X.Y from disk to DECtape:

```
*DTA0:<X.Y
```

To transfer the files A, B, C, D and E from SYS: to DTA3:

```
*DTA3<SYS:A,B,C,D,E
```

To transfer all FORTRAN source files from one DECtape to another, producing a log of those copied:

```
*DTA2:<DTA5:*.FT/L
```

To list all FORTRAN and BASIC files on the line printer in order of appearance on DSK:

```
*LPT:<*.FT,*.BA
```

To list all FORTRAN and BASIC files on the line printer, listing all FORTRAN files before all BASIC files:

```
*LPT:<*.FT,*.BA/U
```

To copy all files other than .SV and .BN files from DTA3: to DSK:, then copy all files other than those whose names begin with a K from DTA2: to DSK:. Log all files copied:

```
*DSK:<DTA3:*.SV,*.BN,DTA2:K?????.*/V/L
```

To copy the file A.B from DSK: to DTA1:, changing its name to C.D. Give the new file today's date:

```
*DTA1:C.D<A.B/T
```

To copy all files from LTA2: that have the extension .PA to SYS:, changing the extension to .PL and allocating storage on SYS: without doing predeletions:

```
*SYS:*.PL<LTA2:*.PA/N
```

To find all files on RKA2: with the name FOO and any extension but those that have today's date, and copy them to SYS:, changing the file name to WXYZ yet keeping the extension:

```
*SYS:WXYZ.*<RKA2:FOO.*/C
```

To delete all disk files (except those with today's date) that either have the extension .LS, .TM, or .BK and those whose file name begins with TMP:

```
*DSK:<*.LS,*.TM,*.BK,TMP???.*/D/O
```

To delete each .BN file for which there is a corresponding .PA file:

```
**.BN<*.PA/D
```

To delete all .LS files on DTA3: for which there is a file on RKA0: with the same name but with an extension of either .PA or .RA or with no extension:

```
*DTA3:*.LS<RKA0:*.PA,*.RA,*/D)
```

To delete all files on the disk for which there are already copies on one of the four DECtape drives:

```
*DSK:<DTA0:*.*,DTA1:*.*,DTA2:*.*,DTA3:*.*/D
```

To produce a log of all files on DTA1: that have the file name FOO and an extension which is the same as any file on SYS: that has a one or two-character file name beginning with a "T". Do not perform any transfers or deletions:

```
*DTA1:FOO.*<SYS:T?.*/N/D/L
```

To change the name of the file DSK:FILE.PA to FILE2.PA:

```
*FILE2.PA<FILE.PA/R
```

To rename all files on DTA6: with a .PA extension to a .PB extension:

```
*DTA6:*.PB<DTA6:*.PA/R
```

To change the extension from .RL to .OL of all files on DTA1: that correspond to files on DSK: with the same name and today's date:

```
*DTA1:*.OL<*,RL/C/R
```

16.2.2 Advantages of Predeletion

The default mode (and the recommended one) of FOTP is the use of predeletion when copying files. Predeletion creates space on the output device for the new file. Suppose that, in Example 1 above (Section 16.2), DTA5 were almost full. There might not be enough space on DTA5 for SMILE.PA. Deleting FROWN.PA first could create enough space for SMILE.PA.

Predeletion normally places the new file in the space occupied by the file you are replacing. In Example 1 above, if FROWN.PA is first deleted, the space where it resided is empty. You could then use this empty space for the new copy of FROWN.PA (the former SMILE.PA). If you did not use predeletion, the new tentative file for FROWN.PA would probably be placed at the end of the tape. This procedure would create a gap (EMPTY) when the old copy of FROWN.PA was deleted; thus the files on DTA5 would be ordered differently.

16.2.3 Advantages of Postdeletion

Postdeletion is a slightly safer method of transferring files because you do not delete the original file until you complete a transfer. Suppose that, in Example 1 above, SMILE.PA is an updated version of the FROWN.PA, existing on DTA5, and that these are the only two copies of a certain source file. If you perform predeletion and SMILE.PA is discovered to have a permanent input error, that source file will have ceased to exist because SMILE.PA will be unreadable and FROWN.PA will have been deleted. The use of postdeletion in this case would save the original copy (FROWN.PA) even though the updated version (SMILE.PA) could not be read.

16.2.4 Control Characters

You can use the special characters CTRL/C and CTRL/P to terminate FOTP operations. When you type CTRL/C, FOTP continues operation until the files on the output device are the same as those in the output device directory. Control then returns to the OS/8 Keyboard Monitor.

CTRL/P causes FOTP to terminate the current operation but to still retain control. The output device directory is updated to reflect the operations completed before the termination occurred. FOTP prints an asterisk and can receive another I/O specification line.

If you type CTRL/C or CTRL/P when deleting (/D) or renaming (/R), no FOTP operations are performed and the following message appears:

```
ORIGINAL DIRECTORY PRESERVED
```

16.3 FOTP OPTIONS

The options listed in Table 16-1 may be used in a FOTP specification line.

Table 16-1
FOTP Options

Option	Meaning
/C	Current date. Consider only those input files with the current date when performing a FOTP operation. For example, if you type the command: *DSK:<DTA0:*.*/C FOTP transfers from DTA0 to DSK only those input files that have the current date.
/D	Do not perform any I/O transfers, that is, perform only deletions. /D is not an abbreviation for delete although it usually performs that operation. This option compares the input specification with the output specification, if any, for matching files. If a match is made, FOTP performs as though transferring the file, and then deletes the transferred file. If no transfer occurs, no postdeletion occurs. Predeletion might still occur unless you include the /N option. If you specify no output device, FOTP assumes the first input device specified as the output device. If you specify no output files or extensions, that is, *.* is specified or assumed, the input file names become the output file names. If you specify no input files, no deletion takes place.
/F	Failsafe. The /F option protects files during a transfer operation. It is particularly useful when transferring a great number of files from disk to DECtape. The /F option allows you to mount a new volume if a large file will not fit on the output device or if all files will not fit on the output device. If, for example, you wish to transfer all .BN files from DSK to DTA0, type: *DTA0:<DSK:*.BN/F If the output device becomes full before transfer is complete (or if a large file will not fit), FOTP prints: *MOUNT NEXT OUTPUT VOLUME: Dismount the current tape and mount a new tape on the same unit. Type any character to continue. The device mounted must have a good OS/8 directory. FOTP then continues the transfer on the new volume and updates the directories of both volumes.

(continued on next page)

Table 16-1 (Cont.)
FOTP Options

Option	Meaning
/L	List on the terminal the names of files affected during the FOTP operation. Note that neither the device nor the output file is listed.
/N	No predeletion. Delete output file names after a successful I/O transfer occurs. If an I/O transfer proceeds, any other files of the same name will automatically be deleted when the file is closed.
/O	Other than the current date. Consider only those input files with a date other than the current date when performing a FOTP operation.
/Q	Query the user about each relevant file name to determine whether you want the specified operation to occur for that file. This relevant file name could be either an input or output file name depending upon the type of FOTP operation being performed. For example, if you are renaming input files, FOTP prints the affected input file names. If you are deleting output files, FOTP prints the output files that will be affected. FOTP prints each relevant file name on the terminal and waits for you to respond. A response of Y causes the specified operation to be performed. Any other response causes that file to be ignored, and FOTP prints the next relevant file name.
/R	Rename the output file without performing any transfer. Perform this operation by specifying the same device as both the input and output device. For example: *DSK:TEST3.PA<DSK:TEST2.PA/R would change the name of the DSK file TEST2.PA to TEST3.PA without performing any transfer. (a) The rename option (/R) now looks at the /T switch. If /T is typed, then not only is the file renamed, but the new file receives today's date. Without /T, the new name has the same date as the old name. (b) The rename option (/R) now allows you to rename a file to its own name. This was not previously permitted. It is not very useful unless you include some other switch, for example /T.

(continued on next page)

Table 16-1 (Cont.)
FOTP Options

Option	Meaning
/R (cont.)	(c) If you have specified no output file with /R, the FOTP assumes the same name as the first input file. Example: To redate all files on a DECtape to Jan. 1, 1976: *DATE 1/1/76 *RENAME DTA0:*.*/T
/T	Assign the current date to the corresponding input file.
/U	Treat each input specification separately. This option causes FOTP to find files in the same order as they are entered in the input specifications. For example, the command: *DTA0:<DSK:TEST.PA,DATA1.FT,TEST2.PA/U/L TEST.PA DATA1.FT TEST2.PA finds the files in the order that they were specified in the command, not in the order in which they may appear on DSK.
/V	Consider only input files which do not have the form specified by the input specifications. For example, the command: *DTA0:<SYS:*.SV,*.HL/V transfers to DTA0 all files on SYS other than those with .SV or .HL extensions.
/W	Print the version number of FOTP on the terminal.

16.3.1 Examples of FOTP Specification Commands

The following are legal command strings to FOTP. When FOTP has completed an operation, control returns to the Command Decoder for additional input, unless you use ALTMODE to terminate the FOTP command line.

Example 4:

```
*DTA0:<A.B
```

This command string transfers the file A.B from the device DSK to DTA0.

Example 5:

```
*DTA3:<SYS:A,B,C,D,E
```

This command string transfers the files A, B, C, D, and E from the system device to DTA3.

Example 6:

```
*DTA2:<DTA5:*.FT/L
```

This command string transfers all FORTRAN source files from DTA5 to DTA2, producing a log of those copied.

Example 7:

```
*LPT:<*.FT,*.BA/U
```

This command string lists all FORTRAN files, then all BASIC files on the line printer.

Example 8:

```
*DSK:<DTA3:*.SV,*.BN,DTA2:K?????.*/V/L
```

This command string copies from DTA3 to DSK all files other than core image (.SV) and binary (.BN). It then copies from DTA2 to DSK all files other than those with names beginning with K. A listing is printed of all files copied.

Example 9:

```
*DTA1:C.D<A.B/T
```

This command copies the file A.B from DSK to DTA1, changing its name to C.D, and assigns the current date to the file.

Example 10:

```
*SYS:*.PL<LTA2:*.PA/N
```

This command copies from LTA2 to the system device all files with .PA extension, changing the extension to .PL.

Example 11:

```
*.LS,*.TM,*.BK,TMP???.*/D/O
```

This command string deletes any disk file that has an extension of .LS, .TM, or .BK or a name beginning with TMP if the file does not have the current date.

16.4 ERROR MESSAGES

The FOTP error messages are listed in Table 16-2.

Table 16-2
FOTP Error Messages

Message	Meaning
ALREADY EXISTS (file name)	An attempt was made to rename an output file with the name of an existing output file.
BAD INPUT DIRECTORY	The directory on the specified input device is not a valid OS/8 device directory.
BAD OUTPUT DEVICE	Self-explanatory. This message usually appears when you specify a non-file-structured device as the output device.
BAD OUTPUT DIRECTORY	The directory on the specified output device is not a valid OS/8 device directory.
DELETES PERFORMED ONLY ON INPUT DEVICE GROUP 1 CAN'T HANDLE MULTIPLE DEVICE DELETES	You specified more than one input device with the /D option when you included no output specification (device or file name).
ERROR ON INPUT DEVICE, SKIPPING (file name)	The file specified is not transferred, but any previous or subsequent files are transferred and indicated in the new directory.
ERROR ON OUTPUT DEVICE, SKIPPING (file name)	The file specified is not transferred, but any previous or subsequent files are transferred and indicated in the new directory.
ERROR READING INPUT DIRECTORY	Self-explanatory.
ERROR READING OUTPUT DIRECTORY	Self-explanatory.
ERROR WRITING OUTPUT DIRECTORY	Self-explanatory.
ILLEGAL *	You entered an asterisk as an embedded character in a file name, for example, TMP*.BN.
ILLEGAL ?	You entered a question mark in an output specification.
NO FILES OF THE FORM xxxx	No files of the form (xxxx) specified were found on the current input device group.

(continued on next page)

Table 16-2 (Cont.)
FOTP Error Messages

Message	Meaning
NO ROOM, SKIPPING (file name)	No space is available on the output device to perform the transfer. Predeletion may already have occurred.
SYSTEM ERROR-CLOSING FILE	Self-explanatory.
USE PIP FOR NON-FILE STRUCTURED DEVICE	An input device specified is not a file-structured device, for example, PTR.

CHAPTER 17

FUTIL

17.1 INTRODUCTION

FUTIL enables you to examine and modify the contents of mass storage devices. It is the only program currently available that you can use to patch programs containing overlays (F4/LOAD outputs). Other possible uses include examination and repair of OS/8 directories; bad block checking and correction; decimal/octal conversion of double precision numbers; output of the Core Control Block (CCB) of .SV files and the HEADER of .LD files; and the creation of special directories. Supporting these functions is signed double-precision arithmetic expression evaluation that you can use in the command syntax whenever you need a numeric value.

FUTIL commands are divided into two groups. The first group uses single letters to direct the program in the examination and modification of single words on the device specified. The second group of commands uses command words to direct the program in the dumping, listing, modifying and searching of the device on a block-by-block basis. Also included in this group is a series of commands to direct the program in some auxiliary functions including setting and resetting switches and variables within the program, showing current FUTIL parameters.

Several examples appear in Section 17.4. The first two examples are especially simple and well-documented and can acquaint you with the features of FUTIL. You may want to look at them at this point to get a better understanding of the material that follows.

17.1.1 Special Characters Used in FUTIL

Several characters, when keyed, cause immediate action from the program. Typing either CTRL/P or CTRL/C will immediately cause the program to stop whatever it is doing. CTRL/P then causes the program to go back to command input mode and wait for you, while CTRL/C returns control to the OS/8 Monitor. CTRL/S and CTRL/Q control program execution (including all I/O). Typing CTRL/S at any time will cause the program to pause and wait for either CTRL/C, CTRL/P or CTRL/Q. Typing CTRL/Q will then allow program execution to resume. Any other characters entered at this point will be simply ignored. If a CTRL/Q is typed by itself at any time, it is simply ignored.

NOTE

CTRL/S and CTRL/Q are active at all times, not just during console output. The result is that both input from the console and program execution with no console interaction (such as SCAN, WORD and STRING command execution) will pause and restart with these keys.

During console terminal input, three other keys help with editing the input string of characters. These keys are RUBOUT, CTRL/U and CTRL/R. The action of RUBOUT and CTRL/U is exactly the same as for the OS/8 Monitor and Command Decoder (including usage of "scope mode" operation to change the action of the RUBOUT key from echoing the rubbed out characters between backslashes to erasing the characters from the screen). The action of CTRL/R is the same as that of the LINE-FEED key for the Monitor and Command Decoder.

If you have upper-lower case terminals, the program translates all lower case characters received from the keyboard to upper case. The characters are echoed and handled internally as upper case characters. While this makes use easier, it does not allow any lower-case characters to be input directly.

In those cases where you need lower-case codes in the modification of a file, either use the codes directly or use a text editor. This translation occurs only on input. Lower-case characters in a file will be printed to the best ability of the output device. This may produce incorrect results on upper-lower case line printers.

All of the commands are taken in context. This means that many of the characters in the single character command set will not be considered to be commands if they are included in a line that begins with a command word or if they are embedded within expressions.

The carriage-return always starts command execution and terminates all word-type command lines.

17.1.2 Running FUTIL

To run FUTIL, type:

```
.R FUTIL
```

or

```
.RU dev:FUTIL
```

When started, FUTIL is set up to access the system device, the ERROR message output mode is set to LONG, the access MODE is set to NORMAL, and no file is known. To access some other device, type:

```
SET DEVICE dev
```

To set the ERROR mode to SHORT, type:

```
SET ERROR SHORT
```

To use some other access mode, type:

SET MODE <mode>

command with a <mode> of LOAD, OFFSET or SAVE. When in OFFSET mode, the OFFSET to be used can be specified by the command SET OFFSET nnnn. Lastly, a file lookup can be performed by giving a FILE command (with three default extensions).

17.1.3 Access Method

The program accesses the OS/8 device one OS/8 block (256 words) at a time. For every location specified, the real block and word are determined and compared with the current block in memory. If the desired block and current block are not the same, the <something-changed> flag is checked to see if anything has been changed in the current block. If nothing has been changed, the new block is read in.

If something has been changed, the current (modified) block is first written out and then the new block is read in. This action happens correctly even when the access mode is changed because it is done at the level of the OS/8 block number just before calling the current device handler. The status of the <something-changed> flag can be determined by simply SHOWing ABS, REL or ODT locations. If the flag is set, the word MOD will be output following location information.

The contents of the OS/8 device, therefore, do not change unless the block in which changes are made is written out either implicitly, as described above, or explicitly, using the WRITE command (discussed near the end of the section on word-type commands). The result is that typing CTRL/C before writing out the current block (assuming it has been modified) will return to the Monitor without modifying the contents of the device.

Note, also, that only one implicit write attempt is made by the program. Should an error occur when the write is attempted (for example, write-locked device), an explicit WRITE command must be given to actually write out the block.

If you change the words within some blocks accidentally, the <something-changed> flag can be reset by using the SET command to reset the device (described further along in this writeup) to the same device currently being used. This will reset the <something-changed> flag, the current block in memory, and the file start block and core-control-block/header-block (if they had been set by a FILE command).

The resetting of the current block in memory will cause the next access to the device to read in the block desired. The resetting of the file information will require a new file command to be given to set it back up. If you cannot remember the current setting of the device, use SHOW DEVICE first and then set it the same.

Files stored on an OS/8 mass-storage device generally fall into one of four categories. The program has four corresponding modes for accessing the device. The current mode of the program can be set by the SET command or by chaining (as described previously) and examined by the SHOW command (to be described later).

The four categories and their corresponding modes are:

1. General (binary, ASCII and data) files - NORMAL mode
2. Core image (save) files - SAVE mode
3. FORTRAN IV load modules - LOAD mode
4. System overlays - OFFSET mode

The actual operation of the program for each of these modes is as follows:

NORMAL — The high order 7 bits of the 15 bit address are added to the current block number to get the actual block number. The low 8 bits of the 15 bit address are used to specify the desired word within that block.

SAVE — The file to be examined must be set up by a FILE command. Block numbers are used to specify an overlay number (future MACREL/LINK support) and must be exactly zero (0) for files without overlays (generated by the monitor SAVE command). The core segment data (pages and fields) from the file's CCB (core-control-block) is used to determine where on the device the desired word is located. This is done by first determining the correct block from the file's CCB and then using the low 8 bits of the address to specify the desired word within that block. Specifying a nonexistent address or overlay for one of the single-character (ODT) commands will cause an error. Specifying a nonexistent address or overlay for any of the word-type commands will cause the program to ignore the address and access no data.

LOAD — The file to be examined must be set up by a FILE command. Block number specifications are actually taken as FORTRAN IV overlay specifications and must be contained within the file. You use the information from the OIT (overlay-information-table) in the header block of the file to determine where on the device the desired word is located. Nonexistent addresses are handled the same way as for SAVE mode.

NOTE

Because the block part of the location specification changes definition depending on the mode in use, it is recommended that the first operation following a switch to SAVE or LOAD mode explicitly specify a block part of 0. Otherwise a previously specified block part will be taken to mean a non-existent overlay number, causing an error.

OFFSET — The 12-bit OFFSET (set by the SET command and examined by the SHOW command) is subtracted from the low order 12 bits of the address and then the same arithmetic as with the NORMAL mode is used. This mode is used mostly with system overlays whose start block number and actual loading address is known. By setting the OFFSET to the loading address (which can only be a 12 bit number), the 12 bit actual addresses of the overlay can be used.

The SAVE and LOAD modes are mentioned together throughout this chapter as MAPPED modes because their method of address translation uses a descriptor block from the file of interest to control access to the file in a noncontiguous manner.

NOTE

For all access modes, the OS/8 block number for the block to be read is stored (for display) in the computer MQ register (if present). The value is stored before checking if the current block needs to be written. It is particularly useful for following the progress of the SCAN command.

17.1.4 Referencing Words on the Device

The words on the OS/8 device are referenced by their location (often abbreviated as <l>). This location consists of an optional block or overlay number (which must be followed by a "." if present), and an address or displacement. The block/overlay number is a 12-bit number which must be in the range 0 thru 7776 (octal), or 4094 (decimal). Block number 7777 (or 4095, decimal) does not exist under OS/8, and the program will ignore this number.

The overlay number is further limited to the number of overlays at a given address. Whenever the block/overlay part of the location is not used, the program will use the last specified value. The address/displacement is a 15 bit number (5 octal digits), but leading 0's need not be specified. Thus, the forms and their corresponding examples are as follows:

Form	Example
<block>.<displacement>	1201.37524
<overlay>.<address>	3.57633
<address>	15721
<displacement>	223

CAUTION

Neither this program nor the OS/8 device handlers generally include checking for legal block numbers. It is assumed that all accesses to the device will be done after checking with the directory for legal file start blocks and lengths, which is the normal mode of operation under OS/8. This can have very interesting results with this program; for example, the RK8/E handler, given a block number greater than 6257 (octal) on device RKA0, will continue on into device RKB0.

For the rest of this document, unless otherwise stated, block will mean block or overlay and address will mean <address> or <displacement>, depending on usage. Therefore the definition will be:

[block.]address=<location>=<l>

Since these location references are numeric input, all of the characteristics described next can also be used when specifying locations.

17.1.5 Numeric Item (or Numbers)

The program uses two switches, CTRL/D and CTRL/K, to allow the input of octal, decimal or mixed numeric input wherever numeric input is used. Each new command line always resets the input mode to octal. The character CTRL/D switches the input mode to decimal. The character CTRL/K switches the input mode back to octal. These two switches may be located anywhere in numeric input.

For example, when inputting a string of numbers, the input would be alternately decimal and octal if it were

^D100,^K100,^D200,^K200,^D300,^K300

Two other characters, the double quote (") and the apostrophe ('), may be used for numeric input. The double quote functions the same way in this program as it does in PAL8: the 8-bit ASCII value of the following character is used as a number. As with all character input, the special characters described earlier cannot be used. The apostrophe functions in the same way that the TEXT pseudo-op operates in PAL8: the following two characters are masked to 6 bits each and packed into a 12-bit word. Two characters must always follow the single quote. If you desire to pack one half of the word with a 6-bit 00, use the character "@". For example, a string equivalent to the file name PIP.SV would be represented by the following string:

'PI,'P@,0,'SV

Expressions may also be used for numeric input when enclosed in parentheses. Use parentheses for each expression, thereby making all the options of the EVAL command available for numeric input. For example, the contents of the switch register can be used for a number by the expression (S), or the current block number +5 could be used by the expression (B+5). See the discussion of the EVAL command for the other options available.

NOTE

Parentheses must surround the expression. Neither digits nor the switch characters may be outside of the parentheses or an error will result. This is required because many of the non-alphabetic characters have multiple meanings (commands or operators) so the use of parentheses eliminates ambiguity.

17.1.6 Errors and Error Messages

Whenever the program recognizes an error, it outputs an error message. The message tells both what went wrong and where in the command line the error occurred. Depending on the setting of the ERROR mode switch, either short or long messages are output:

```
?<ee>at<cc><error message>
```

or

```
?<ee>at<cc>
```

where <ee> is the error code, <cc> is the number of the column in the command line where the program stopped scanning, and <error message> is the message itself. There are currently 45 error conditions with corresponding codes and messages to assist you. The error codes and their messages can be printed out by the SHOW ERRORS command. The ERROR mode is set by the SET command.

The error messages are swapped with the USR but not in the normal manner, allowing write-locked startup with the loss of the message text (see Section 17.5 for more information).

17.2 SINGLE-CHARACTER (ODT-LIKE) COMMANDS

These commands allow you to modify and examine words on an OS/8 device in the same way that ODT allows you to modify and examine main memory.

In all of the following commands where the numeric item <n> is specified, the operation of closing the location is to place the value of <n> into the word, if open. If the current location is not open, or if <n> is not specified, no change takes place. Refer to Introduction to Programming and to Chapter 19, on ODT, for more information. Note that [<n>] (with the following commands) means that a numeric item may be supplied optionally.

<l>/	Open and output the contents of location <l> in the current OUTPUT mode.
/	Reopen the last location opened by one of these commands and output its contents in the current OUTPUT mode.
[<n>]#	Close the current location, reopen it and output its contents in BCD (3-digit binary-coded decimal).
[<n>]$ (dollar sign)	Close the current location, reopen it and output its contents in OS/8 ASCII.
[<n>]%	Close the current location, reopen it and output its contents in BYTE octal (8 bits with OS/8 packing).
[<n>]&	Close the current location, reopen it, and output its contents in XS240 format packed ASCII.
[<n>]:	Close the current lcoation, reopen it, and output its contents in SIGNED decimal.

[<n>]<	Close the current location, reopen it, and output its contents in OCTAL.
[<n>]=	Close the current location, reopen it, and output its contents in UNSIGNED decimal.
[<n>]>	Close the current location, reopen it, and output its contents in PDP (symbolic).
[<n>]?	Close the current location, reopen it, and output its contents in DIRECTORY format [negated DECIMAL, DATE (see "@" next) and packed (ASCII)].
[<n>]@	Close the current location, reopen it, and output its contents in DATE format: dd-mmm-yy 2 digits each for the day and year and 3 alphabetic characters for the month (except for illegal month numbers, which are output as a space and 2 decimal digits).
[<n>][	Close the current location, reopen it, and output its contents in ASCII.
[<n>]\	Close the current location, reopen it, and output its contents in FPP (symbolic).
[<n>]]	Close the current location, reopen it, and output its contents in packed ASCII.
[<n>]$ ("ALTMODE" or "ESCAPE" key)	Close the current location, reopen it, and type its contents as specified by the current FORMAT.
[<n>]<cr>	Close the current location.
[<n>];	Close the current location and open the next sequential location. Neither address nor contents are output, but one space is echoed.

NOTE

The ";" command advances through addresses without outputting their value in octal when some other format is more helpful. For example, when examining a directory, the file name and extension can be output using the "]" command (PACKED ASCII), the date can be output using the "@" command, and the file length can be output using the ":" command. All of this information can be made to appear on one line by using the ";" command. This does the incrementing between each of the output commands. The result would look similar to this:

```
2.5/2317]SO;]UR;]CE;]PA;@30-AUG-72;:-0071
```

For the following commands, the location of the newly opened word is output before the contents are output. This location is composed of the 12-bit block number (4 octal digits), a "." for a separator, and the 15 bit address (5 octal digits). This is immediately followed by a slash (/) to separate the contents from the address.

[<n>]<line feed>	Close the current location; open and output the contents of the next sequential location in the current OUTPUT mode.
[<n>]!	Close the current location; open and output the contents of the previous sequential location in the current OUTPUT mode.
[<n>]^(circumflex or up-arrow)	Close current location; open the location (that would have been referenced if the contents were a PDP-8 memory reference instruction), and output the contents of the new location in the current OUTPUT mode. This command works like the stand-alone version of ODT (not like the OS/8 version). Even if bit 3 of the word is set, this command will not do the equivalent of an indirect reference.
[<n>]_(backarrow or underline)	Close the current location, take its contents as an address, open that location, and print its contents in the current OUTPUT mode. This operates as an indirect address into the current field. The field currently being examined (the high octal digit of the 5-digit location) will not be changed by this operation.
<l>+	Open the location <l> locations forward from the current location, and output its contents in the current OUTPUT mode. 15-bit arithmetic is used and the block part is ignored, so this will operate across field boundaries, that is, within a 32K area.
<l>-	Open the location <l> locations backward from the current location and output its contents in the current OUTPUT mode. Same restrictions as with the '+' command.

The current OUTPUT mode has been mentioned several times above. The program will output the contents of a location either as a four-digit octal number or as a four-digit octal number with two spaces and the symbolic representation (PDP or FPP) of the word. See the SET and SHOW commands (Sections 17.3.2.4 and 17.3.2.5) and the following section.

17.2.1 Symbolic Output Formats

The symbolic typeout is in nearly the same format that input to an assembler would need to be to generate the contents of the current location. It is assumed that these contents are either a PDP-8 or an FPP-12/8A instruction, depending on the output selected. If the word to be output is not an instruction (as is the case for the second word of all 2-word instructions), the decoding will be meaningless.

For PDP-8 instructions, decoding into mnemonics is done for all memory reference instructions, for all legal operate instructions (including 8/E EAE instructions except for SWAB), for all 8/E processor, extended memory and memory parity IOT's, for teletype and high-speed paper-tape IOT's, for 8/E redundancy check option IOT's, for programmable real-time clock IOT's and for FPP IOT's.

There are currently a total of 96 IOT's, and the program has space for an additional 32 IOT codes and their mnemonics. These can be patched directly into the program using itself. The first word of each four-word entry is the IOT code (for example, 6221 for CDF 20), followed by 3 words containing up to 6 packed ASCII characters padded with trailing 0's.

No attempt is made to decode any micro-coded IOT's. Either an exact match for the current contents will be found in the table or the program will output:

```
IOT nnnn
```

where nnnn is the octal typeout of the low 9 bits of the code.

The next free location in the table (in field 1) is pointed to by the contents of location 10000. The table is terminated by the first 0 for an IOT code, so additions must be contiguous and added directly at the current end of the table.

For FPP instructions, the full FPP-8/A instruction set is decoded except for IMUL, which is actually an integer mode LEA. For the data manipulation instructions, the op-code mnemonic is followed by a "#" for the long-indexed format, by a "%" for the indirect-indexed format, and by a space for the base addressing format.

For the indirect-indexed and base addressing formats, the operand address is output as:

```
B+nnn
```

where nnn is the 3-digit octal value of the displacement (3 or 7 bits) multiplied by 3. These formats are those used by the RALF assembler. This is also true for LEA instructions (that is, LEAI is decoded as LEA%).

Both jump and load-truth instruction decoding is done as a single mnemonic whose last two characters indicate the specified condition. All instructions that use two words are decoded with an asterisk in the location in the normal assembler format where the value of the second word would go.

Index register number and "+" for auto-increment (if used) are also shown in the assembler format. Any combinations that are not in the FPP-8/A instruction definitions are output as unused.

NOTE

For both of these output formats, the use of the mapped access modes (and the OFFSET mode for PDP decoding) allow the use of the actual addresses when decoding the instruction.

17.3 WORD-TYPE COMMANDS

These commands are grouped by function, as follows:

Group 1:

DUMP	type/list out the contents of one or more blocks.
LIST	type/list out the contents of one or more locations.
MODIFY	modify one or more locations.

Group 2:

WORD	word search
STRING	string search
SMASK	set up string search mask

Group 3:

SET	set up program switches and variables
SHOW	show settings of program switches and variables
FILE	look up file(s) on device
WRITE	write out current buffer
SCAN	scan for bad blocks
REWIND	move device to block 1 and reset directory segment

Group 4:

OPEN	open an output file on a file-structured device
CLOSE	close the open output file

Group 5:

IF	cause command skipping based on expression value
END	resume command execution after unsatisfied IF
COMMENT	pass user commentary to output device
EXIT	exit to OS/8 (same as CTRL/C)

Group 6:

EVAL	evaluate a signed, double-precision expression.

Command words may always be abbreviated to their first two characters, as with the Monitor and BUILD, and some of the commands and their options may also be abbreviated to only one letter. When this is the case, the command forms given will include the one-letter form. The option forms will give the one-letter form directly under the full word form.

NOTE

In many cases, two or more words start with the same letter. In these cases, only one of these words may be abbreviated to one letter.

The descriptions for each command include each of the possible forms of the command; an example of that form follows it on the same line.

17.3.1 Output Formats

The FORMAT option is used to SET up the output format for the "$" (ALTMODE or ESCAPE) command, described earlier, and the default format for the DUMP, LIST and MODIFY commands, described below. The syntax of this command is shown with the other SET commands, but is described here to make the descriptions of the following three commands more understandable. The format may be one of the following:

ASCII A	output each word as a single ASCII character.
PACKED P	Output each word as two 6-bit trimmed and packed ASCII characters. This is the format of PAL8 TEXT strings.
OS	Output each word as 1 or 2 OS/8 packed ASCII characters. The even address words output 1 character and the odd address words output 2 characters.
XS240	Output each word as two 6-bit packed ASCII characters by adding a space (240 octal) to the contents of each 6-bit byte. This is the format of PAL12 SIXBIT strings.
BYTE	Output each word as 1 or 2 OS/8 packed bytes of 8 bits each as 3-digit octal numbers. The even address words output 1 number and the odd address words output 2 numbers.
UNSIGNED U	Output each word as an unsigned decimal number.
SIGNED S	Output each word as a signed decimal number.
OCTAL O	Output each word as a 4-digit octal number.
BCD B	Output each word as 3 BCD digits. The digits 0 through 9 are followed by ":" (10), ";" (11), "<" (12), "=" (13), ">" (14), and "?" (15).
PDP FPP	Output each word as an octal number, followed by 2 spaces and its mnemonic representation, assuming it to be a PDP-8 or an FPP-8A instruction. See the symbolic output description.
DIRECTORY	Output each word in octal, decimal (signed), date (see "@" command) and packed ASCII formats.

The FORMAT is initialized to packed ASCII.

The output from the DUMP and LIST commands for each of these formats is set up as follows:

1. At the beginning of each line the current location is output in location format with a 4 digit block number and a 5 digit address, both in octal, as

   <block>.<address>:

   For example, 1271.17205: - location 17205(8) relative to block 1271(8).

2. The maximum number of words per line is set up as follows:

   a. The four character formats output 16 words per line with no extra characters.
   b. The five numeric formats output 8 words per line with 2 spaces between each number.
   c. The symbolic and directory formats output 1 word per line.

For LIST with A or B, the first line may be shorter than succeeding lines to force the second and following address outputs to be even multiples of 10 (octal).

17.3.1.1 **DUMP** - The DUMP command outputs one or more whole 256-word device blocks in the default or an optionally supplied format. This command has the following forms:

```
DUMP [<format>] <block string>

DUMP <block string>                 DU 100,200-213,250
D <block string>                    D (B)-(B+10),(S)
DUMP <format><block string>         DU PA 212
D <format><block string>            D OS 514
```

where the optional <format> is one of those given for the FORMAT option above, and the <block string> is one or more numeric items separated by commas and dashes. The dash is used when it is desired to dump a group of blocks, and is used as

```
<start block>-<end block>
```

the comma separates single blocks or groups of blocks if there is more than one per line.

NOTE

In a mapped mode (SAVE or LOAD), the DUMP command cannot dump any block except that block containing location 0. To eliminate the confusion that this would produce, the command will simply output an error message reminding you that the proper command to use in a mapped mode is the LIST command.

The output from the DUMP command is sent to the DDEV (dump device), which can be either the console terminal, the line printer, or a file. See the SET command for setting the dump device and output mode.

17.3.1.2 **LIST** - The LIST command outputs the contents of one or more words on the device in the default or in an optionally supplied format. This command has the following forms:

```
LIST [<format>] <location string>

LIST <location string>              LI 123.200-517,200.0
L <location string>                 L 312.10-17,100-117,176
LIST <format><location string>      LI UN 200-227
L <format><location string>         L SI 200-277
```

where the optional <format> is one of those given for the FORMAT option above, and the <location string> is one or more locations, separated by commas. When it is desired to list a group of words, the dash is used to separate the start and end addresses as

[<block>.]<start address>[-<end address>]

If the block part is not specified, the last block number specified to the program will be used. If an end address is specified, the start address is assumed to be in the same field as the end address (that is, the highest octal digit of the 5-digit address), so a maximum of 4096 words can be specified by each group.

As with the DUMP command, the output from the LIST command is sent to the DDEV. For more information see the last paragraph of the DUMP command, the SET command, and Section 17.5.

17.3.1.3 **MODIFY** - The MODIFY command allows a string of locations on the device to be easily changed. Specify the format of the input, letting the program do the work of storing the data properly. This command has the following forms:

MODIFY [<format>] <location string>

MODIFY <location string>	MO 200.0-17,35-43
M <location string>	M 32745-32777
MODIFY <format><location string>	MO PA 12342-12360
M <format><location string>	M AS 367.7261-7275

where the <location string> has exactly the same format as for the LIST command (the <format> options are shown below). If the <format> is not specified (as with the first form), the program will pick the format that corresponds to the current setting of the FORMAT option. The formats are shown below.

MODIFY format	FORMAT setting and MODIFY action.
ASCII A	ASCII - one character of input is stored in each word to be modified.
PACKED P	PACKED - two characters of input are packed as trimmed 6-bit characters, padded with trailing 00's. Control characters (those with codes less than 240 octal) are packed as a 6-bit 77 (flag) and the low-order 6-bits of the character. Note that this means that "@" is packed as a terminator (00) and that "?" is not unique.
OS	OS - three characters of input are packed into two words to be modified. In this format, the start address must be even and the end address must be odd.
XS240	XS240 - a space (240 octal) is subtracted from each character and then it is packed as 6-bit bytes. Control characters are handled as with PACKED format.

NUMERIC N	SIGNED & UNSIGNED decimal, BCD, OCTAL, BYTE, PDP, FPP and DIRECTORY formats - the input is a string of numeric items which are stored one per 12-bit word. See the section on numeric items. Note that bcd, byte, directory and symbolic are not included, that decimal or octal input are determined by the CTRL/D and CTRL/K switches and that signed numbers must be input enclosed in parentheses, for example, 17, (-10), ^D200, (-^K312), 40, (-^D35*129).

For each location or group of locations specified by the <location string>, the program will prompt for the input by printing the start location in the same format as described under the output format options above.

CAUTION

> The program always modifies exactly the number of words specified by each item in the <location string>. If you input extra characters for the character formats or extra numeric items for the numeric format, they will be ignored. If you do not input enough characters or items, the rest of the words to be modified will be set to the FILLER value (see the SET command). The program will not output any message if either of these things takes place. This does, however, make it possible to fill from 1 to 16 blocks on a device with zero or some other value by specifying all the words to be filled in NUMERIC format and then responding to the prompt with a single F (the value of the FILLER) and RETURN.

Input to the program is always terminated by a carriage-return. It is not possible to insert a carriage-return into a word using this command. All of the editing keys are available for use during input, so the CTRL/C, CTRL/Q, CTRL/S, CTRL/R, CTRL/P, CTRL/U and RUBOUT characters cannot be entered using this command. For all of the character input formats, spaces (excluding leading spaces, which are ignored) and tabs in the input string are packed as they are seen. For numeric input, spaces are ignored and the numeric items must be separated by commas.

You can always abort the command by CTRL/P if you change your mind before you press the RETURN key.

17.3.2 Search Limits

The program has two search commands: the WORD search and the STRING search. Both search from a lower limit to an upper limit. The limits are either the LOWER and UPPER limits set by the SET command (the default) or the limits set up by the FROM <1> and/or TO <1> clauses that can optionally follow the command word. FROM <1> overrides the lower limit, and TO <1> overrides the upper limit. Leaving out the

block parts of either of the two temporary limits will cause the program to use the block part of the corresponding default limit set by the SET command. In a mapped (SAVE or LOAD) access mode, searching through non-existent locations or overlays will never produce a match. Whenever a match is found, the program outputs the location where the match occurred, followed by the word or string that matched.

NOTE

> You cannot search through more than one overlay per search command. To do so would require different and separate handling of the block and address parts of the limits when in the mapped modes, including the resetting of the address part. The result is that, in the mapped modes, the block parts are used to set the overlay to be searched (lower limit only), and only the address parts are used in the determination of the number of words to be searched.

17.3.2.1 **WORD (Search)** - The WORD search command searches for a word or words which, masked by the MASK (which is set by the SET command), will match the search word (also masked). This command and its five options follow:

```
WORD [UNEQ] [ABS] [MEM] [FROM <l>] [TO <l>] <n>

WORD <n>                    WO 217
W <n>                       W (S)
WORD UNEQUAL <n>            W UN 0
WO U <n>                    WO U (C&377)
WORD ABSOLUTE <n>           WO AB 7402
W A <n>                     W A 7000
WORD MEMREF <n>             WOR MEM 41
WO M <n>                    WO M 40
WORD FROM <l><n>            WO FR 213.0 2317
W F <l><n>                  W F 1.35 (S)
WORD TO <l><n>              W TO 213.345 1111
W T <l><n>                  WORD T 6257.377 7777
```

...and any combination and order of the above options.

In this command and its options, <n> is the bit pattern being searched for, UNEQUAL means that all words which are not equal to <n> under the mask do match, and the temporary limits clause is as described above. ABSOLUTE means that the location where the match occurred is to be output as an absolute block number and displacement rather than as a relative location. MEMREF means that only words whose high-order octal digit is 0 thru 5 (that is, the PDP-8 memory reference op-codes) are allowed to match, independent of the setting of the MASK.

When you want to search for those words that reference a specific location, set the MASK to 377 (octal) and then use the MEMREF option. This will exclude all Operate (op-code 7) and IOT (op-code 6) instructions from the output. This will make it easier to find the desired information (for example, you will not output the location of every CIA, 7041 octal, when you are looking for references to location 41 octal).

NOTE

UNEQUAL has a higher priority than MEMREF, so first each word is tested under the mask for equal/UNEQUAL and if the specified condition is true, then the word is tested for the MEMREF condition.

17.3.2.2 **STRING (Search)** - The STRING search command searches for a string of numbers (bit patterns) under an optional string mask. This command has four options and has the forms:

```
STRING [MASKED] [ABS] [FROM<1>] [TO<1>] <numeric string>
```

```
STRING <numeric string>              ST 4557,0,0
STRING MASKED <numeric string>       ST MA 4577,0,1203
ST M <numeric string>                ST M 5566,0
STRING ABSOLUTE <numeric string>     ST AB 'PI,'P@
ST A <numeric string>                ST A "A, "B
STRING FROM <1><numeric string>      STR FR 100 1,4000,2
STR F <1><numeric string>            ST F 123.4567 (S),(-S)
STRING TO <1><numeric string>        STR T 7577 'ER, 'RO, 'R@
ST F <1> T <1><numeric string>       ST F 1.0 T 7.0 'FO, 'TP
```

...and any combination and order of the above options.

In this command and its options, the numeric string is simply a string of numeric items separated by commas. MASKED specifies that the search is to be done under the string mask. ABSOLUTE is as for the WORD search, and the temporary limits clause is as described above.

When the MASKED option is used, each item of the numeric string is masked by a separate mask word from the string mask. If the string mask is shorter than the search string, it is used in a circular fashion (the first word follows the last) as many times necessary to mask all of the items of the search string. If the string mask is longer than the search string, the extra words are not used. This feature allows for very complex searches to be done.

For example, you want to find all calls to a certain subroutine in a file and also see their arguments. This could be done as follows:

```
FILE FUTIL                      -look up file to be searched
FUTIL.SV 6070-6120 ^P           -you stop typeout
SE MODE SAVE                    -set access mode to mapped
SMASK (-1),0,0                  -set mask for 2 arguments per call
ST M 4547,0,0                   -search for 4547 and 2 dummies
```

The output will give the address of the subroutine call (which requires an exact match due to the mask of 7777) and the contents of the two following words (which can be anything, since they are masked by 0).

Using the mask specified above, a search could be made for an exact match, 2 "don't care words" and another exact match by simply specifying a search string with 4 arguments. The first item of the string mask will be used to mask both the first and the last items of the search string.

This command can be particularly useful when trying to find certain kinds of references in programs for which no CREF listing (or perhaps no listing at all) is available.

17.3.2.3 **SMASK** - The SMASK command sets up the string mask. It has the following form:

SMASK <numeric string>	SM (-1),0,0,7000,0

where the numeric string is the same as for the STRING search command above. The current contents of the string mask may be examined by the SHOW command.

17.3.2.4 **SET** - The SET command sets up various switches and variables within the program. It has many options, each the name of the switch or variable, and is always followed by a word or number describing how it is set. All items are separated by spaces. The command has the following two forms:

SET <option(s)>	SE OU PDP ERR LONG MODE SAV
S <option(s)>	S LO 100.0 UP 123.377 IDEV LPT

where the options are as follows:

OUTPUT OCTAL OUTPUT O O PDP O P OUT FPP O F	Set the output mode for the single-character commands. Initialized to OCTAL.
ERROR SHORT E S E LONG ERROR L	Set the mode for error message output. The SHOW ERRORS command will list all error messages. Initialized to LONG. Also set to SHORT by write-locking system device.
FORMAT <format>	Set output format for LIST, DUMP, etc. The formats have been described previously. Initialized to PACKED ASCII.
OFFSET <1>	Set the offset to the low 12 bits of <1>. Initialized to 0.
FILLER <n>	Set the filler to the low 12 bits of <n>. Initialized to 0.
LOWER <1>	Set the lower search limit. Initialized to 0.200.
UPPER <1>	Set the upper search limit. Initialized to 0.17577.
DEVICE <device name[:]>	Set up the OS/8 device for access. The handler is fetched at this time. Initialized to SYS (device 01). ":" In <device name[:]> is optional. <device name> is an assigned or permanent OS/8 mass storage device name.
DDEV <device name[:]>	Set up the dump device. Initialized to SYS. See also DMODE below and OPEN and CLOSE' commands.

MODE NORMAL MODE N MODE SAVE MODE S MO LOAD MO L MO OFFSET MO O	Set up the device access mode. These have been described previously. Initialized to NORMAL.
DMODE NONE DMODE PART DMODE ALL	Set up the dump output mode. Initialized to NONE, which sends all output to console only. PART sends DUMP, LIST and SHOW ERRORS output to the DDEV (perhaps to a file). ALL sends all output to both the console device and to the DDEV. (See section on file output.)
MASK <n> M <n>	Set the WORD search mask to the low 12 bits of <n>. Initialized to 7777.
TEMP <n>	Set the TEMP storage to the 24-bit value of <n>. Value is returned by subsequent use of the T in expressions.

As many options as desired may be specified on one command line, separated by spaces. In the event of an error, none of the options past the point where the error occurred will have been set. If you have any question, use the SHOW command.

17.3.2.5 **SHOW** - The SHOW command lists the current setting of any of the program switches and variables set by the SET command and other information. The program outputs either words or numbers to best describe the current settings. As with the SET command, as many of the options for this command as desired may be specified on single command line, separated by spaces. This command has the form:

SHOW <option(s)> SH BL CCB LOW UP ODT REL ABS

where the options are as follows:

BLOCK B	Output in octal the start block number of the last file specified by the last FILE command.
CCB C	Output the core control block of the last file specified by the FILE command. If the file is not a SAVE file, an error will occur. The start address of the file is output as a 5-digit octal number, the job status word (JSW) is output in octal, and the core segments are output as 5-digit octal addresses.
HEADER H	Output the header block information for the last file specified by the last FILE command. If the file is not a LOAD file, an error will occur. The start address is output as a 5-digit octal number, followed by the next free address as a 5-digit octal number, the loader version number in octal and a message if Extended Precision is required. Then, for each level, a line is output with the number of overlays, the 5-digit start address, the relative start block and the length of the overlays (in blocks) for this level.

ABSOLUTE A	Output the absolute location of the last word accessed on the device in <location> format (a 4 digit octal block number, a "." and a 5-digit octal address) and the word MOD if the current block has been changed (the <something-changed> flag is set).
RELATIVE R	Output the relative location (what you specified) of the last word accessed on the device in <l> format and the word MOD if the current block has been changed.
ODT	Output the relative location of the last word accessed by one of the special-character commands in <l> format and the word MOD if the current block has been changed..
LOWER	Output the search lower limit in <l> format.
UPPER	Output the search upper limit in <l> format.
FILLER	Output the value of the filler in octal.
MASK M	Output the WORD search mask in octal.
SMASK	Output the current contents of the STRING search mask as a string of octal numbers.
OFFSET	Output the value of the offset in octal.
MODE	Output the name of the current setting of the device access mode switch (NORMAL, SAVE, LOAD or OFFSET).
DEVICE	Output the OS/8 deivce name and number.
DDEV	Output the name of the dump device.
OUTPUT O	Output the name of the current single-character (ODT) command OUTPUT mode (OCTAL, PDP or FPP).
FORMAT F	Output the name of the current output format.
VERSION	Output the current version number of FUTIL.
ERRORS E	Output a complete list of all error codes and their corresponding messages. Note: this list is output to the DDEV (dump device) so that it can be output using the LPT handler for your system. Note that Version number is also output with errors.

17.3.2.6 **FILE** - The FILE command locates files on the OS/8 device and sets up the start block of a file for the mapped access modes, SHOW CCB, etc. This command has the forms:

```
FILE <file name string>      | FI FUTIL PIP.SV
F <file name string>         | F MICRO.LD
```

where the <file name string> is a string of one or more OS/8 file names, separated by spaces. Any other characters except "." will be

taken as part of the file names. The program assumes extensions of .SV, .LD and null (in this order) when looking up the file. This can lead to a substantial amount of time when a large directory is searched three times for a file that does not exist. Specifying an extension will cause only one lookup attempt to be made. A null extension, if desired, may be specified by making the "." the last character of the file name. The program does one or more separate lookups for each file name specified and outputs either

```
<file name> ssss-eeee oooo (dddd) b.lll dd-mmm-yr
```

or

```
<file name> ssss-eeee oooo (dddd) b.lll
```

or

```
<file name> LOOKUP FAILED
```

where "ssss" is the start block of the file in octal, "eeee" is the last block of the file in octal, "oooo" is the length of the file in octal, "dddd" is the length of the file in decimal, "b.lll" is the block (segment) and location within that block of the first word of the file entry (the first two characters of the name) in the directory, and dd-mmm-yy is the file date. If the directory does not contain the extra word required for the date or the date word of the file is 0, the second form with no date will be output rather than the first form. The LOOKUP FAILED message means either that the file name was not found on the device or that the device is a write-only device.

The actual lookup operation is performed by the OS/8 USR, which is swapped as needed (see section on program execution). Since the USR keeps track of the current device once the first FILE command is given, it will have the wrong directory in memory if the medium (tape or disk) is changed on the physical device. This can be solved one of three ways:

1. Use the REWIND command to rewind the device being removed and clear the directory segment from the USR.

2. Do a SHOW ERRORS and abort the output when the message output begins. This will have swapped out the USR. If messages are not available, use 1 or 3.

3. Use EXIT or CTRL/C to return to OS/8 and then directly restart FUTIL with the OS/8 START command. This will have swapped out both error messages and USR from memory.

Any of these methods should be followed by a SET command to reset the device and the rest of the I/O parameters desired.

The last file name specified that did not have a LOOKUP FAIL will be the file used in the mapped access modes, SHOW CCB, etc. The program is initialized with no known file, so attempting to access any location in a mapped access mode or attempting to SHOW CCB or SHOW HEADER without giving a valid FILE command will cause an error.

17.3.2.7 **WRITE** - The WRITE command forces the program to write out the block currently in memory. It has the form:

WRITE [<block>]

where the optional <block> overrides the default number of the block that was read to specify where the current block is to be written. This dangerous operation does allow a limited amount of copying in a special situation, e.g., allowing a directory to be backed up by moving a copy to the end of the device (see the examples section) or copying a single block from one device to another by changing the DEVICE and then doing a WRITE (with or without an argument). Again, as stated in the section on accessing the device, caution must be used because attempting to write beyond the end of a device may not be checked by the handler.

17.3.2.8 **SCAN** - The SCAN command does a rapid scan for read errors on the current device. It has the form:

SCAN <block string> SC 0-6257

where the block string is of the same form as for the DUMP command. Each block is simply read. If an error occurs, it is reported as:

oooo BAD BLOCK

where "oooo" is the block number in octal, and the scan continues. This is the only FUTIL command that will continue on a read error. If the current block has been changed, and if any other blocks are included in the scan, an implicit write will be attempted by FUTIL. An error on this implicit write will be reported and then the command will be aborted. This is the only time that this command will attempt a write. The command can then be repeated if desired and it will execute (only one implicit write attempt is ever made by FUTIL).

NOTE

The OS/8 actual block number for the block to be read is stored for display in the computer MQ register, if present. It is particularly useful for following the progress of this command. The value is stored before checking if the current block needs to be written.

17.3.2.9 **REWIND** - The REWIND command is used to move a tape back to block 1 and to reset the USR directory segment. It has the form:

REWIND

and must be terminated by the RETURN key. It causes a read of block 1 of the device and resets the directory segment in the USR (if in memory). Any subsequent FILE command will cause the directory to be read.

17.3.3 File Output

Output to file-structured or non-file-structured dump devices is provided through two commands, OPEN and CLOSE, and two SET options, DDEV and DMODE. They can be used to simply make fast hard copy output from the DUMP, LIST and SHOW ERRORS commands, to provide a hard copy log of all operations carried out with a video terminal, to provide an ASCII file output of some data for later processing by another program, etc.

Output to file-structured and to non-file-structured devices (serial devices) is handled in two separate ways. Output to the file-structured device is done by first setting the DDEV and DMODE and then OPENing an output file. No output to the device will be done until the file is open (to protect your directories), and then output will be done one block at a time. When output to the file is complete, CLOSE your file to make it a permanent file (properly terminated with a CTRL/Z and padded with nulls).

Output to a non-file-structured device is done by simply setting the DDEV and DMODE. Output to the device will be done one line at a time, as soon as specified by the DMODE, and neither the OPEN nor the CLOSE commands are needed. The output is done by padding the buffer with nulls after each line is ready and then calling the output device handler, so the handler used should ignore nulls (which leaves out the PTR: handler, for example).

17.3.3.1 **OPEN** - The OPEN command opens an output file on file structured devices for partial or total output from the program. It has the form:

OPEN <file name>

where the file name should be a standard OS/8 file name. The extension defaults to .DU (for dump) if none is supplied.

WARNING

> FUTIL gives significance only to the characters space, carriage-return and "." when scanning file names. It is your responsibility not to include characters that are not legal to other OS/8 programs or the files will be able to be accessed only through FUTIL or the CCL command decoder.

This command must be given after the dump device is SET by the DDEV option. The output specified by the DMODE will then be sent to this file, one block at a time (packed only 8 bits per word), until either the DMODE is changed or the file is closed.

Files can be opened at will without closing any previous file. This gives the user additional flexibility, but at the expense of possibly losing an output file if it is not closed.

Should an error occur on the output device while doing output, the file is simply thrown away (it cannot be closed).

17.3.3.2 **CLOSE** - The CLOSE command closes an output file previously opened. It has the form:

```
CLOSE
```

and must be on a line by itself. If given with no file open, it is simply ignored.

17.3.4 Batch Operation

Operation of FUTIL under BATCH allows repeated operations to be done without re-entry. All of the operations provided under interactive operation are provided except that the RUBOUT character is simply ignored, input is taken directly from the BATCH stream and console output goes to the log output device.

Four commands have been added specifically to support use of FUTIL under BATCH: IF, END, COMMENT and EXIT. These commands are also available for interactive use, but are not as important in that mode.

17.3.4.1 **IF** - The IF command was implemented specifically to allow FUTIL, when operating under BATCH, to be sure that the correct operations are proceeding before modifying something incorrectly. It has the form:

```
IF<expression>                    IF C-3575
```

where <expression> is a general expression of the same form as used by the EVAL command. If the expression evaluates to exactly zero (as a 24-bit integer), command execution will continue as though the command had not been seen. If the result is not exactly zero, command skipping will begin and will continue until a line containing the single word END is found. Command execution will then resume.

This command was set up to test only for zero under the assumption that a test is to be made for some exact quantity. However, the capabilities of the expression evaluator can be used to generate sufficiently complex expressions for other tests. For example:

```
IF 40000000&(......)      will test for positive
IF -(40000000&(...))-1    will test for negative
IF 10000&(-(77770000!(...))) will test for 12-bit non-zero
```

17.3.4.2 **END** - The END command re-enables command execution following an unsatisfied IF command. It has the form:

```
END
```

and must be on a single line by itself. When encountered during command execution, it is ignored. The IF/END commands cannot be nested because the first END found will re-enable command execution for any number of previous IF commands. For example:

```
IF...
IF...
IF...
END                       will terminate all three.
```

17.3.4.3 **COMMENT** - The COMMENT command allows optional comments in command input which will simply be ignored during execution. It has the forms:

```
COMMENT  [<comment>]          COMMENT THIS IS ONE
C        [<comment>]          C
```

where [<comment>] is an optional comment. Note that blank lines may also be used for formatting of the output log but that they will also close any open location.

17.3.4.4 **EXIT** - The EXIT command provides a method of return to OS/8 other than CTRL/C. It has the form:

```
EXIT
```

and the rest of the line is ignored. Exit does not write out the last block modified. Use WRITE to make changes permanent.

17.3.4.5 **EVAL** - The EVAL command evaluates a parenthesized expression of signed double-precision integers. It has the forms:

```
EVAL <expression>             EV S*^D4096+D
E <expression>                E B*400+L
```

where the <expression> follows the normal rules for arithmethic expressions. Legal operators, in their order of precedence are:

```
(    evaluate inner expression
/    signed division
*    signed multiplication
-    subtraction
+    addition
&    logical product ("and")
!    logical sum ("or")
)    expression end
```

Besides 24-bit numeric input (which can be octal, decimal or mixed octal and decimal) under the control of the CTRL/D and CTRL/K switches and ASCII and packed ASCII using " and ', the following variables may be used:

```
C    current contents (of location L).
L    current location (15 bit, same value as is output by the
     SHOW RELATIVE command).
B    current block number (as for L).

F    contents of FILLER (12 bits).
T    contents of TEMP (24 bits).

S    contents of the console switch register.

R    the remainder of the last division or the high product of
     the last multiplication.  (24 bits, the sign may not be
     correct.)

D    contents of OS/8 Monitor date word.
```

Overflow on addition, subtraction and multiplication are ignored, but trying to divide by 0 will cause an error.

If no errors occur, the program evaluates the expression and types out the results in the form:

```
=oooooooo (sddddddd)
```

where "oooooooo" is the double precision result in octal and "sddddddd" is the signed double precision result in decimal (the sign is either a dash or a space).

17.4 EXAMPLES

These examples help provide an overview of the use of the program. The first two examples are discussed in detail to illustrate the mechanics of the operations, while the following examples are intended primarily to show what can be done with the program. Should questions arise on the mechanics, review the first two examples and the discussions of the commands in question.

Example 1:

Assume that you would like to know what CCL remembers of your last .UA command. What it remembers is stored on block 65 (octal) of the system device. As described in the source of CCL, each unit of what it remembers is allocated 40 (octal), or 32 (decimal) words in this block. The first four of these words contain binary information, and the last 34 words contain the last input command, stored as packed ASCII characters. The lines contain the inputs for the commands as follows: TECO and MAKE (line 0), EDIT and CREATE (line 1), COMPILE and EXECUTE and PAL (line 2), UA (line 3), UB (line 4), and UC (line 5). Thus, the saved .UA command can be listed by outputting the contents of the 4th through 37th words of area 3 in block 65 as packed ASCII characters as follows:

```
.R FUTIL                          -call FUTIL from OS/8

EVA 3*40+4                        -calculate start displacement
=00000144 (0000100)               -of the 3rd line (=144[8])
```

Now list the words of this line with the LIST command, specifying the output format to be PACKED ASCII characters and the words to list to be block 65 locations 144 (from above) through 144+33 (the expression for the location of the last word of this line). FUTIL responds with the start location and a line of characters, and the next location with a multiple of 10[8] as an address and a line of characters.

```
LIST PACKED 65.144-(144+33)       -list the words wanted

0065.00144: DIR R:FUT???.*/E/R=3
0065.00160:                         -that's it!
```

NOTE

For the examples above and below, the symbol <cr> is used to show that you need to terminate your command lines with a carriage return. All other lines above are output by the program.

Example 2:

Now assume that you would like to make the simple patch for OS/8 FORTRAN IV users with an FPP-8/A to use the lockout feature of the FPP-8/A (from the August 1976 DIGITAL Software News). This requires changing the contents of location 15776 of FRTS (the Fortran Run Time System) from 400 to 410 (which adds the lockout bit). You also want to update the date word of the directory entry for FRTS (the 4th word beyond the start of the entry) to show that the file has been updated. This is done as follows:

```
.R FUTIL                                -call it

SET MODE SAVE                           -set FUTIL to a mapped mode
FILE FRTS                               -look up the file to map
FRTS.SV 0671-0722 0032(0026) 1.327 31-DEC-75
                                        -1.327 is start of entry!
```

Now use ODT command / to open and change one word.

```
15776/0400 410                          -add LOCKOUT bit

SET MODE NORMAL                         -switch to unmapped
```

Now use ODT command / with an expression to open the date word, command @ to output it in date format and then put today's date (as an octal value) in its place.

```
1.(327+4)/6375
@31-DEC-75(D)                           -change file date to today's date

WRITE                                   -send out this change
```

NOTE

First the file FRTS.SV is changed, and then the OS/8 directory is updated to the current date. Changing the address desired from FRTS to the directory automatically writes out the modified block of FRTS before reading in the directory segment that contains the file name. However, the changed directory segment must be written out explicitly because there are no other blocks to examine for this example.

Example 3:

While doing a /S transfer with PIP, PIP gives a read error in your file SOURCE.PA. Attempting to read it with EDIT causes EDIT to type ?0^C and return to the Monitor. Find out what is wrong as follows:

```
.R FUTIL

FISOURCE.PA                             -look up the file
SOURCE.PA 0243-0351 0107 (0071) 2.005 30-AUG-74

SE MASK 0 LO 243.0 UP 351.377           -set up mask & limits

W UNE 0                                 -search the file
```

```
?ee AT 08 FATAL READ ERROR      -here is the problem
[Note:  "ee" may change with version, so is left out.]
SH ABS                          -find out where it is
ABS.LOC=0271.00000

WR                              -attempt to clear error

DU OS (B+L/400)                 -it worked, now dump it

0271.00000:....^P               -change your mind

W UN FR 272.0 0                 -check the rest of the file

^C                              -ok, now go fix the source
```

This sequence can also be carried out using the SCAN command as follows:

```
.R FUTIL
F1 SOURCE.PA                  - use CCL to call & lookup

SOURCE.PA 0243-0351 0107 (0071) 2.005 30-AUG-74
SCAN 243-351                  - scan the area

0271 BAD BLOCK                - here is the problem!

271.0/ ?ee AT 07 FATAL READ ERROR - get block with trouble

WR                            - attempt to clear error

DU OS (B+L/400)               - it worked, now dump it

0271.00000:....^P             - change your mind

^C                            - ok, now go fix the source
```

If the error had been of some type other than a clearable error, the WR command might also have failed.

Example 4:

After using BUILD to change your system, find out the device number for DTA1:

```
.R FUTIL

SE DEV DTA1                   - fetch the device handler
SHOW DEV
DEVICE = DTA1 (06)            - number is decimal
```

Example 5:

By accident you zero a DECtape directory which contains the only copy of a file you need. You have the PIP /E listing of the directory but only want to re-build it enough to get the wanted file. The name of the file is LOST.FI:

```
.R FUTIL

SE DEV DTA1                   - it was here
EV ^D5+14+11+10+16+13+8+5     - lengths of all preceding
= 00000122 (0000082)          - files
EV ^D730- ^K61- ^D82-25       - rest of DECtape room
= 00001076 (0000574)
```

```
1.0/ 7777 (-3)                  - now 3 files
4/ 7777                         - 1 extra word per entry
0001.00005\ 0000 'DU            - set up a "DUMMY" file
0001.00006\ 7556 'MM            - over the old <EMPTY>
0001.00007\ 1752 'Y@
0001.00010\ 3451 0              - a null extension
0001.00011\ 6234 (D)            - put in today's date
0001.00012\ 4235 (-^D82)        - length
0001.00013\ 5761 'LO            - the desired file
0001.00014\ 3341 'ST
0001.0015\ 2371 0
0001.00016\ 1107 'FI            - the extension
0001.00017\ 1366 (D)
0001.00020\ 3015 (-^D25)        - its length
0001.00021\ 3415 0              - an <EMPTY> to end it
0001.00022\ 2713 (^D574)        - the rest of the tape

WRITE                           - now write it out
^C                              - & exit to use it
```

The LINE-FEED key was used to advance through the words.

The above example is exactly the same as hand calculating the required length of the DUMMY file and then doing the following sequence using PIP:

```
.R PIP
*DTA1:DUMMY</I=122              - enter the DUMMY file
*DTA1:LOST.FI</I=31             - enter the LOST.FI
*^C
```

Note that the lengths of the files are specified for PIP in octal.

Example 6:

Search for the end of each page of text in the file WRITE.UP. Since the file is an OS/8 ASCII file, which has two characters packed in the low 8 bits of two words and a third character packed in the high 4 bits of both of the two words, the form-feed character (^L) may be packed as the third character in some cases. So it is necessary to search both through the low 8 bits of each word and through the high 4 bits of each pair of words. Do it as follows:

```
.R FUTIL

FI WRITE.UP
WRITE.UP 0301-0437 S^P          - typeout stopped
SE MA 377
SE'LO 301.0 UP 437.377          - char mask & limits set

W A "^L                         - search for form-feed

.......typeout occurs here

SMASK 7400,7400                 - set up string mask

ST M A ("^L*20),("^L*400)       - search for 3rd char f-f

.......more typeout here        - only even addresses are real
                                - parts of form-feed pair!
```

In the string search, both the string and the data searched are masked by the string mask.

Example 7:

You just assembled and saved PROG.SV but forgot to use the /P switch to ABSLDR. Fix the CCB (core control block) as follows:

```
.R FUTIL
FI PROG.SV
PROG.SV 0341-+P              - stop output
341.1/ 6203                  - the "CDF CIF" part &
0341.00002\ 6400             - the address
0341.00003\ 0000 400         - change the JSW

WR                           - write the new CCB

SHOW CCB                     - check it this way
CCB:
  SA = 06400,JSW = 0400
  CORE ^P                    - ok, output stopped
```

Example 8:

The CREF listing file for your source file is about 732 blocks long (just over one full DECtape). If you do want to CREF the file onto a DECtape, you must do it either with the /X (do not process literals) switch or else you could use FUTIL to set up the directory with 735 blocks (by starting at block 2) as follows:

```
^R pip
*dtal:</z                    - zero the directory
*^C

.R FUTIL

SE DEV DTA1                  - ** see WARNING below **
1.1/ 0007 2                  - change first block number
6/ 6446 (C-5)                - 5 more blocks
WR                           - write it out
^C                           - now CREF it....
```

WARNING

Do not copy files onto a device that has been fixed this way with FOTP (COPY command) because it writes out a directory of six blocks after the transfers are finished and this will zap blocks 2 through 6 (the first 5 blocks of the first file) after the copy is done. PIP and other processors do not monkey around with the directory and will handle this correctly.

Example 9:

Something is wrong in your system and you have been losing your directory repeatedly. After fixing it up with both PIP and FUTIL, you just want to back it up while you generate your output files onto another device. Since your system device has a total of 6260 (octal) blocks (an RK8E) you back up the directory as follows:

```
.R FUTIL
1.0/ 7714 WR 6251          - transfer blocks up by
2.0/ 7740 WR 6252          - 6250 blocks
3.0/ 7770 WR 6253
4.0/ 0000 3.2/ 0000        - block 3 was last, so
^C                         - all done
```

Shortly after this, everything crashes totally, i.e., directory smashed, system gone from disk. Rebooting from DECtape you use PIP to restore the system area and then use FUTIL to restore the directory:

```
.R FUTIL
SET DEV RKA0               - load non-system device
6251.0/ 7714 WR 1          - transfer by 6250 blocks
6252.0/ 7740 WR 2          - the other way
6253.0/ 7770 WR 3          - the last one

SCAN 0-6250                - do a SCAN for good luck
```

Example 10:

During a SCAN of a device a bad block is found in an important data file and you would like to know just how far the read of that block really succeeded (e.g., on a DECtape, the type of error will determine whether the read will abort immediately or wait until the end of the physical block). The following commands assume that the block number is "bbbb" and set the input/output buffer in FUTIL to zeros before doing the read:

```
bbbb.0/ ?ee AT 07 FATAL READ ERROR - do read to set up

MOD NUM 0-377
bbbb.00000: 0              - set whole buffer to 0

SET DEV same               - set to device now in use

/ ?ee AT 01 FATAL READ ERROR  - force the read again

DUMP OC bbbb               - dump & examine the block
```

This example makes use of the fact that changing the DEVICE resets the status of the buffer without changing its contents. This status includes the block number known and the <something-changed> flag. Therefore the next access to the block causes the block to be re-read without attempting to write it out. Following the second error, as much of the block as possible will have been read into memory and can now be examined for non-zero values (assuming that the data itself was not all zeros). If the read terminated before the end of the block, there should be an obvious separation between the zero and non-zero values.

Example 11:

Your system has a line printer that can output 132 characters per line and 68 lines per page and you would like to change PAL8 and CREF to make use of this to use less paper. Allowing two lines at the bottom of the page, the lines per page should be set to 66 (call this nl). Three changes need to be made to PAL8 to change the global numberr of lines per page (nl), the number of items per column of the symbol table (-nl+1) and the number of symbols per page (3*[nl-1]). One change needs to be made to CREF to change the number of lines per page (nl) and three changes need to be made to change the number of items per line of cross references. Since CREF uses 10 characters for the symbol name and six characters per line number, 19 references can comfortably fit on one line (19*6+10= 124). The following changes to these two programs will increase the number of lines per page and the numbers of items per line in the cross-reference outputs and then update the dates of the two programs in the directory:

```
.R FUTIL FILE PAL8.SV

PAL8.SV 0200-0217 0020 (0016) 1.057 03-APR-76
SET MODE SAVE
1104/ 0070 ^D66                 - global lines per page

1256/ 7711 (-^D65)              - symbol table column size

1273/ 0245 (3*^D65)             - symbols per page

FILE CREF                       - ** SEE NOTE BELOW **
CREF.SV 0220-0234 0015 (0013) 1.065 18-JAN-74

2564/ 7704 (-^D66)              - lines per page as above

2017/ 1102 1366> TAD 2166       - change instructions here

2132/ 1102 1366> TAD 2166       - and here to get new

2166/ 0077 (-^D19)              - references per line

SET MODE NORM                   - reset access mode

1.(57+4)/ 2036 (D)              - change dates of PAL8

(65+4)/ 0624 (D)                - and CREF.

WRITE                           - output the last changes
```

Location 2166 was not used previous to this patch. Note that the first reference to the word in CREF will cause the last block that was modified in PAL8 to be written out. Similarly, the first reference to the directory will cause the last block that was modified in CREF to be written out.

NOTE

These patches were empirically determined and applied to PAL8 V9H and CREF V3C. They have been applied to some other versions of both programs but have not been tested with OS/8 V3D. USE THESE WITH CAUTION!

17.5 PROGRAM EXECUTION AND MEMORY ALLOCATION

The start address is 06400. When the program is started here, it resets the internal CCB buffer, resets the start address to 00200, tests the scope mode status (changing the action of RUBOUT if it is set), performs initialization for the extended date format, attempts to write out the error messages (resetting the ERROR mode control if unsuccessful), tests the BATCH-in-progress status (changing all console I/O to BATCH I/O if it is set) and jumps to 00200. If you want to manually re-start the program after it has been loaded, re-start it at 00200.

The error messages are swapped with the USR, but not in the normal manner, allowing write-locked startup with the loss of the message text. When the program starts execution, it writes the messages onto the system device in the same area used by the USR in swapping. Once this has been done, the USR or error messages need only be read into memory, as needed. In the case where it is not possible to write on the system device, that is, it is write-locked, the messages are discarded, SHORT mode is set permanently, and execution continues without a hitch. Similarly, if an error occurs when reading the messages, SHORT mode is set permanently, and an error is given to warn that this has happened (with no message).

The program uses almost all of the available memory in an 8K PDP-8. It is allocated as follows:

00000-06237	program proper
06240-06577	buffer for arguments
06400-06777	- once only code for chaining
06600-07177	dump device handler area, 2 pages
07277-07577	device handler area, 2 pages
10000-11777	USR area & error messages (swapped)
12000-12577	CCB/header input and test, file output
12600-15700	text strings, lists
15700-16377	string mask, command buffer stack
16400-16577	CCB buffer, 1 page
16600-17177	"dump" device buffer, 2 pages
17200-17577	I/O buffer, 2 pages

The buffer for arguments in field 0 is defined long enough to store 45 numeric string items. The string mask buffer, in field 1, is 66 words long, and the command buffer, also in field 1, is 140 characters long. These lengths were chosen in anticipation of input from console devices with up to 132 characters per line. No checking of any kind is done to protect against overflow of any of these buffers under the assumption that these buffers are large enough for any reasonable input to this program; however, the arrangement of the buffers is set up in such a way that the most valuable data is the farthest distance from a variable buffer.

The expression evaluation stack buffer uses the area in field 1 from the end of the command buffer (approximately location 16130) to the beginning of the CCB buffer (location 16377). This should provide ample room for any expression to fit on one line. Again, no checking to prevent overflow is done.

17.6 COMMAND SUMMARY

SINGLE-CHARACTER commands: ([<n>] = optional <item>)

```
[<l>]/   <l>+   <l>-
[<n>] with      # $ : % & < = > ? @ [ \ ]
$ (ESCAPE) RETURN; LINE FEED ! ^
```

WORD-TYPE commands: (And modifiers, many of which are optional)

```
  ASCII PACKED OS XS240 UNSIGNED SIGNED BCD BYTE OCTAL PDP FPP
 DIR
DUMP        [<format>] <block string>      ([<format>]s above)
LIST        [<format>] <location string>   ([<format>]s above)
MODIFY      [<format>] <location string>   ([<format>]s below)
  ASCII PACKED OS XS240 NUMERIC

WORD        <option(s)> <n>
            UNEQUAL ABSOLUTE MEMREF FROM <l> TO <l>
STRING      <option(s)><number string>
            MASKED ABSOLUTE FROM <l> TO <l>
SMASK       <number string>      e.g., 1,34,0,7700,0,(-1),377

SET         <option>      <setting>
            OUTPUT        OCTAL PDP FPP
            ERROR         LONG SHORT
            FORMAT        <format>
            OFFSET        <l>
            LOWER         <l>
            UPPER         <l>
            DEVICE        <device name[:]>
            DDEV          <device name[:]>
            MODE          NORMAL SAVE LOAD OFFSET
            DMODE         NONE PART ALL
            MASK          <n>
            FILLER        <n>
            TEMP          <n>
SHOW        <option(s)>
            BLOCK CCB ABSOLUTE RELATIVE ODT LOWER UPPER
            MASK SMASK OFFSET MODE DEVICE OUTPUT FORMAT
            HEADER FILLER VERSION ERRORS DDEV
FILE        <file name(s)>
WRITE       [<block>]
SCAN        <block string>
REWIND

OPEN        <file name>
CLOSE

IF          <expression>
END
COMMENT     [<comment line>]
EXIT

EVAL        <expression>      e.g., (1!(S+^D17))*^K15+)C&7600)
            ! & + - * / ( ) C L B F T S R D
```

Numeric Input:

```
^D ^K <digits> "<1 character> '<2 characters>
(...all eval options...)
```

Control Characters:

```
^P ^C ^U ^R RUBOUT ^S ^Q
```

17.7 SINGLE-CHARACTER COMMAND OUTPUT FORMAT SUMMARY

([<n>] = optional numberic item)

Output in octal or octal & symbolic (PDP or FPP):

```
<l>/    /       [<n>]LINE-FEED   [<n>]! [<n>]^ [<n>]
<l>+    <l>-
```

Output in a specified format:

```
[<n>]#          BCD
[<n>]$          OS/8 ASCII
[<n>]:          SIGNED decimal
[<n>]%          BYTE octal
[<n>]&          XS240 format packed ASCII
[<n>]<          OCTAL
[<n>]=          UNSIGNED decimal
[<n>]>          PDP symbolic
[<n>]?          DIRECTORY
[<n>]@          DATE format (extended, in alpha)
[<n>][          ASCII
[<n>]\          FPP symbolic
[<n>]]          PACKED ASCII
[<n>]$          (ESCAPE)  As SET by last SET FORMAT x
```

No output: [<n>];

CHAPTER 18

MAGTAPE/CASSETTE PERIPHERAL INTERCHANGE PROGRAM (MCPIP)

You may use MCPIP to transfer files between standard cassettes or magnetic tapes and other OS/8 system devices, delete those files, and transfer directories. MCPIP allows you to read or write any standard cassette file on a cassette or magnetic tape. In particular, MCPIP can read or write any file created by or to be used by the CAPS-8 system or by the OS/8 system (using any OS/8 device handler). MCPIP can also read or write any magnetic tape file that is in standard cassette file format, that is, a file created by MCPIP or CAPS-8.

You may run MCPIP on any OS/8 system equipped with at least 8K of memory and TA8E cassette or TM8E magnetic tape drives. MCPIP supports any OS/8 system device. Before running MCPIP, you must load the OS/8 cassette or magnetic tape handlers as described in the OS/8 System Generation Notes.

18.1 CALLING AND USING MCPIP

To call MCPIP from the OS/8 system device, type:

```
.R MCPIP
```

in response to the Keyboard Monitor dot. The Command Decoder then prints an asterisk in the left margin of the terminal and waits to receive a line of I/O files and options. MCPIP accepts one input file and performs output to a single output file. It transfers the contents of the input file to the output file in image mode. In response to the asterisk, type an I/O specification of the following form:

```
*outfile<infile/(options) = size
```

Each file specification consists of a device and an optional file name (for file-structured devices). To perform I/O on a given cassette drive, your OS/8 system should be configured with an OS/8 cassette handler for that drive.

The permanent device names for cassettes are CSA0-CSA7. Magnetic tapes have the permanent device names MTA0-MTA7. Permanent device names for other OS/8 devices are listed in the Keyboard Monitor section of Chapter 1. You use these device names in the I/O specification, along with any file name that is necessary. For example, to transfer a CAPS-8 file named DATA01 to the disk, type:

```
*DSK:DATA01<CSA1:DATA01
```

if you have mounted the standard cassette on drive 1 and if your OS/8 system has a handler for drives 0 and 1 (unit 0) with entry point

names of CSA0 and CSA1. If you specify a cassette handler without any file name, MCPIP uses the handler without modification, i.e., it uses the cassette as a non-file structured device similar to a paper tape reader or punch. Thus, the command:

```
*CSA2:<DSK:SI SCO.BN
```

would perform the same operation with MCPIP as the command:

```
*CSA2:<SI SCO.BN/I
```

would perform with OS/8 PIP.

If you specify a magnetic tape handler with a file name, MCPIP considers the magnetic tape as a file-structured device and assumes that it has the same format as a standard cassette.

Since MCPIP performs file transfers for all types, there are no assumed extensions assigned by MCPIP to file names for either input or output files. You must explicitly specify all extensions, where present, except when using the /B option.

Following completion of a MCPIP operation, the Command Decoder again prints an asterisk in the left margin and waits for another MCPIP I/O specification line. You can return to the Keyboard Monitor by typing CTRL/C or by ending a MCPIP specification line with an ALTMODE.

18.1.1 MCPIP Options

Table 18-1 details the options allowed on a MCPIP I/O specification line.

Table 18-1
MCPIP Options

Option	Meaning
/B	Transfer files in special CAPS-8 binary format. If you use the /B option and no extensions are specified, MCPIP assumes .BN for OS/8 files and .BIN for cassette files. If input is from PTR: (high-speed paper tape reader), you must position the paper tape on the leader.
[]	The square bracket "[]" option allows you to specify a decimal file type on a cassette output file. The notation in brackets does not refer to the file sizes in this case. Hence, to create a file with the name CAS50.BI on cassette drive 1 and give it a file type of 3, type: *CSA1:CA550.BI[3]< For output files other than cassette, square brackets have the same meaning as in OS/8 PIP. For information on file types, see the Cassette Programming System User's Manual (DEC-8EOCASA-B-D), Appendix E.

(continued on next page)

Table 18-1 (Cont.)
MCPIP Options

Option	Meaning
/D	Delete the file specified from the output cassette or magnetic tape. The /D option is valid only if the output device is a cassette or magnetic tape. For example: *MTA1:OFILE</D will delete OFILE from the magnetic tape on drive 1.
=n	Specify in the low order 12 bits of n the number of words (characters) per record that occur in the cassette or magnetic tape output file. The low order 12 bits of the n specification may be between 0 and 1000 (octal), inclusive. If not specified, 200 is assumed. You need not specify the = option for cassette or magnetic tape input files because MCPIP will determine the record size from the file's header record. If the output record size specified is greater than 1000 or if an input record size is 0, MCPIP prints an error message since it cannot handle variable-length records. The high order 11 bits of the = option are used to specify the version number for the file. The = option is ignored if the output file is not a cassette or magnetic tape file.
/I	Assume the input device is a cassette drive. You must also specify an input device on the command decoder line, but it is ignored. Use this option when there are no cassette handlers configured into your system. The drive number is specified as an option, for example, /1 represents drive 1. Do not use the /I and /O options in the same command line.
/O	Assume the output device is a cassette drive. You must also specify an output device on the command decoder line, but it is ignored. Use this option when there are no cassette handlers configured into your system. You specify the drive number as an option. Do not use the /I and /O options in the same command line.
/L	Read the input cassette or magnetic tape directory and write it onto the output file. Notice that in this case the input file itself is not transferred, only the directory. The /L option applies only if the input device is a cassette or magnetic tape.
/Z	If you have not specified a file name, you should zero the cassette or magnetic tape on the drive specified as output by writing a sentinel file on it. Every magnetic tape or cassette should be zeroed before you use it for the first time. If you specify a file name (for a cassette or magnetic tape drive), write a sentinel file after the file specified.

Although cassette or magnetic tape file names may have 3-character extensions, OS/8 allows only 2-character extensions. Thus, when looking up a cassette file, although all three characters may be specified, only the first two are significant. For example, CSA0:FILE.PAL might match a file called FILE.PAT. All files on a standard cassette must be unique with respect to the file name and the first two characters in the extension. On output, the third character of the extension is always a space (unless you specify the /B option).

NOTE

If you type CTRL/C while a write operation is in progress on a cassette or magnetic tape, MCPIP writes an end-of-file before returning to the Keyboard Monitor.

18.2 MCPIP ERROR MESSAGES

Error messages that appear while MCPIP is running are listed in Table 18-2. If you specify an output file on a cassette or magnetic tape and a file by that name already exists, the file on the output drive is deleted before any transfer is performed. If MCPIP detects an error while a cassette or magnetic tape output file is open, it tries to close the output file by writing a sentinel file on the output cassette or magnetic tape.

Table 18-2
MCPIP Error Messages

Message	Meaning
CANNOT HANDLE VARIABLE LENGTH RECORDS	The records on the input and output files specified are not the same size. MCPIP cannot handle variable length records.
CLOSE ERROR	MCPIP is not able to close the file. A bad file just created on magnetic tape or cassette must be removed by placing a sentinel file after the preceding file. (See the /Z option.)
device DOES NOT EXIST	The device specified does not exist on the OS/8 system. "Device" is a set of four characters given when MCPIP expected an OS/8 device name such as DTA0.
ENTER ERROR	Error occurred while trying to enter an output file. This message usually means that the cassette or magnetic tape has no sentinel file.
FETCH ERROR	Error occurred while trying to fetch an OS/8 device handler.

(continued on next page)

Table 18-2 (Cont.)
MCPIP Error Messages

Message	Meaning
file NOT FOUND	The file specified cannot be found. "File" is the actual name of the file that was not found.
ILLEGAL * OR ?	Wild card * or ? was specified in a MCPIP command line. MCPIP does not accept the wild card construction.
ILLEGAL SYNTAX	The command line to the Command Decoder contains an illegal character or was incorrectly formatted.
INPUT ERROR	An input error occurred while reading the file.
NO INPUT FILE	No input file was specified when one was required.
NO OUTPUT FILE	No output file was specified when one was required.
OUT-IN	Both the input and output devices were specified as the same cassette or magnetic tape drive.
OUTPUT DEVICE FULL	Either the device or the directory lacks room.
OUTPUT ERROR	Output error - possibly a WRITE LOCKed device, parity error, or attempt to output to a read-only device.
RECORD SIZE TOO BIG	The output record size is greater than 1000 or an input record size is 0.
TOO MANY FILES	More than one output device or more than one input device was specified.

CHAPTER 19

OCTAL DEBUGGING TECHNIQUE (ODT)

ODT allows you to run your program on the computer, control its execution, and make alterations to the program by typing instructions on the keyboard.

19.1 FEATURES

ODT features include location examination and modification, and instructions breakpoints to return control to ODT (breakpoints). ODT makes no use of the program interrupt facility and is invisible to your program.

The breakpoint is one of ODT's most useful features. When debugging a program, allow it to run normally up to a predetermined point, where you may examine and possibly modify the contents of the accumulator (AC), the link (L), or various instructions or storage locations within your program, depending on the results you find. To accomplish this, ODT acts as a monitor to the program.

You decide how far you wish the program to run, and ODT inserts an instruction in your program which, when encountered, causes control to transfer back to ODT. ODT immediately preserves in designated storage locations the contents of the AC and L at the breakpoint. It then prints out the location where the breakpoint occurred and the contents of the AC at that point. ODT will then allow you to examine and modify any location of your program (or those locations containing the AC and L). You may also move the breakpoint and request that ODT continue running your program. ODT will restore the AC and L, execute the trapped instruction, and continue in your program until it encounters the breakpoint again or terminates the program normally.

19.2 CALLING AND USING ODT

Call ODT into use by typing:

```
.ODT
```

in response to the Keyboard Monitor dot. Before you call ODT, you should have a running version of your program in memory. Running ODT disturbs none of your memory because the sections of the program ODT may occupy when in memory remain on the system device and swap back into memory as necessary. ODT uses the Job Status Word of the particular program to determine whether or not swapping occurs. If the program does not use locations 0-1777 in field 0, less swapping occurs during use of the breakpoint feature.

If you are typing any amount of program directly into memory (in octal), the memroy control block of the program may not reflect the true extent of the program. If you make octal additions below location 2000 in field 0, ODT may give erroneous results. You can correct this condition by correcting the Job Status Word, which is location 7746 of field 0. You can examine and change this by using ODT. Location 7745 of field 0 is the 12-bit starting address of the program in memory, and location 7744 contains the field designation in the form 62n3 (where n is the field designation of the starting address).

When using the breakpoint feature of ODT, you should keep the following operating characteristics in mind:

- If a breakpoint is inserted at a location which contains an auto-indexed instruction, the auto-indexed register is bumped immediately after the breakpoint is hit. Thus, when control returns to you in ODT, the contents of the register will be incremented by one. The breakpoint instruction is executed properly, but the index register, if examined, may appear to contain one greater than it should.

- ODT keeps track of the TTY flag and restores the TTY flag when it continues from a breakpoint.

- The breakpoint feature uses locations 4, 5, and 6 in the memory field where the breakpoint is set.

- The breakpoint feature of ODT uses the table of user-defined device names as scratch storage, destroying any device names you may have created. After a session with ODT in which you use breakpoints, give a DEASSIGN command to clear out the user-device name table.

- Do not set breakpoints in the Monitor, in the device handlers, or between a CIF and the following JMP instruction.

You should not use user-defined device names in programs being developed with ODT breakpoints.

If you attempt any operations in non-existent memory, ODT ignores the command and types "?". Thus, if the machine in use has 8K (fields 0 and 1) and you attempt to examine locations in field 2 and above, ODT responds with ?.

ODT should not be used to debug programs that use interrupts. Typing CTRL/C returns control to the Keyboard Monitor; you can save the program on any device.

19.3 COMMANDS

19.3.1 Special Characters

Slash (/) - Open Preceding Location

The location examination character (/) opens the location addressed by the octal number preceding the slash and prints its contents in octal. You can then modify the open location by typing the desired octal number and closing the location. Any octal number from one to five

digits long is legal input. If you enter more than five digits, only the last five entered are accepted by ODT. Typing / with no preceding argument opens the location named last, for example:

```
400/1540
400/1540 2468?
400/1540 02345
/02345
```

Return - Close Location

If you have typed a valid octal number after ODT has printed the content of a location, typing the RETURN key causes the binary value of that number to replace the original contents of the opened location and the location to be closed. If you typed nothing, the location closes but the content of the location does not change, for example:

```
400/6046        location 400 is unchanged.
400/6046 2345   location 400 is changed to contain 2345.
/2345 6046      replace 6046 in location 400.
```

Typing another command will also close an opened register, for example:

```
400/6046 401/6031 2346   location 400 is closed and unchanged
40/6046 401/2346         and 401 is opened and changed to 2346.
```

Line Feed - Close Location, Open Next Location

The LINE FEED key has the same effect as the RETURN key, but it also opens the next sequential location and prints its contents, for example:

```
400/1540                 location 400 is closed unchanged and 401
 is
00401 /2345              opened. User types change, 401 is closed
00402 /7650              containing 1234 and 402 is opened.
```

^ (Shift /N) - Close Location, Take Contents as Memory Reference and Open Same

The up arrow will close an open location just as the RETURN key does. Further, it will interpret the contents of the location as a memory reference instruction, open the referenced location and print its contents, for example:

```
                         3270 symbolically is "DCA, this page,
404/3270 ^               relative location 70," so ODT opens
00470 /4512 0000         location 470.
```

< (Shift /O) Close Location, Open Indirectly

The back arrow will close the location that is currently open, and then interpret its contents as the address of the location whose contents it will print and open for modification, for example:

```
365/3203 <
00203 /3572
035/2 /0216
```

19.3.2 Illegal Characters

Any character that is neither a valid control character nor an octal digit causes the program to ignore the current line and to print a question mark, for example:

```
4:?
                        ODT opens no location.
4U?

406/1136 67K?           ODT ignores modification and closes
/1136                   location 406.
```

19.3.3 Control Commands

nnnnG - Transfer Control to User at Location nnnn

Clear the AC then go to the location specified before the G. The program will initialize all indicators and registers and insert the breakpoint, if any. Typing G alone will cause a jump to location 0.

nnnnB - Set Breakpoint at User Location nnnn

Instructs ODT to establish a breakpoint at the location specified before the B. If you type B alone, ODT removes any previously established breakpoint and restores the original contents of the break location. You may change a breakpoint to another location whenever ODT is in control by simply typing nnnnB, where nnnn is the new location. Only one breakpoint may be in effect at one time; requesting a new breakpoint removes any previously existing one.

You may not set a breakpoint on any of the floating-point instructions that appear as arguments of a JMS.

The breakpoint (B) command does not make the exchange of ODT instruction your instruction, it only sets up the mechanism for doing so. The actual exchange does not occur until you execute a "go to" or a "proceed from breakpoint" command.

When, during execution, your program encounters the location containing the breakpoint, control passes immediately to ODT (via location 0004). ODT saves the C(AC) and C(L) at the point of the interruption in special locations accessible to ODT. Your instruction that the breakpoint was replacing is restored before the address of the trap and the content of the AC are printed. ODT has not yet executed the restored instruction. It will not until you give the "proceed from breakpoint" command. Any user location, including those containing the stored AC and Link, can now be modified in the usual manner. You can also move or remove the breakpoint at this time.

An example of breakpoint usage follows the section "Continue and Iterate Loop..."

A - Open C(AC)

When ODT encounters the breakpoint it saves the C(AC) and C(L) for later restoration. Typing A after having encountered a breakpoint opens for modification the location in which the AC was saved and prints its contents. You may now modify this location in the normal manner (see Slash), and the modification will be restored to the AC when you give the "proceed from breakpoint" command.

Open C(L)

Typing L opens the Link storage location for modification and prints its contents. You may modify the Link location as usual (see Slash), and that modification will be restored to the Link when you give the "proceed from the breakpoint" command.

C - Proceed (Continue) from a Breakpoint

Typing C after ODT encounters a breakpoint causes ODT to insert the latest specified breakpoint (if any), restores the contents of the AC and Link, executes the instruction trapped by the previous breakpoint, and transfers control back to your program at the appropriate location. Your program then runs until ODT encounters the breakpoint again.

NOTE

If you do not encounter a breakpoint set by ODT while ODT is running your program, the instruction that causes the break to occur will not be removed from the program.

nnnnC - Continue and Iterate Loop nnnn Times Before Break

You may wish to establish the breakpoint at some location within a loop of your program. Since loops often run to many iterations, some means must be available to prevent a break from occurring each time ODT encounters the break location. This is the function of nnnnC (where nnnn is an octal number). After ODT encounters the breakpoint for the first time, this command specifies how many additional times the loop will repeat before another break is to occur. The section on the B command describes the break operations.

The following program, which increases the value of the AC by increments of 1, illustrates the use of the Breakpoint command.

```
*200

            0200  *200
00200       7300          CLA CLL
00201       1206  A,      TAD ONE
00202       2207  B,      ISZ CNT
00203       5202          JMP B
00204       5201          JMP A
00205       7402          HLT
00206       0001  ONE,     1
00207       0000  CNT,     0
                  $
*200

A           0201
B           0202
CNT         0207
ONE         0206
0201B
200G
00201 (0;0000
C
00201 (0;0001
C
00201 (0;0002
4C
00201 (0;0006
```

You have now loaded and started ODT. ODT inserts a breakpoint at location 0201. Execution stops here, showing the AC initially set to 0000. The Proceed command (C) executes the program until ODT encounters the breakpoint again (after one complete loop), and shows the AC to contain a value of 0001. Execution continues again, incrementing the AC to 0002. At this point, use the command 4C, allowing execution of the loop to continue 4 more times (following the initial encounter) before stopping at the breakpoint. The contents of the AC have now incremented to 0006.

M - Open Search Mask

Typing M opens for modification the location containing the current value of the search mask and prints its contents. Initially the mask is set to 7777. You may change it by opening the mask location and typing the desired value after the value that ODT printed, then closing the location.

M Line Feed - Open Lower Search Limit

The word immediately following the mask storage location contains the location where the search will begin. Typing the LINE FEED key to close the mask location opens the lower search limit for modification and prints its contents. Initially the lower search limit is set to 0000. You may change it by typing the desired lower limit after the one ODT printed, then closing the location.

M Line Feed - Open Upper Search Limit

The next sequential word contains the location where the search will terminate. Typing the LINE FEED key to close the lower search limit opens the upper search limit for modification and prints its contents. Initially, the upper search limit is the beginning of ODT itself, 7577. You may also change it by typing the desired upper search limit after the one ODT printed, then closing the location with the RETURN key.

nnnnW - Word Search

The command nnnnW (where nnnn is an octal number) will conduct a search of a defined section of memory, using the mask and the lower and upper limits you have specified, as indicated above. Use the word searching operations to determine if a given quantity is present in any of the locations of a particular section of memory.

The search operates as follows: ODT masks the expression nnnn you type preceding the W, and saves the result as the quantity it is searching for. (Do all masking by performing a Boolean AND between the contents of the mask word, C(M), and the word containing the instruction ODT will mask.) ODT then masks each location within your specified limits and compares the result to the quantity it is searching for. If the two quantities are identical, ODT prints the address and the unmasked contents of the matching location, and the search continues until ODT reaches the upper limit.

A search does not alter the contents of any location. The following example is for a search of locations 3000 to 4000 for all ISZ instructions, regardless of what location they refer to (that is, search for all locations beginning with an octal 2).

M/7777 7000 7453/5273 3000	Change the mask to 7000, open lower search limit. Change the lower limit to 3000, open upper limit.
7454/1335 4000 2000W 00005 /2331 00006 /2324 00033 /2575	Change the upper limit to 4000, close location. Initiate the search for ISZ instructions. This section of core has 4 ISZ instructions.

19.4 ADDITIONAL TECHNIQUES

19.4.1 Current Location

ODT remembers the address of the current location, or last location examined, which remains the same, even after you type the commands G, C, and B. You may open this location for inspection by typing the slash (/) character.

19.4.2 Indirect References

When ODT encounters an indirect memory reference instruction or an address constant, open the actual address by typing ^ and < (SHIFT /N and SHIFT /O, respectively).

19.5 ERRORS

The only legal inputs are control characters and octal digits. Any other character will cause ODT to ignore the character or line and to print a question mark. Typing G alone is an error. You must precede G with an address to which control will be transferred. Typing G by itself will cause control to be transferred to location 0.

19.6 PROGRAMMING NOTES SUMMARY

ODT will not turn on the program interrupt, since it does not know if your program is using the interrupt. It does, however, turn off the interrupt when it encounters a breakpoint, to prevent spurious interrupts.

Breakpoints are fully invisible to "open location" commands; however, you may not place breakpoints in locations your program will modify in the course of execution, or ODT will destroy the breakpoint. Use caution in placing a breakpoint between a call to USR function code 10 and the following call to USR function code 11.

If your program does not encounter a trap that ODT set, the breakpoint instruction will remain.

You can use ODT to debug programs using floating-point instructions because the intercom location is 0004 and because you can set breakpoints on a JMS with arguments following.

19.7 SUMMARY OF ODT COMMANDS

Table 19-1 presents a brief summary of the ODT commands. You can input all addresses as 5 digits; they are printed as 5 digits.

Table 19-1
ODT Command Summary

Command	Meaning
nnnnn/	Open location designated by the octal number nnnnn, where the first digit represents the memory field. ODT prints the contents of the location and a space, and waits for you to enter a new value for that location or close the location.
/	Reopen latest opened location.
nnnn;	Deposit nnnn in the currently opened location, close that location, and open the next sequential location for modification. You can deposit a series of octal values in sequential locations through use of the ; character. Multiple ;'s skip a memory location for each ; typed and prepare to insert subsequent values beyond the one(s) skipped.
RETURN key	Close the previously opened location.
LINE FEED key	Close location; open the next sequential location for modification, and print the contents of that location.
n+	Open the current location plus n for modification and print the contents of that location.
n-	Open the current location minus n for modification and print its contents.
↑ or ^ (up-arrow or circumflex)	Close location, take contents of that location as a memory reference, and open the location referenced, printing its contents. NOTE No distinction is made between instruction op-codes when using ^. Thus, all op-codes (0-7) are treated as memory reference instructions. Also, exercise great care when using ^ with indirectly referenced auto-index registers. If you use ^ in this case, the contents of the auto-index register are incremented by one. Check to see that the register contains the proper value before proceeding.

(continued on next page)

Table 19-1 (Cont.)
ODT Command Summary

Command	Meaning
← or _ (back-arrow or underline)	Close location, take contents of that location as a 12-bit address, and open that address for modification, printing its contents.
nnnnnG	Transfer control of program to location nnnnn, where the first digit represents the memory field.
nnnnnB	Establish a breakpoint at location nnnnn, where the first digit represents the memory field. ODT allows only one breakpoint at any given time.
B	Remove the breakpoint.
A	Open for modification the location where ODT stored the contents of the accumulator when it encountered the breakpoint.
L	Open for modification the location where ODT stored the contents of the link when it encountered the breakpoint.
C	Proceed from a breakpoint.
nnnnC	Continue from a breakpoint and iterate past the breakpoint nnnn times before interrupting your program at the breakpoint location.
M	Open the search mask, initially set to 7777, which you can change by typing a new value.
M LINE FEED	Open the lower search limit. Type in the location (4 octal digits) where the search will begin.
M LINE FEED	Open the upper search limit. Type in the location (4 octal digits) where the search will terminate.
nnnnW	Search the portion of core as defined by the upper and lower limits for the octal value nnnn. Search can only be done on a single memory field at a time. See the F command.
D	Open for modification the word containing the data field which was in effect at the last breakpoint. Contents of D always appear as multiples of 10(8) - i.e., 10 means field 1, 20 field 2, etc.

(continued on next page)

Table 19-1 (Cont.)
ODT Command Summary

Command	Meaning
CTRL/O	Stop any printing currently in progress.
F	Open for modification the word containing the field used by ODT in the W (search) command, in the < and ^ (indirect addressing) commands, or in the last breakpoint (depending upon which was used most recently. The contents of F are always expressed as multiples of 10(8) (as in the D command).
RUBOUT key	Cancels previous number typed, up to the last non-numeric character typed.

CHAPTER 20

PERIPHERAL INTERCHANGE PROGRAM (PIP)

Use PIP to transfer files between devices, to merge and delete files, and to list, zero, and compress directories.

20.1 CALLING AND USING PIP

To call PIP from the system device, type:

```
.R PIP
```

in response to the Keyboard Monitor dot. The Command Decoder then prints an asterisk in the left margin of the teleprinter paper and waits to receive a line of I/O files and options. PIP accepts up to nine input files and performs output to a single output file; you generally place options at the end of the command string.

Since PIP performs file transfers for all file types (ASCII, Image or SAVE format, or Binary), there are no assumed extensions PIP assigns to file names for either input or output files. You must specify all extensions.

Following completion of a PIP operation, the Command Decoder again prints an asterisk in the left margin and waits for another PIP I/O specification line. You can return to the Keyboard Monitor by typing CTRL/C or by terminating the specification line with the ALTMODE key.

20.1.1 PIP Options

Table 20-1 details the options allowed on a PIP I/O specification line. Generally, you indicate /A, /B, or /I for each transfer; if you have specified none of these, the system proceeds as if you had typed /A.

Table 20-1
PIP Options

Option	Meaning
/A	Transfer files in ASCII mode. PIP modifies the file as it copies it: it deletes embedded blank tape and rubouts and it reduces leader/trailer code to a standard length. PIP may also do some editing of the input file under control of the /C and /T options (see below).

(continued on next page)

Table 20-1 (Cont.)
PIP Options

Option	Meaning
/B	Transfer files in Binary mode (used for absolute and relocatable binary files). PIP reduces leader/trailer code to a standard length, but it does not recalculate the checksum. NOTE If you combine several absolute binary files into one, indicate the /S option to the Absolute Loader in order for the files to load properly. (The Linking Loader will not load combined files.)
/C	Eliminate trailing blanks. Valid in ASCII mode only.
/D	Delete the old copy of the output file before doing any data transfer. If you do not use /D, PIP will not delete the old copy until it has processed all input. For example: *DTA1:OFILE<DTA2:NFILE/D will first delete file OFILE on DTA1, and then transfer the data from NFILE to a new OFILE. /D is useful when the output device does not have room for both the old file and the new file. You may also use /D to delete up to three files at a time by specifying the files to be deleted as output files and not specifying any input files. For example: *OLDABC,DTA3:FILES/D< This command string deletes OLDABC from DSK and FILES from DTA3.
/E	List directories in extended form (the lengths of the empty files are also listed).
/F	List directories in short form (file names only).
/G	Ignore any errors that occur during a file transfer and continue copying.

(continued on next page)

Table 20-1 (Cont.)
PIP Options

Option	Meaning
/I	Transfer files in image mode. Used to transfer core image (SAVE format) files, and any other files which do not fall into either ASCII or Binary categories. This option always opens the output file even if you specified no input files. Thus, the /I combined with the =n option allows you to substitute a named file for an empty one. For example, suppose you accidentally deleted a 23-block file named IMPORT.PA. You can recover it with the following command: *IMPORT.PA[23]</I=27 Note that 23(10) = 27(8).
=n	Save n extra words per file entry in the directory to contain descriptive information about the file (only the 2 low order octal digits on nnnn are significant). For use with the /Z and /S options only. Typing =1 allows PIP to automatically store the date of the file creation in the directory. (=1 is assumed after /Z or /S options unless otherwise specified. Specifying =0 will still reserve one extra word per entry.) Specifying =100 will reserve no extra words per entry. If you include an = option with an image mode (/I) transfer, the low order 12-bits of the = option specify the desired length with which to close the output file. PIP gives the output file this length except in the following two cases: 1. If the data written is greater than the specified length, PIP gives the output file its correct size. 2. If the length specified is greater than the empty space available, PIP transfers the data but does not close the file. PIP prints the error message: MONITOR ERROR 6 AT xxxx (DIRECTORY OVERFLOW) and control returns to the Keyboard Monitor. PIP does not destroy data in the file following the EMPTY.
/O	Okay to compress files or to zero the directory. When used with the /S or /Z option, /O prevents the messages ARE YOU SURE? and ZERO SYS? from printing. The system assumes you want the /S or /Z option.

(continued on next page)

Table 20-1 (Cont.)
PIP Options

Option	Meaning
/S	Move all files from the input device to the output device, eliminating any embedded empty files. You should explicitly state all device names, as no default devices are assumed. The directory of the output device will contain only those files that appeared on the input device. Whenever a /S is initiated, PIP asks: ARE YOU SURE? Respond with a "Y" if you want the compression; typing any other character aborts the command. NOTE When you use the /S option, PIP reads the output device directory to determine whether it is a system directory. If a system exists on the output device, PIP will preserve that system on the /S transfer. To eliminate the system directory, perform a /Z before the /S. In addition to compressing directories, /S provides a means of copying one device to another. You can copy DECtapes, for example, by compressing one DECtape onto another tape.
/T	Perform the following conversion of special characters: Character — Is Converted To: TAB — enough spaces to reach the next TAB stop (every eighth position) Vertical TAB — 5 LINE FEEDs FORM FEED — 9 LINE FEEDs /T option is valid in ASCII mode only.
/V	Print the current version number of PIP. You should include this option in the first command line entered after you call PIP. PIP prints the version number on the console terminal.

(continued on next page)

Table 20-1 (Cont.)
PIP Options

Option	Meaning
/Y	Copy the OS/8 System Area (records 0, 7-67) between the output and first input file. Both devices must be file-structured devices. If you specified no file name after a device name, the System Area of that device is assumed. If you use the /Z option with /Y, PIP places a zeroed system directory on the output device before the system transfer takes place. A system directory indicates that file storage starts at record 70 rather than record 7.
/X	Zero directory of output device before file transfer. Before using a DECtape for the first time, always use the /Z option to create an empty file directory. No input files are specified, for example: *DTA2:/Z=1 PIP uses one extra word per entry if you specify no "=". Thus, the DATE word is always left available in a new directory. If you attempt to zero the directory of the system device, the message: ZERO SYS? appears. A response of 'Y' will zero the directory; any other response will abort the command and return control to the Command Decoder. NOTE PIP does not ask the question ZERO SYS for a handler that is co-resident with the SYS: handler. For example, if both SYS: and DTA0 are LINCtape 0, a request to zero LINCtape 0 will not produce the question. This is a potentially dangerous command. No data transfer occurs if you do not specify any input files. Thus, as mentioned previously, you can use /Z to zero a directory, and /D to delete a permanent file without creating a file. For the three directory listing options (/E, /F, /L), if you do not specify an output device, the device TTY: is assumed. If you do not specify an input device, device DSK: is assumed.

20.1.2 Examples of PIP Specification Commmands

The following are legal command strings to PIP. When PIP has completed an operation, control returns to the Command Decoder for additional input.

Example 1, ASCII Transfer:

```
.R PIP
*SYS:BLACK<PTR:
```

This command string transfers a tape from the paper tape reader to a file on the system device under the name BLACK. PIP assumes that the input tape is in ASCII format. (Control returns to the Command Decoder; therefore, you need to give the .R PIP command only once.)

Example 2, ASCII File Merge:

```
*DTA3:MERGE<DTA1:FILE1,FILE2
```

This command string instructs PIP to merge the ASCII files FILE1 and FILE2 on DTA1 into one ASCII file, MERGE, on DTA3.

Example 3, Binary Transfer:

```
*BIN.BN<PTR:/B
```

The above command reads a binary paper tape from the paper tape reader and creates a binary file BIN.BN on the device DSK.

Example 4, Image Transfer:

```
*SYS:GAG.SV<PAL8.SV/I
```

PIP transfers the core image file PAL8.SV from the device DSK to GAG.SV on the system device.

NOTE

A problem occurs when you transfer files longer than 255 blocks in Image Mode from a directory device. If you attempt this, the transfer will not end with the real end-of-file, but will continue until you reach the output limit; an error message will occur. For example, trying to transfer FORT.PA or SABR.PA from the directory device using Image Mode will cause this error. Use ASCII mode for all PIP transfers of this type, or use the FOTP program.

Example 5, Directory Listing:

```
*TTY:<ESC>/E
```

This command string produces an extended listing of the device DSK on the Teletype. An extended listing contains all files with their associated lengths and all empty spaces in the directory. For example, an extended listing might appear as follows. (The current

date appears before the file listing provided you have given the DATE command; see the section concerning the Keyboard Monitor for a description of the DATE command.)

```
 2/17/72
EDIT   .SV   12  1/10/72
TEST2        4  1/10/72
ABCD   .DA    1  2/17/72
<EMPTY>      7
TEST2  .RL    4  1/10/72
<EMPTY>    702
 709 FREE BLOCKS
```

The file lengths and number of free blocks are designated as decimal values. The date of file creation is printed if at least one additional information word is present in the directory (refer to the section Additional Information Words in File Directories).

Example 6, Directory Listing:

```
*/F
```

This command produces a directory listing of file names only. Thus, the preceding directory would appear on the teleprinter as follows:

```
 2/17/72
EDIT   .SV
TEST2
ABCD   .DA
TEST2  .RL
 709 FREE BLOCKS
```

Example 7, System Area Transfer:

```
*DTA1:HEAD=/Y
```

PIP transfers records 0 and 7-67 from SYS: to a file named HEAD on DTA1.

Example 8, System Area Transfer:

```
*SYS:=DTA1:HEAD/Y
```

PIP transfers the contents of the file HEAD on DTA1 to the System Area (records 0 and 7-67) of the system device. It also checks the input file for validity before the transfer occurs.

Example 9, System Transfer with Directory Zero:

```
*DTA1:=DTA0:(YZ)
```

This first creates a zero system directory on DTA1, and then transfers the system area from DTA0 to the System Area on DTA1. A system directory indicates that file storage begins at record 70 rather than record 7.

Example 10, System Area Transfer:

```
*DTA1:TRAN<DTA2:TRAN/Y
```

This command string instructs PIP to transfer TRAN from DTA2 to DTA1. Since you used the /Y option, TRAN must be a copy of the OS/8 System Area. However, since transfers of this type involve files on both the I/O devices and not the System Area, PIP treats the transfer as an image transfer, and you can use either the /Y or /I options.

20.2 ADDITIONAL INFORMATION WORDS IN FILE DIRECTORIES

If a device has any additional information words specified in its directory, OS/8 automatically enters the last date specified in a DATE command into the first of the additional information words when you create a file on that device. Dates put into these additional words appear in directory listings. Words after the first are not used by the OS/8 system.

Whenever you give a /Z or /S, you can specify additional words by a /Z=n or /S=n construction. You can change the number of additional words by compressing a device onto itself. The system uses the first additional information word for the file's creation date.

NOTE

DIGITAL initially creates the system with one additional word in the file directory.

20.3 PIP ERROR MESSAGES

Table 20-2 lists the PIP error messages and their meanings.

Table 20-2
PIP Error Messages

Message	Meaning
ARE YOU SURE?	Occurs when using /S option. A response of Y will compress the files.
BAD DIRECTORY ON DEVICE # n	Occurs when: 1. PIP is trying to read the directory, but it is not a OS/8 directory. 2. The output device does not have a system directory; that is, file storage begins at record 7 (occurs during a /Y transfer). n is the number of the file in the input file list.
BAD SYSTEM HEAD	Occurs when you use the /Y option and the area being transferred does not contain OS/8.
CAN'T OPEN OUTPUT FILE	Occurs when: 1. Output file is on a read-only device. 2. No name has been specified for the output file. 3. A /Y transfer to a non-directory device has been attempted. 4. Output file has zero free blocks.

(continued on next page)

Table 20-2 (Cont.)
PIP Error Messages

Message	Meaning
DEVICE # n NOT A DIRECTORY DEVICE	Occurs when: 1. Trying to list the directory of a non-directory device. 2. The input designed in a /Y transfer is not on a directory device. n gives the number of the device in the input list.
DIRECTORY ERROR	Indicates an error has occurred while reading or writing the directory during a /S option. The option is aborted; output is likely to be garbled.
ERROR DELETING FILE	You attempted to delete a file that does not exist. Check that the device name was explicitly given for all files.
ILLEGAL BINARY INPUT,FILE # n	Self-explanatory; n is the number of the file in the input file list.
INPUT ERROR, FILE # n	An input error occurred while reading file number n in the input file list.
IO ERROR IN (file name) -CONTINUING	An error has occurred during a /S transfer. The name of the file being transferred is indicated.
LINE TOO LONG IN FILE # n	In ASCII mode a line has been found greater than 140 characters. Make certain the file is an ASCII file. n is the number of this file in the input list.
NO ROOM FOR OUTPUT FILE	Either the device or the directory lacks room.
NO ROOM IN (file name) -CONTINUING	Occurs during use of the /S option. The output device cannot contain all of the files on the input device. The message is printed for each file which will not fit into the output device. The file name is indicated.
OUTPUT ERROR	Output error - possibly a WRITE LOCKed device, parity error, or attempt to output to a read-only device.
PREMATURE END OF FILE,FILE # n	Occurs in Binary Mode (/B) only. PIP found a physical end-of-file before the final leader/trailer.

(continued on next page)

Table 20-2 (Cont.)
PIP Error Messages

Message	Meaning
SORRY - NO INTERRUPTIONS	Occurs if: 1. You type ^C (CTRL/C) while compressing a file onto itself; the transfer continues. 2. You do a /Y transfer with system device as the output device, or if the transfer has both input and output on the same device.
ZERO SYS?	If you make any attempt to zero the system device directory, this message occurs. Responding with Y causes the directory to be zeroed. Responding with other character aborts the operation.

CHAPTER 21

PIP10

PIP10 is designed to provide file compatibility with the DECsystem-10 computer. PIP10 is capable of transferring files to and from DECsystem-10 formatted DECtapes, and it provides the facilities for transferring ASCII, Image (PAL10 binary output), and sequenced ASCII (LINED output) files.

PIP10 uses an internal DECsystem-10 DECtape routine. This routine optimizes file storage in the same way that the DECsystem-10 Monitor does, thus resulting in the most efficient algorithm for block storage.

PIP10 has the following features:

- Automatically determines which of the specified DECtapes is a DECsystem-10 tape (384(10) words/blocks).
- Works interchangeably on TC08 and TD83 DECtape controllers.
- Reads and writes to DECsystem-10 tapes in both forward and reverse directions on TC08 tapes, forward only on TD83 tapes.
- Keeps the DECsystem-10 DECtape directory in core during the file-copying operations of PIP10, thus eliminating the necessity for rereading the directory. The directories are purged from core when PIP10 reads another command line.
- Permits transfers between two OS/8 devices as well as transfers between two DECsystem-10 tapes.
- Zeroes DECsystem-10 DECtape directories, deletes DECsystem-10 files, and lists DECsystem-10 directories.

You cannot use PIP10 while running the OS/8 BATCH program.

21.1 CALLING AND USING PIP10

To use PIP10, type:

```
.R PIP10
```

PIP10 responds with an asterisk and waits to receive a command line of I/O files and options. The command line must have one output specification and may have from zero to nine input specifications. PIP10 merges multiple input files onto the output file.

A DECsystem-10 file name may have a 0- to 3-character file extension; an OS/8 file name may have a 0- to 2-character extension.

Since PIP10 automatically determines which DECtape mounted is a DECsystem-10 tape, no indication for DECsystem-10 is necessary.

Following completion of a PIP10 operation, the PIP10 command decoder again prints an asterisk in the left margin and waits for another PIP10 I/O command line. To return to the Keyboard Monitor, type CTRL/C.

NOTE

PIP10 uses its own command decoder, not that of the standard OS/8; however, the command decoders are functionally the same.

21.2 HOW TO COPY LARGE FILES WITH PIP10 (SR)

The DIGITAL version of PIP10 V3 cannot copy an OS/8 file greater than 255 blocks long in image mode.

The following patch creates a program called PIP10X (with version number X3); you may use it to copy large OS/8 files in image mode. However, this patch prevents you from copying concatenated input files. Do not use this patch if you are concatenating several OS/8 input files. Concatenate them first with PIP, then use PIP10X.

```
.GET SYS PIP10
.ODT
3236/1034       6201;1642;6211;5244;5700
4317/4026       4030
^C
.SAVE SYS PIP10X
```

21.3 PIP10 OPTIONS

The following table details the various options allowed on a PIP10 I/O command line. The general format for PIP10 command lines is the same as that for the standard OS/8 Command Decoder.

Option	Meaning
/B	Transfer files in DECsystem-10 binary mode. The output device must be a DECsystem-10 DECtape.
/D	Delete the old copy of the output file before continuing the transfer. If you do not use /D, PIP10 copies the file before it deletes the old copy.
/F	List the short form of DECsystem-10 DECtape directory.
/I	Copy in Image mode (compatible with PAL10 binary files) rather than ASCII mode.
/L	List the directory of the input device. This input device must be a DECsystem-10 DECtape. If you specify no output device, TTY is assumed to be the output device.
/P	Preserve LINED sequence numbers in DECsystem-10 format. Sequence numbers are normally deleted.
/Z	Zero the output device directory. The output must be a DECsystem-10 DECtape.

21.4 PIP10 EXAMPLES

The following examples assume that you have mounted a DECsystem-10 DECtape on DTA7. In an actual operation, you may use any unit since PIP10 can access any of the tape drives.

Example 1:

```
*DTA7:FILE.EXT<FILE.EX/Z
```

The command line in Example 1 zeroes the DECsystem-10 directory on DTA7 and transfers FILE.EX from DSK to the DECsystem-10 DECtape on DTA7. If you do not use /Z, make sure that the DECsystem-10 tape has a valid directory on it before you attempt transfers.

Example 2:

```
*DTA7:FILE.EXT<DTA1:P1,PTR:  ,, DTA7:PARZ,TTY:
```

Example 2 merges five input files onto one DECsystem-10 output file (FILE.EXT). The first input file is an OS/8 file (P1) on DTA1; the second and third files are read from the paper tape reader; the fourth is a DECsystem-10 file named PARZ on DTA7; and the fifth is from the terminal. This example shows that input files need not be all OS/8 or all DECsystem-10.

Example 3:

```
*DTA1:FILE.BN[10]<DTA7:FILE.BIN/I
```

The command line in Example 3 copies the DECsystem-10 file (FILE.BIN) in Image mode since the DECsystem-10 file is a binary file. You must use /I to copy DECsystem-10 binaries. Note the use of square brackets [] in the command; they have the same meaning as in the OS/8 command decoder.

Example 4:

```
*DTA7:FILE.EXT</D
```

Example 4 indicates the deletion of a DECsystem-10 file (FILE.EXT) from a device.

Example 5:

```
*DTA7:/L
```

If DTA7 has a DECsystem-10 DECtape mounted, the command line in Example 5 will produce a directory listing of the device.

21.5 ERROR MESSAGES

All errors cause PIP10 to abort the current command and print another asterisk. You can then enter the command correctly. (See Table 21-1.)

Table 21-1
PIP10 Error Messages

Message	Meaning
DEVICE FULL	DECsystem-10 ran out of space on the output file during a transfer.
ERROR DELETING FILE	The output file of a /D command was not found, or an error deleted the file.
FILE NOT FOUND	The requested file was not found on the specified device.
I/O ERROR	I/O device error, for example, parity, write lock, out of paper.
NO SUCH DEVICE	Device name is not legal in this OS/8 system.
NOT OS8 FILE	The output device specified with a /L or /F option was not an OS/8 device or file.
NOT PDP-10 FILE	The output device specified with a /Z option was not a DECsystem-10 tape, or the input device specified with a /L or /F option was not a DECsystem-10 tape.
OUTPUT FILE OPEN ERROR	PIP10 could not open the output file. Check output directory to ensure that enough space exists on the device.
PIP10 CANNOT BE CHAINED TO	Self-explanatory.
SYNTAX ERROR	Invalid PIP10 command line.

CHAPTER 22

RESOURCES (RESORC)

Using RESORC, you can determine the device handlers present on a given OS/8 system. Other information about the handlers is available through the use of RESORC options.

22.1 CALLING AND USING RESORC

To call RESORC from the system device, type:

```
.R RESORC
```

in response to the Keyboard Monitor dot. You may also call RESORC via the CCL command RES (see the CCL section in Chapter 1). The Command Decoder prints an asterisk in the left margin and waits to receive a line of I/O files and options. RESORC accepts up to nine input files and performs output to a single output file; you generally place options at the end of a command string.

The output specification is the device where you are sending the RESORC listing (specifying a file name and extension is optional). If you do not specify an output device, TTY is assumed. If no file name is specified, RE is assumed. If you do not specify a file name extension, .LS is assumed.

The input specification may be one of three types:

- No input specification
  If you do not enter an input specification, the OS/8 system device is assumed.

- A device name only (dev:)
  If the input specification is a device name only, the device must be file-structured and is presumed to contain a valid OS/8 directory and Keyboard Monitor. The device handlers built into the system on that device are the ones RESORC lists. These handlers are not available to you unless you bootstrap onto the specified device (see the BOOT program in this chapter).

- A device and a file name (dev:file.ex)
  If you use this type of input specification, the file must be a system-head file. (The /Y option in PIP creates such files which are copies of the system portions of devices.) If you specify no file name extension, the extension .SY is assumed. RESORC prints the handlers in the system that were saved on the specified file. System-head files are 50 (decimal) blocks long.

22.2 RESORC OPTIONS

RESORC has three operating modes specified by options in the command line. These modes are:

Option	Mode
/E	Extended mode -- detailed handler information
/F	Fast mode -- 1-line printout (default)
/L	Limited mode -- 3-column printout

22.2.1 Fast Mode (/F Option)

If you specify the /F option in a RESORC command line, or if you specify no options, RESORC prints the permanent device names for handlers on the system. If RESORC cannot determine the ASCII device name for one of the devices, it prints the internal octal representation of the device name and encloses it in parentheses. (The OS/8 Software Support Manual includes this octal representation.) For example:

```
.R RESORC
*/F
SYS,DSK,DTA2,DTA0,DTA1,(4667),TTY,LPT
```

The first two devices are always SYS and DSK. When you use the fast mode, the devices are separated by commas and listed in order of their internal device numbers.

22.2.2 Limited Mode (/L Option)

If you use the /L option in a RESORC command line, RESORC prints the handler information in three columns. For example:

```
.R RESORC
*/L

128 FREE BLOCKS

NAME  TYPE  USER
SYS   RK8E
DSK   RK8E  IN
DTA0  TC08  0
TTY   TTY
LPT   LPTR  LPT

OS/8 V3F
```

Preceding the table of device names, RESORC prints the number of free blocks on the device. This information is not available for system-head.

The first column (NAME) lists the permanent names of devices on the system. The second column (TYPE) lists the physical type of the handler. OS/8 assigns a unique number to each type of device. RESORC associates this number with a name as listed in Table 22-1. Note that different devices which are similar in function have the same internal type code. For example, line printers LP08, LS8E, and L645 have an internal code of 04.

The third column (USER) lists the name given to the device with the Monitor ASSIGN command. If RESORC cannot determine the name from the internal octal, it prints the octal code enclosed in parentheses.

Table 22-1
RESORC Device Types

Internal Type Code	RESORC Name	Explanation
00	TTY	Console terminal
01	PTR	Paper tape reader
02	PTP	Paper tape punch
03	CR8E	Card reader
04	LPTR	Line printer
05	RK8	RK8 disk
06	RF08	RF08 disk (1 platter)
07	RF08	RF08 disk (2 platter)
10	RF08	RF08 disk (3 platter)
11	RF08	RF08 disk (4 platter)
12	DF32	DF32 disk (1 platter)
13	DF32	DF32 disk (2 platter)
14	DF32	DF32 disk (3 platter)
15	DF32	DF32 disk (4 platter)
16	TC08	TC08 DECtape
17	LINC	LINCtape
20	TM8E	Magnetic tape
21	TD8E	TD8E DECtape
22	BAT	Batch input handler
23	RK8E	RK8E disk
24	NULL	NULL handler
27	TA8E	Cassette
30	VR12	PDP-12 scope

DIGITAL reserves codes 25-26 and 31-37 for future use. Codes 40-57 are reserved for user handlers.

22.2.3 Extended Mode (/E Option)

When you use the /E option in a command line, RESORC provides more detailed information about the handlers configured into the system. The /E option produces a table with the following headings.

Heading	Meaning
#	Internal device number for the handler. If a number is missing, there is no internal number for this handler.
NAME	Permanent device name for the handler. If RESORC cannot determine the name, it prints the internal coding.
TYPE	Type of device as listed in Table 22-1.
MODE	One or more of the following three letters: R The handler may be used for reading. W The handler may be used for writing. F The handler controls file-structured devices.

Heading	Meaning
SIZ	The size of the device in decimal OS/8 blocks. This is only applicable for file-structured devices.
BLK	The block on the system device where this handler resides. If this number is followed by a +, this indicates that the handler is two pages long. If this entry is SYS, the handler is permanently resident in core location 07600.
KIND	This entry tries to differentiate the handler more specifically than the TYPE column. Since several devices of the same type have the same device code, there may be several handlers for the same device. If the device type has only one handler, this entry may be blank. The KIND specification has no meaning for user-written handlers. The kinds of handlers that may be on the system are as follows.

Kind	Type	Description	How Identified
AS33	TTY	1-page handler	by number of pages
KL8E	TTY	2-page handler	by number of pages
KS33	PTR	low-speed reader	by IOT codes
PT8E	PTR	high-speed reader	by IOT codes
KS33	PTP	low-speed punch	by IOT codes
PT8E	PTP	high-speed punch	by IOT codes
026	CR8E	DEC-026 card codes	by table codes
029	CR8E	DEC-029 card codes	by table codes
LP08	LPTR	old LP08 handler	location dependent
LS8E	LPTR	old LS8E handler	location dependent
LPSV	LPTR	LP08/LS8E/LV8E handler	location dependent
LV8E	LPTR	LPSV altered for LV8E	location dependent
L645	LPTR	Anelex line printer	location dependent

U — Unit -- the particular unit number of a multiple unit device handler. For example, the RK8E disk can have as many as four physical drives (0, 1, 2, 3) on an OS/8 system. OS/8 considers the disk cartridge in each drive as two logical units. The lower half is the A unit and the upper half is the B unit. Thus drive 2 consists of two logical units called A2 and B2.

Since the U column in the printout has space for only one character, RESORC numbers the logical units from 0 to 7. The following table shows the correspondence between the U printout, the logical unit, and the physical device, for RK8E.

U	Logical Unit	Physical Device
0	A0	0
1	B0	0
2	A1	1
3	B1	1
4	A2	2
5	B2	2
6	A3	3
7	B3	3

V	Version number (letter) of handler. No entry means the handler predates OS/8 Version 3. Version numbers are of the form A-Z. The 6-bit of the ASCII representation of the handler version letter resides in the handler's entry point location. For example, a handler with a version A has a representation of 01. (See Appendix A for a list of the 6-bit octal codes.)
ENT	The relative entry point of the handler.
USER	Same as for /L option. Your current name for the handler as assigned by the Monitor ASSIGN command.

In addition to the preceding, the /E option also provides the following information. If you specified a device, as opposed to a system-head file, RESORC prints:

- number of files in directory
- number of blocks used
- number of directory segments used
- number of free blocks
- number of empties or a blank to indicate a single empty
- number of additional information words

RESORC also lists the following:

- number of free device slots
- number of free block slots
- version number of Monitor if device is a system device

```
.R RESORC
*/E

164 FILES IN 1025 BLOCKS USING 6 SEGMENTS
2167 FREE BLOCKS (14 EMPTIES)

#  NAME TYPE MODE SIZ BLK KIND U V ENT USER
01 SYS  RK8E RWF 3248 SYS       0 B  07
02 DSK  RK8E RWF 3248 SYS       0 B  07
03 DTA0 TD8E RWF  737 16+ TD8A  0 A  10
04 DTA1 TD8E RWF  737 16+ TD8A  1 A  14
05 RKB0 RK8E RWF 3248 SYS       1 B  21
06 TTY  TTY  RW       17+ KL8E    C 176
07 PTP  PTP  W        20  PT8E    A  00
10 PTR  PTR  R        20  PT8E    A 112
11 LPT  LPTR W        21  LPSV    B  03

FREE DEVICE SLOTS: 06,  FREE BLOCK SLOTS: 04
OS/8 V3F
```

22.3 RESORC ERROR MESSAGES

Table 22-2 lists messages that may appear during a RESORC operation.

Table 22-2
RESORC Error Messages

Message	Meaning
?BAD DIRECTORY	Input device directory cannot be read.
%BAD MONITOR	The input device may be a system device but the Monitor cannot be accessed.
%DEV IS NOT FILE STRUCTURED	The input device specified is not a file-structured device, e.g., PTR.
?INPUT ERROR	An input error occurred during a RESORC operation.
%NON SYSTEM DEVICE	The input device specified in a RESORC command line is not an OS/8 system device.
%NOT A SYSTEM HEAD	The file name specified is not a system-head file.
?OUTPUT DEVICE FULL	The output device specified does not have enough room to copy the RESORC file.
?OUTPUT DEVICE IS READ ONLY	The output device specified is a read-only only device, for example, PTR.
?OUTPUT ERROR	An error occurred while attempting to output during a RESORC operation.
?TTY DOES NOT EXIST	You did not specify an output device in the RESORC command line, and the TTY handler does not exist on the OS/8 system. See the BUILD chapter for instructions on inserting TTY handlers.

CHAPTER 23

RKLFMT DISK FORMATTER PROGRAM

The RK8E/RK8L disk formatter program writes and checks the format of the complete disk cartridge. Only standard DIGITAL surface format is available (that is, sectors numbered in the normal numerical sequence 0, 1, 2, 3, 4, 5, etc.). RKLFMT occupies locations 0000 to 4177 of the current field.

The RK8L control, which can control up to 8 drives, will not run with the DW8E bus adapter; the RK8L control uses IOT0 for extended drives 4-7 (not available on the DW8E).

RKLFMT requires the following hardware:

- PDP-8/E, 8/F, 8/M or 8/A Computer
  Other family of 8-compatible computer with necessary DW8E bus adapter for RK8E control only.
- At least 4K of read/write memory, and at least 8K of memory is needed for operation of the console package.
- ASR-33 teletype or equivalent
- RK8E disk control or RK8L disk control
- RK05J or RK05F disk drive(s)

NOTE

The RK05F drive operates as two separate units. When answering questions for each separate unit, specify: DSK0?, DSK1?, DSK2?, etc.

23.1 RUNNING THE PROGRAM

To format an RK05, type the following command:

```
.R RKLFMT
```

Mount the disk (write enabled) and enter the instructions that follow.

If the formatter program fails to operate correctly, run the following programs:

- All basic and extended memory diagnostics
- For the RK8E control, run the RK8E diskless control test and the RK8E drive control test.
- For the RK8L control, run the RK8L instruction test.

23.2 STANDARD TEST PROCEDURES

To run the formatter program, follow the procedure in Section 23.3. The following two procedures describe the drive setup procedure for the RK05F and the drive cartridge mounting procedure for the RK05J.

23.2.1 RK05J Drive Cartridge Mounting Procedure

The cartridge mounting procedure for the RK05J disk drive is listed below. Any deviation results in an error condition.

1. Set switch labeled RUN/LOAD to the LOAD position.
2. Turn AC power on.
3. Check that the light labeled PWR is on.
4. Wait for the light labeled LOAD to come on.
5. Verify that the lights labeled RDY, ON CYL, FAULT, WT, and RD are off.
6. Open access door.
7. Insert cartridge.
8. Close access door.
9. Set switch labeled RUN/LOAD to the RUN position.
10. Wait for lights labeled RDY and ON CYL to come on.
11. Toggle the switch labeled WT PROT and check that the light labeled WT PROT goes on and off.
12. Toggle the switch labeled WT PROT until light labeled WT PROT goes off.
13. Check that the lights labeled FAULT, WT, RD, and LOAD are off.

23.2.2 RK05F Drive Setup Procedure

The drive setup procedure for the RK05F disk drive follows. Any deviation results in an error condition.

1. Set switch labeled RUN/LOAD to the LOAD position.
2. Turn AC power on.
3. Check that the light labeled PWR is on.
4. Wait for the light labeled LOAD to come on.
5. Check that the lights labeled RDY, ON CYL, FAULT, WT and RD are off.
6. Set switch labeled RUN/LOAD to the RUN position.
7. Wait for the lights labeled RDY and ON CYL to come on.

8. Toggle the switch labeled WT PROT and verify that the light labeled WT PROT goes on and off.

9. Toggle the switch labeled WT PROT until the light labeled WT PROT goes off.

10. Verify that the lights labeled FAULT, WT, RD, and LOAD are off.

23.3 FORMAT PROGRAM

1. Make all drives ready to be formatted:

   For RK05J drives, use the RK05 drive mounting procedure (23.3.1).

   For RK05F drives, use the RK05 drive setup procedure (23.3.2).

2. Set switch labeled RUN/LOAD to the LOAD position on all drives that you are not formatting.

   The TTY will type the following program name, information, and questions.

   ```
   RK8E/RK8L DISK FORMATTER PROGRAM
   ```

   For each question type Y for YES or N for NO.

   ```
   FORMAT DISK 0? (type Y or N)

   FORMAT DISK 1? (type Y or N)

   FORMAT DISK 2? (type Y or N)

   FORMAT DISK 3? (type Y or N)

   FORMAT DISK 4? (type Y or N)

   FORMAT DISK 5? (type Y or N)

   FORMAT DISK 6? (type Y or N)

   FORMAT DISK 7? (type Y or N)
   ```

   The program then types the following question on the TTY:

   ```
   ARE YOU SURE?
   ```

3. Typing N repeats all the previous questions. Typing Y executes the operation selected.

4. Program execution is approximately 80 seconds for each disk drive. After the program has formatted and checked all disks selected, the TTY types the following pass-complete message and question.

   ```
   RK8E/RK8L DISK FORMATTER PASS COMPLETE
   FORMAT SAME DISK(S) AGAIN?
   ```

5. If you want to repeat the operation selected, type Y. Typing N repeats the initial start-up questions.

23.4 ERRORS

When a recoverable error occurs, the TTY prints an ERROR HEADER and error information pertaining to the failure. Possible error headers are:

```
DISK DATA ERROR
READ STATUS ERROR
WRITE STATUS ERROR
RECALIBRATE STATUS ERROR
```

After the TTY types the error header, it prints some of the following error information pertaining to the failure.

PC: Program Location of Failure
GD: Expected Information
EX: Extended Drive Bit
CM: Software Command Register
ST: Contents of Status Register
DA: Software Cylinder, Surface, and Sector Register
CA: Initial Current Address
AD: Address of Data Break
DT: Data Found During Data Break

After the TTY types the error information, it types one of the following questions, asking the error recovery desired.

1. If the error was a recalibrate error, TTY types the following question.

   ```
   TRY TO RECALIBRATE SAME DISK AGAIN?
   ```

   Typing a Y repeats the recalibrate sequence on the disk in error. Typing N moves the program to the next available disk.

2. If the error was a write error the TTY types the following question:

   ```
   TRY TO FORMAT SAME CYLINDER AGAIN?
   ```

   Typing Y repeats the write sequence on the current cylinder. Typing N moves the program to the next sequential cylinder.

3. If the error was a read or check error, the TTY types the following question:

   ```
   TRY TO CHECK SAME CYLINDER AGAIN?
   ```

   Typing a Y repeats the read and check sequence on the current cylinder. Typing N moves the program to the next sequential cylinder.

23.5 PROGRAM DESCRIPTION

The formatting is actually a function of the RK8E or RK8L control and drive logic. The program writes data on every sector in the WRITE ALL mode, then checks the data while in the READ DATA mode to verify that the header words written on every sector are also correct. The READ DATA MODE automatically performs a check header function.

The first two words of every sector are set to the absolute disk address (that is, command register bits 9-11 and cylinder, surface, and sector bits 0-11, respectively). The remainder of the data area is set to all zeros when the data is written. Only the first two words of every sector (that is, the addressing information) are checked when data is read in the READ DATA mode.

23.6 CONTROL CHARACTERS

Use control characters to give the operator the ability to perform the following functions.

NOTE

The program will respond to the control character in five seconds or less.

CTRL/C Starts the monitor at location 7600.

CTRL/R Restarts the program.

CTRL/E Continues the program from an error if allowed by the diagnostic or from a waiting statement.

CTRL/L Switches the terminal messages from the display to a line printer. To restore the messages on the terminal, type CTRL/L again. If no printer is available and you type CTRL/L, the console package will wait for CTRL/C or CTRL/R. The CTRL/L sends output to the line printer and the program attempts to continue as if you typed a CTRL/E.

CTRL/D Allows you to change the switch register during program operation. Typing this character results in an interrogation of the switch register question.

CTRL/S Stops program execution and waits in a loop for a continue. The only way to continue is to type a CTRL/Q, R or C. This is a nonprinting character.

CTRL/Q Causes continuation of a program after you type a CTRL/S. This is a nonprinting character.

23.7 MISCELLANEOUS

23.7.1 Waiting Message

The waiting message gives you time to decide what control character to type. This message appears at the end of a pass message if the halt-on-pass bit is set. You may now use the control characters to perform the needed function.

The waiting message is printed after an error message if the halt-on-error bit is set. Here again you may use the control characters.

The waiting message is printed if operator intervention is required.

23.7.2 End of Pass

The normal program pass complete as described in Section 23.4 is used.

23.7.3 Errors

The standard error reports described in Section 23.5 are used.

23.7.4 Location Changes

You can change the following location to meet the specific need to modify the diagnostic.

3637 Is the location set for the number of filler characters after a CRLF set to four (4)

CHAPTER 24

RXCOPY PROGRAM

You can use the RXCOPY program to copy or transfer the entire contents and system head of one RX floppy disk to another RX floppy disk. Use this program only with RX permanent device names or a user-defined name that you have assigned to an RX device. Specifying file names in the I/O specification line results in an error message.

To load and run RXCOPY, type:

```
.R RXCOPY
*output dev:<input dev:/options
```

Example:

```
.R RXCOPY
*RXA1:<SYS:
```

When you have loaded RXCOPY and entered the I/O specification line at the keyboard, the program copies the input device to the output device on a sector-by-sector basis. When the operation is complete, the Monitor dot appears on the screen, and the specified output device becomes an exact duplicate of the input device.

Table 24-1 lists the options for use with the RXCOPY program. These options modify the RXCOPY operation.

Table 24-1
RXCOPY Options

Option	Meaning
/P	Pause and wait for your response before and after execution of RXCOPY program.
/N	Copy the contents of one device to another but don't check them for identical contents unless otherwise specified.
/M	Check both devices for identical contents and list the tracks and sectors that do not match but do not perform a transfer unless otherwise specified.
/R	Read every block on the specified device and list the bad tracks and sectors but do not perform a transfer unless otherwise specified.
/V	Print the current version number of the RXCOPY program.

If you specify no options, RXCOPY assumes both the /N and /M options.

If an error occurs during the execution of RXCOPY, RXCOPY aborts the current job and control returns to the Monitor.

Table 24-2 lists the RXCOPY error messages and their meanings.

Table 24-2
RXCOPY Error Messages

Message	Meaning
NO INPUT DEVICE	No input device is specified.
CAN'T LOAD INPUT DEVICE	The name of the input device specified in the command line is not a permanent device name.
CAN'T LOAD OUTPUT DEVICE	The name of the output device specified in the command line is not a permanent device name.
COMPARE ERROR	When using the /M option all the areas that do not match are printed as COMPARE ERRORS. Since this is a non-fatal error, the RXCOPY operation continues.
INPUT DEVICE READ ERROR	Bad input, bad tracks or sectors. Since this is a non-fatal error, the RXCOPY operation continues.
OUTPUT DEVICE READ ERROR	Bad data on output device, tracks and sectors bad. Since this is a non-fatal error, the RXCOPY operation continues.
OUTPUT DEVICE WRITE ERROR	Fatal output error. Since this is a non-fatal error, the RXCOPY operation continues.

CHAPTER 25

SET PROGRAM

With the SET program you can modify the operating characteristics of OS/8 according to the attributes you specify, and you can make frequently required standard changes to system programs, especially I/O handlers. You can identify these changes by specifying certain attributes in the SET command string, which has the following format:

```
.R SET
#SET device [NO] attribute [argument]
```

where:

SET	is the operation you are performing.
device	indicates the handler of the device you want modified.
[NO]	indicates that the attribute specified does not apply. You cannot use [NO] with every attribute.
attribute	is the characteristic you are modifying. (See Table 25-1.)
[argument]	is an optional parameter that you must supply for certain SET commands.

SET error messages are listed in Table 25-2.

Table 25-1
SET Command Attributes

TTY	Card Reader	Mag Tape	SYS	LPT	Any Device
ARROW CODE n COL n ECHO ESCAPE FILL FLAG HEIGHT m LC PAGE PAUSE n SCOPE TAB WIDTH n	CODE n	PARITY x FILES	INIT xxxxx OS8 OS78	LA78 LA8A LC LV8E WIDTH n	FILES DVCODE LOCATION n=m READONLY VERSION x BLOCK b

Table 25-2
SET Error Messages

Message	Meaning
?SYNTAX ERROR	Incorrect format used in SET command or NO specified when not allowed.
?UNKNOWN ATTRIBUTE FOR DEVICE dev	An illegal attribute was specified for the given device.
?CAN'T -- DEVICE IS RESIDENT	No modifications are allowed to the system handler.
?CAN'T -- OBSOLETE HANDLER	The handler has an old version number.
?CAN'T -- UNKNOWN VERSION OF THIS HANDLER	The version of the handler is not one recognized, possibly because it is a newer version.
?ILLEGAL WIDTH	A width of 0 or a width too large was specified; or, for the TTY, a width of 128 or one not a multiple of 8 was specified.
?NUMBER TOO BIG	The number specified was out of range.
?CAN'T -- DEVICE DOESN'T EXIST	A nonexistent device was referenced.
?I/O ERROR ON SYS:	

25.1 TERMINAL ATTRIBUTES

25.1.1 Arrow

Specifying this attribute causes each control character the KL8E handler typed to be printed in the form:

```
^x
```

where:

^ indicates that you are typing a control character, and

x specifies which control character you are typing (100 + code for control character).

Format:

```
.SET TTY [NO] ARROW
```

Example:

```
.SET TTY ARROW
```

If you type a CTRL/E character, by the KL8E handler, an ^E is printed on the terminal. Note that ARROW is the default.

Using this attribute with the NO modifier causes each control character the KL8E handler typed to print with no modification.

Example:

```
.SET TTY NO ARROW
```

Now if the KL8E handler types a CTRL/E character, it will send a CTRL/E (ASCII code 5) to the terminal. The result is that no character is printed.

NOTE

On some terminals, the arrow (^) is replaced by the circumflex (^).

25.1.2 CODE n

where n is an octal number in the range

```
1 <n<77
```

This command changes the internal IOT code for keyboard to n. The internal device code for the teleprinter is set to n+1. For example, if you have a VT05 hooked to your system with device codes of 40 and 41, you would type SET TTY CODE 40. SET will not permit the NO restriction.

Example:

```
.SET TTY CODE = 3
```

25.1.3 COLumn n

Specifying this attribute changes the default number of columns used to print the directory (using the DIRECT command) to the decimal number you specify as n. The initial default number of columns is equal to one and the decimal number you specify should be in the range of 1-7.

Format:

```
.SET TTY COL n
```

Example:

```
.SET TTY COL 3
```

Specifying this attribute does not change the behavior of the KL8E handler. Also, you may not use the NO modifier with this attribute.

25.1.4 ECHO

Specifying this attribute causes all TTY characters typed at the keyboard as input or received on the terminal as output to be printed. Specifying this attribute affects the KL8E handler only and does not affect character echoing by the Keyboard Monitor.

Format:

```
.SET TTY [NO] ECHO
```

Example:

```
.SET TTY ECHO
```

If you do not want character echoing to take place, use the NO modifier in the command line. If you specify the NO modifier, all TTY characters on input or output are not printed and do not appear on the terminal screen.

Example:

```
.SET TTY NO ECHO
```

25.1.5 ESCape

Specifying this attribute causes the escape character (ASCII code 33) to print as a control character (see also ARROW attribute).

Format:

```
.SET TTY [NO] ESC
```

Example:

```
.SET TTY ESC
```

Specifying the NO modifier in the command line causes escape to print as a dollar sign ($).

Example:

```
.SET TTY NO ESC
```

The ARROW attribute can also affect escapes. Specifying NO ARROW sends escapes to the terminal with no modification. This is useful for sending escape sequences to a CRT terminal.

25.1.6 FILL

Specifying this attribute types two fill characters following a tab. You should use this attribute only with the TAB attribute.

Format:

```
.SET TTY [NO] FILL
```

To remove these fill characters, use the NO modifier in the command line.

Example:

```
.SET TTY NO FILL
```

25.1.7 FLAG

When you specify this attribute, the handler flags lower-case characters on output by printing them as upper-case characters preceded by a quote.

Format:

```
.SET TTY [NO] FLAG
```

Example:

```
.SET TTY FLAG
```

If you want to remove the quote preceding upper case characters, use the NO modifier in the command line.

Example:

```
.SET TTY NO FLAG
```

25.1.8 HEIGHT m

Specifying this attribute changes the number of lines that are printed on the terminal between pauses. The default value of m is 24 lines.

Format:

```
.SET TTY HEIGHT m
```

Example:

```
.SET TTY HEIGHT 12
```

This attribute has no meaning unless you also specify the PAUSE attribute.

25.1.9 LC

When you specify this attribute, the KL8E handler accepts lower case characters on input.

Format:

```
.SET TTY [NO] LC
```

Example:

```
.SET TTY LC
```

Specifying the NO modifier in the command line converts lower-case characters on input to upper case.

Example:

```
.SET TTY NO LC
```

25.1.10 PAGE

Specifying this attribute adds both the CTRL/S and CTRL/Q features to the keyboard monitor.

Format and Example:

```
.SET TTY PAGE
```

When used with the NO modifier, this attribute removes the CTRL/S and CTRL/Q features.

Example:

```
.SET TTY NO PAGE
```

25.1.11 PAUSE n

Specifying this attribute sets the pause time between terminal output frames to the decimal number you specify as n. The time depends on the cycle time of your machine.

Format:

```
.SET TTY PAUSE n
```

Example:

```
.SET TTY PAUSE 5
```

If you want no pause to take place, specify either the NO modifier in the command line or zero as n.

Example:

```
.SET TTY NO PAUSE
```

or

```
.SET TTY PAUSE 0
```

25.1.12 SCOPE

When you specify this attribute, the characters you erase with the rubout or delete key disappear from the CRT screen. You should not specify this attribute if you do not have a CRT.

Format and Example:

```
.SET TTY SCOPE
```

25.1.13 TAB

When you specify this attribute, the handler prints real tabs (ASCII code 211). You can use this only if your handler has the TAB feature.

Format:

```
.SET TTY [NO] TAB
```

Example:

```
.SET TTY TAB
```

If your handler does not have the TAB feature, use the NO modifier in the command line. When you specify the NO modifier, the handler simulates all tabs as spaces.

Example:

```
.SET TTY NO TAB
```

25.1.14 WIDTH n

Specifying this attribute changes the width of the terminal to the decimal number you specify as n. The decimal number you specify should be a multiple of eight and in the range of 1-255. However, n must not be 128. If your TTY handler does not have the tab feature, the width you specified in the command line may not be your final result. You may not use the NO modifier with this attribute. Placing an equal sign (=) between the attribute and the decimal number you specify is optional.

Format:

```
.SET TTY WIDTH n
```

Example:

```
.SET TTY WIDTH 64
```

25.2 CARD READER ATTRIBUTES

25.2.1 CODE n

When you specify this attribute, the card reader uses the card code you specify.

Format:

```
.SET CDR CODE n
```

where:

n is a decimal number having a value of either 026 or 029.

Example:

```
.SET CDR CODE 026
```

You may not use the NO modifier with this attribute.

25.3 MAGNETIC TAPE ATTRIBUTES

25.3.1 PARITY x

When you specify this attribute, the parity check becomes either even or odd.

Format:

```
.SET MTxx:PARITY x
```

Example:

```
.SET MTA0:PARITY EVEN
```

You may not use the NO modifier with this attribute.

25.3.2 FILES

When you specify this attribute, the handler will not issue an automatic rewind when referencing block 0.

Format:

```
.SET MTxx [NO] FILES
```

Example:

```
.SET MTA1:FILE
```

If you want the automatic rewind to take place when block 0 is referenced, use the NO modifier in the command line.

Example:

```
.SET MTA0:NO FILES
```

25.4 SYSTEM ATTRIBUTES

25.4.1 INITIAL xxxxx

When you specify this attribute, the system device executes the command you specify as when the system is bootstrapped. This command can contain a maximum of five characters excluding a RETURN key.

Format:

```
.SET device INIT xxxxx
```

Example:

```
.SET SYS INIT HELP
```

If you do not specify xxxxx, @INIT is assumed, and the system executes the command in the file INIT.CM when bootstrapped. You must create the INIT.CM file prior to bootstrapping.

If you do not want the system to execute special commands at system bootstrap, use the NO modifier in the command line. When you specify the NO modifier, the system prints the monitor dot immediately after bootstrapping.

Example:

```
.SET SYS NO INIT
```

If you specify an initial command and you have bootstrapped the system, this command destroys anything previously in memory.

25.4.2 OS8

Specifying this attribute modifies the system handler to be OS/8.

Format and example:

```
.SET SYS OS8
```

You may not use the NO modifier with this attribute.

25.4.3 OS78

Specifying this attribute modifies the system handler to be OS/78.

Format and example:

```
.SET SYS OS78
```

You may not use the NO modifier with this attribute.

25.5 LINE PRINTER ATTRIBUTES

25.5.1 LA78

Specifying this attribute modifies the LPSV handler to handle an LA78 line printer.

Format and example:

```
.SET LPT LA78
```

25.5.2 LA8A

Specifying this attribute restores the LPSV handler to its original state.

Format and example:

```
.SET LPT LA8A
```

25.5.3 LC

When you specify this attribute, the handler prints lower-case characters. You may use this attribute only with line printers that can print lower-case characters.

Format:

```
.SET LPT:[NO] LC
```

Example:

```
.SET LPT:LC
```

Specifying the NO modifier in the command line converts lower-case characters to upper case prior to printing.

Example:

```
.SET LPT:NO LC
```

25.5.4 LV8E

Specifying this attribute modifies the LPSV handler to work on an LV8E line printer.

Format:

```
.SET LPT:[NO]LV8E
```

Example:

```
.SET LPT:LV8E
```

When you specify the NO modifier in the command line, this command will work on an LP08 and LS8E line printer.

Example:

```
.SET LPT:NO LV8E
```

25.5.5 WIDTH n

Specifying this attribute sets the width of the line printer to the decimal number you specify as n.

Format:

```
.SET LPT WIDTH n
```

where:

n is a decimal number in the range of 1-256.

Example:

```
.SET LPT WIDTH 80
```

You may not use the NO modifier with this attribute.

25.6 ANY DEVICE ATTRIBUTES

25.6.1 FILES

When you specify this attribute, the handler handles a file-structured device.

Format:

```
.SET device [NO] FILES
```

Example:

```
.SET MTA0:FILES
```

If you want the handler to handle non-file structured devices, use the NO modifier in the command line.

Example:

```
.SET DTA1:NO FILES
```

NOTE

This attribute remains in effect until the next time you bootstrap, when the original status will be restored.

25.6.2 DVCode nn

Specifying this attribute sets the IOT device code the handler uses to the decimal number you specify as nn. This number should be from 30-77.

Format:

```
.SET device DVC nn
```

Example:

```
.SET RXA0 DVC 64
```

You could use this example if you hooked up your diskettes to the non-standard device code of 64. You may not use the NO modifier with this attribute.

25.6.3 LOCation n=m or LOCation n

Specifying the first argument changes the contents of the location in the handler you specify as n to contain the value you specify as m. Both n and m are octal numbers.

where:

n is an octal number from 0-177 for one-page handlers and from 0-377 for two-page handlers.

m is an octal number from 0-7777.

Format:

```
.SET device LOC n=m
```

Example:

```
.SET LPT LOC 37-1234
```

When you specify the second argument, the system prints the current contents of the location in the handler you specify as n. Follow this with a slash. Enter a new value in that location by typing that value followed by a carriage return. If you want to leave the contents of that location unchanged, type a carriage return only.

Format:

```
.SET device LOC n
```

Example:

```
.SET PTP LOC 144
```

25.6.4 READOnly

When you specify this attribute, the device specified becomes a read-only device. Therefore, any output sent to this device causes an error message informing you that the output device is a read-only device.

Format:

```
.SET device [NO] READO
```

Example:

```
.SET TTY READO
```

To remove the READONLY attribute, use the NO modifier in the command line.

Example:

```
.SET TTY NO READO
```

NOTE

The READONLY attribute remains in effect only until the next time you bootstrap, when its original status will be restored.

25.6.5 VERSION x

Specifying this attribute changes the version number of the handler to the letter you specify as x.

Format:

```
.SET device VERSION x
```

Example:

```
.SET TV:VERSION G
```

You may not use the NO modifier with this attribute.

25.6.6 BLOCK b, LOCation n=m or BLOCK b, LOC n

Specifying the first attribute changes the contents of the location in the handler you specify as n, located in the block you specify as b. The contents of that relative location changes to the value you specify as m.

where:

b is an octal number

n is an octal number from 0-177 for one-page handlers and from 0-377 for two-page handlers.

m is an octal number from 0-7777.

Format:

```
.SET device LOC n=m
```

Example:

```
.SET RKB1 LOC 10 = 2420
```

When you specify the second attribute, the system prints the current contents of the location in the handler you specify as n, located in the block you specify as b. Follow this with a slash. Enter a new value in that location by typing that value followed by a carriage return. If you want to leave the contents of that location unchanged, type a carriage return.

Format:

```
.SET device LOC n
```

Example:

```
.SET LPT LOC 175
```

CHAPTER 26

SRCCOM

SRCCOM compares two source files line by line and prints all their differences. Usually, the two files are different versions of a single program. In this case, SRCCOM prints all the editing changes that transpired between the two versions, making it a useful debugging tool.

26.1 SRCCOM ASSEMBLY INSTRUCTIONS

To make SRCCOM.BN from SRCCOM.PA, type:

```
.R PAL8
*dev:SRCCOM (,dev:SRCCOM.LS) <dev:SRCCOM
```

The listing file shown in parentheses is optional.

To make SRCCOM.SV from SRCCOM.BN, type:

```
.R ABSLDR
*dev:SRCCOM$
.SAdevSRCCOM
```

To load and save the binary papertape (DEC-S8-OSYSB-A-PB17), type

```
.R ABSLDR
*PTR:$^   (Type any character in response to ^)
.SAVE dev SRCCOM
```

26.2 LOADING SRCCOM

To use SRCCOM, type:

```
.R SRCCOM
*OUTPUT<INPUT1,INPUT2
```

INPUT1 and INPUT2 are both the source files you are comparing and the input devices. You must specify both files, and they must be non-empty. If you omit an input device, it is assumed to be DSK.

OUTPUT specifies the output file and device where the program will list the differences. If you specify an output file name, the default output device is DSK. If the output device is non-file structured, a file name is unnecessary. If output is to a file-structured device, you must specify an output file name. If no output specification exists, TTY is assumed.

Table 26-1 lists the run-time options accepted by SRCCOM.

Table 26-1
SRCCOM Run-Time Options

Option	Meaning
/C	Do not count differing comment fields as a difference.
/S	Do not compare tabs and spaces when considering lines different.
/T	Convert tabs to spaces on output.
/B	Count blank lines in the comparison. A blank line is considered as a carriage return only. In particular SRCCOM does not treat a space and carriage return combination under /S/B as a blank line.
/X	Like /C but does not print comment fields on the output file.

Examples:

```
.R SRCCOM
*DSK:DIFFIL<DTA1:ORIG,DTA2:COPY
```

Compare the source file ORIG on DTA1 and COPY on DTA2, and store the differences on DSK as DIFFIL.

```
.R SRCCOM
*DIFFIL<FIRST,SECOND
```

Compare the source files FIRST and SECOND on DSK, and output the differences to DIFFIL on DSK.

```
.R SRCCOM
*LPT:<DTA1:FILE1,PTR:
^
```

Compare source file FILE 1 on DTA1 and one from the high-speed paper tape reader, and output the differences to the line printer.

26.3 SRCCOM OUTPUT

The first line of output printed by SRCCOM is "SRCCOM Vx", where x is the current version number, then two header lines followed by as many difference groups as necessary. The header lines are printed as follows:

```
file 1)   header line of file 1
file 2)   header line of file 2
```

A difference group has the form:

```
1)  nnn line 1, file 1
1)      line 2, file 1
1)      line 3, file 1
  .
  .
1)      line n, file 1
  .
****
2)  nnn line 1, file 2
2)      line 2, file 2
  .
  .
  .
2)      line m, file 2
```

where nnn is the number of the page in the PAL listing. Lines 1 through n-1 of file 1 and 1 through m-1 of file 2 did not agree. SRCCOM compares areas of the two programs, and prints differences until it finds 3 lines that agree. The last lines printed (line n of file 1 and line m of file 2) are the first lines that agreed. You can change the number of consecutive lines to check for agreement to any number (k) with the option =k in the command line.

Example:

File 1	File 2	SRCCOM OUTPUT
		file 1) A
A	A	file 2) A
B	X	1) B
C	C	1) C
D	D	****
E	E	2) X
F	G	2) C
G	H	********
H	J	1) F
I		1) G
J		1) H
		1) I
		1) J
		****
		2) G
		2) H
		2) J

Occasionally, a decimal number appears following the close parenthesis after the file number. This decimal number indicates the source page in this file from which this line and all following lines (until the next such number) come.

If the two files are identical, SRCCOM prints the message:

NO DIFFERENCES

in the output file.

26.4 ERROR MESSAGES

SRCCOM error messages are of the form:

```
?n
```

where n is a single digit. The meaning of the various digits are:

?0	Insufficient core; this means that the differences between the files are too large to allow for effective comparison. Use of the /X option may alleviate this problem.
?1	Input error on file 1 or less than 2 input files specified.
?2	Input error on file 2.
?3	Output file too large for output device.
?4	Output error.
?5	Could not create output file

APPENDIX A

CHARACTER CODES

Table A-1
ASCII* Character Set

Character	8-Bit Octal	6-Bit Octal	Decimal Equivalent (A1 Format)	Character	8-Bit Octal	6-Bit Octal	Decimal Equivalent (A1 Format)
A	301	01	96	!	241	41	-1952
B	302	02	160	"	242	42	-1888
C	303	03	224	#	243	43	-1824
D	304	04	288	$	244	44	-1760
E	305	05	352	%	245	45	-1696
F	306	06	416	&	246	46	-1632
G	307	07	480	'	247	47	-1568
H	310	10	544	(	250	50	-1504
I	311	11	608	)	251	51	-1440
J	312	12	672	*	252	52	-1376
K	313	13	736	+	253	53	-1312
L	314	14	800	'	254	54	-1248
M	315	15	864	-	255	55	-1184
N	316	16	928	.	256	56	-1120
O	317	17	992	/	257	57	-1056
P	320	20	1056	:	272	72	-352
Q	321	21	1120	;	273	73	-288
R	322	22	1184	<	274	74	-224
S	323	23	1248	=	275	75	-160
T	324	24	1312	>	276	76	-96
U	325	25	1376	?	277	77	-32
V	326	26	1440	@	300		32
W	327	27	1504	[	333	33	1760
X	330	30	1568	\	334	34	1824
Y	331	31	1632	]	335	35	1888
Z	332	32	1696	^(^)**	336	36	1952
0	260	60	-992	<-(-)**	337	37	2016
1	261	61	-928	Leader/Trailer	200		
2	262	62	-864	LINE FEED	212		
3	263	63	-800	Carriage RETURN	215		
4	264	64	-736	SPACE	240	40	-2016
5	265	65	-672	RUBOUT	377		
6	266	66	-608	Blank	000		
7	267	67	-544	BELL	207		
8	270	70	-480	TAB	211		
9	271	71	-416	FORM	214		

* An abbreviation for American Standard Code for Information Interchange.

** The character in parentheses is printed on some Teletypes.

APPENDIX B

LOADING PROCEDURES

B.1 INITIALIZING THE SYSTEM

Before using the computer system, it is good practice to initialize all units. To initialize the system, make sure that all switches and controls are as specified below.

1. Main power cord is properly plugged in.
2. Terminal is turned OFF.
3. Low-speed punch is OFF.
4. Low-speed reader is set to FREE.
5. Computer POWER key is ON.
6. PANEL LOCK is unlocked.
7. Console switches are set to 0.
8. SING STEP is not set.
9. High-speed punch is OFF.
10. DECtape REMOTE lamps are OFF.

The system is now initialized and ready for your use.

B.2 LOADERS

READ-IN MODE (RIM) LOADER

When you receive a computer in the PDP-8 series, it is nothing more than a piece of hardware; its core memory is completely demagnetized. The computer "knows" absolutely nothing, not even how to receive input. However, you can manually load data directly into core using the console switches.

LOADING PROCEDURES

The RIM Loader is the first program you load into the computer, and you load it using the console switches. The RIM Loader instructs the computer to receive and store, in core, data punched on paper tape in RIM-coded format. You use the RIM Loader to load the BIN Loader described below.

There are two RIM loader programs: you use one when you want to input from the low-speed paper tape reader, and the other when you want to input from the high-speed paper tape reader. The locations and corresponding instructions for the low-speed reader are listed in Table B-1. Information for the high-speed reader is listed in Table B-2.

For each step in the table, place each of the PDP-8/E console SWITCH REGISTER switches numbered 0 to 11 either in the up position if the corresponding table entry is 1, or in the down position if the corresponding table entry is 0. When all 12 switches have been set to correspond to a line in the table, follow the instructions in the right-hand column and proceed to the next line. The tables also include octal values of the binary switch settings for the benefit of users familiar with octal numbers.

Table B-1
RIM Loader for Low-Speed Reader

Step #	Octal Values	Switch Register Setting				And Then
		012	345	678	91011	
1	0000	000	000	000	000	press EXTD ADDR LOAD
2	7756	111	111	101	110	press ADDR LOAD
3	6032	110	000	011	010	lift DEP key
4	6031	110	000	011	001	lift DEP key
5	5357	101	011	101	111	lift DEP key
6	6036	110	000	011	110	lift DEP key
7	7106	111	001	000	110	lift DEP key
8	7006	111	000	000	110	lift DEP key
9	7510	111	101	001	000	lift DEP key
10	5357	101	011	101	111	lift DEP key
11	7006	111	000	000	110	lift DEP key
12	6031	110	000	011	001	lift DEP key
13	5367	101	011	110	111	lift DEP key
14	6034	110	000	011	100	lift DEP key
15	7420	111	100	010	000	lift DEP key
16	3776	011	111	111	110	lift DEP key
17	3376	011	011	111	110	lift DEP key
18	5356	101	011	101	110	lift DEP key

Table B-2
RIM Loader for High-Speed Reader

Step #	Octal Values	Switch Register Setting				And Then
		012	345	678	91011	
1	0000	000	000	000	000	press EXTD ADDR LOAD
2	7756	111	111	101	110	press ADDR LOAD
3	6014	110	000	001	100	lift DEP key
4	6011	110	000	001	001	lift DEP key
5	5357	101	011	101	111	lift DEP key
6	6016	110	000	001	110	lift DEP key
7	7106	111	001	000	110	lift DEP key
8	7006	111	000	000	110	lift DEP key
9	7510	111	101	001	000	lift DEP key
10	5374	101	011	111	100	lift DEP key
11	7006	111	000	000	110	lift DEP key
12	6011	110	000	001	001	lift DEP key
13	5367	101	011	110	111	lift DEP key
14	6016	110	000	001	110	lift DEP key
15	7420	111	100	010	000	lift DEP key
16	3776	011	111	111	110	lift DEP key
17	3376	011	011	111	110	lift DEP key
18	5357	101	011	101	111	lift DEP key

After you have loaded RIM, it is good programming practice to verify that all instructions were stored properly. You can do this by performing the steps illustrated in Figure B-2, which also shows how to correct an incorrectly stored instruction.

When loaded, the RIM Loader occupies absolute locations 7756 through 7776.

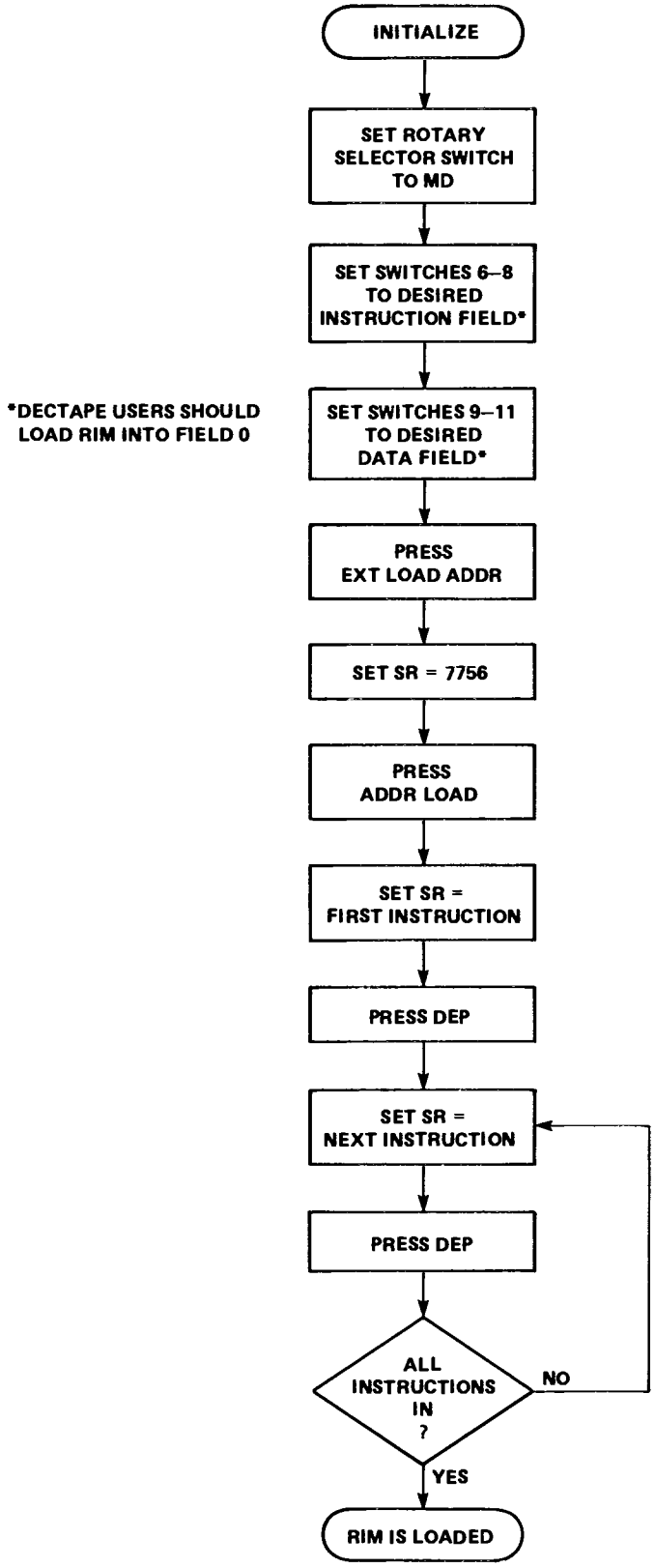


Figure B-1 Loading the RIM Loader

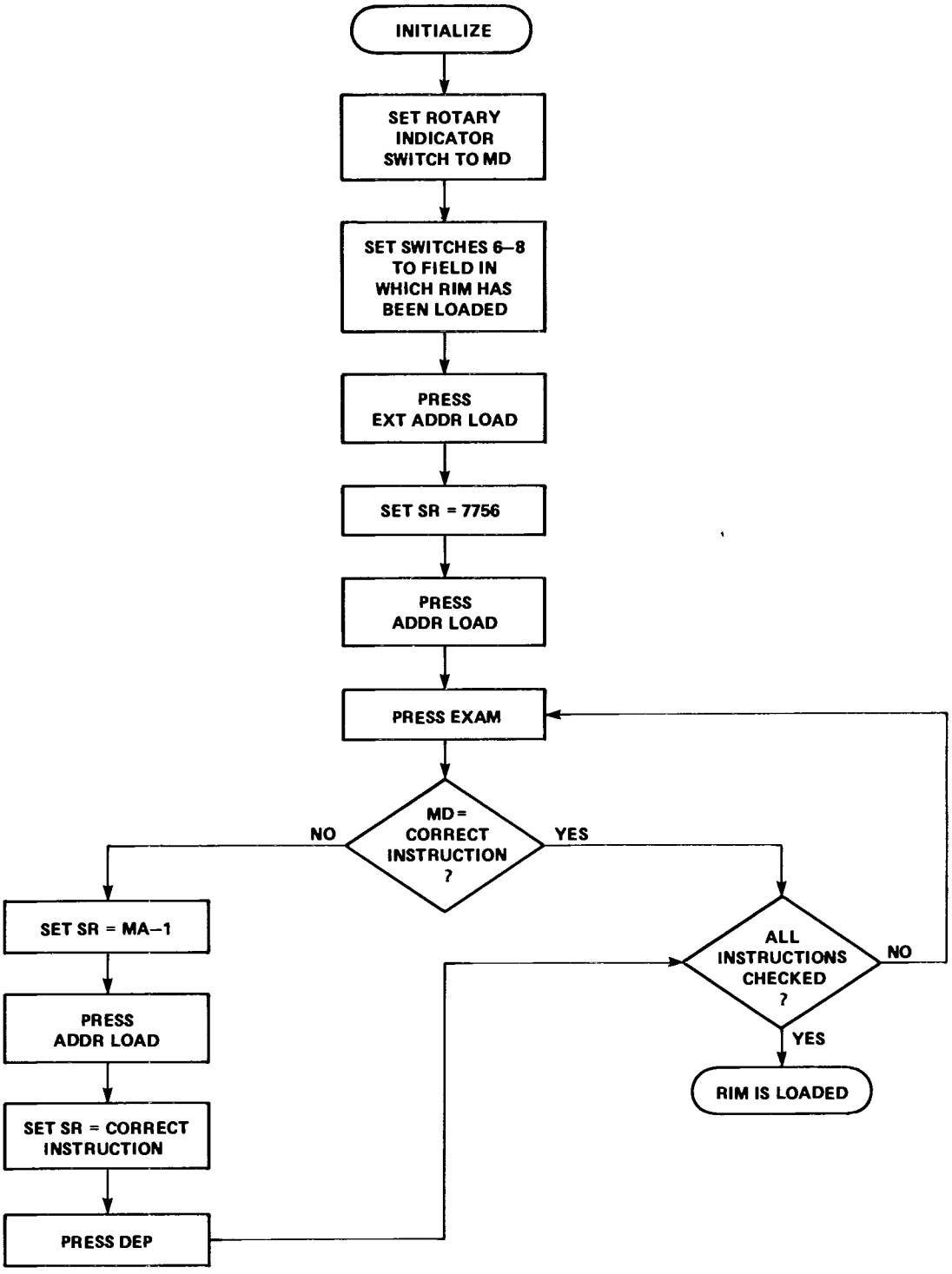


Figure B-2 Checking the RIM Loader

B.2.1 Binary (BIN) Loader

The BIN Loader is a short utility program that, when in core, instructs the computer to read binary-coded data punched on paper tape and store it in core memory. Use BIN primarily to load the programs furnished in the software package (excluding the loaders and certain subroutines) and your binary tapes.

BIN is furnished to you on punched paper tape in RIM-coded format. Therefore, RIM must be in core before BIN can be loaded. Figure B-3 illustrates the steps necessary to properly load BIN. Note that when you load BIN, you should use the same input device (low- or high-speed reader) as when you loaded RIM.

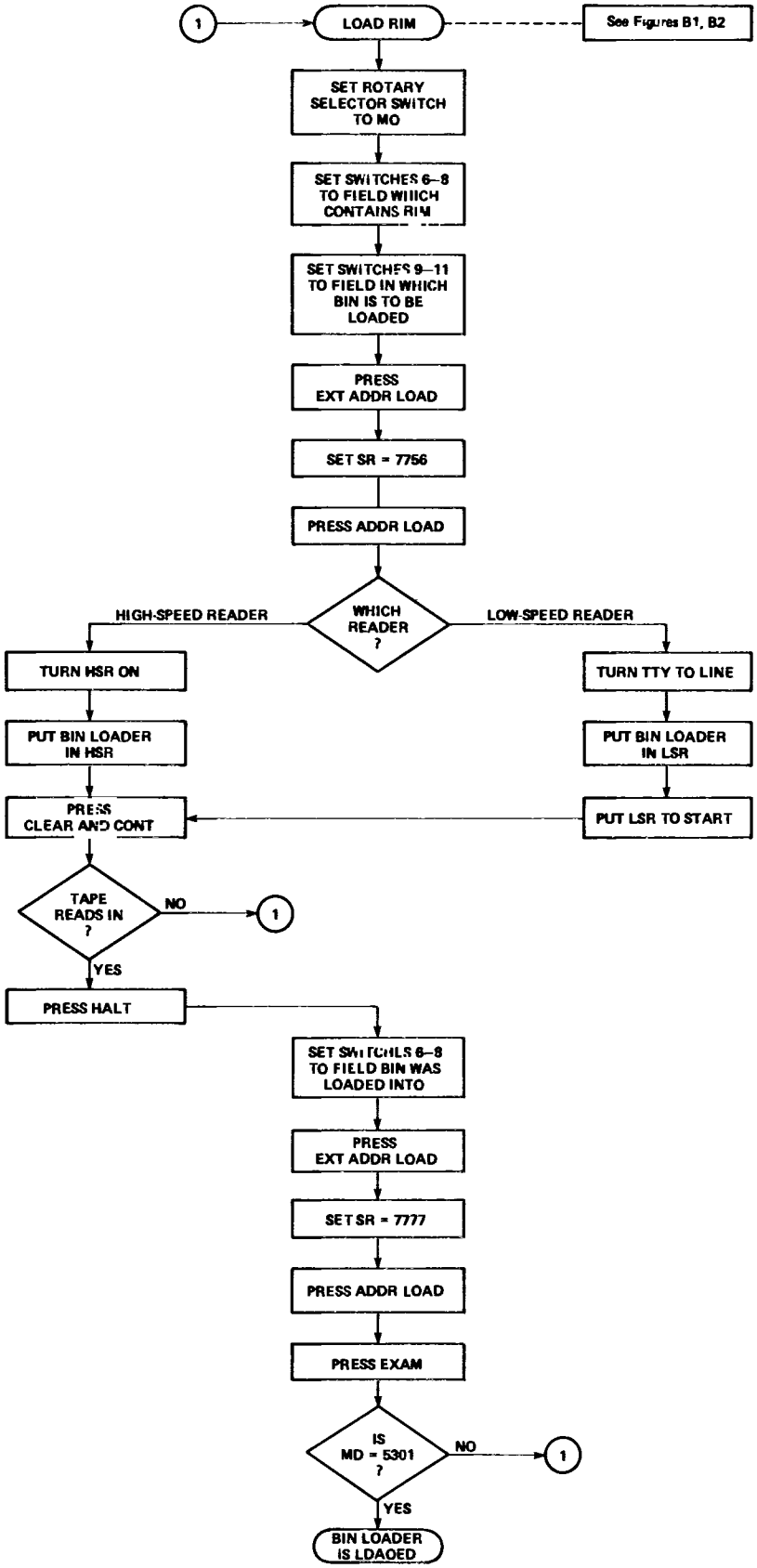


Figure B-3 Loading the BIN Loader

When stored in core, BIN resides on the last page of core, occupying absolute locations 7625 through 7752 and 7777.

BIN was purposely placed on the last page of core so that it would always be available for use -- the programs in DEC's software package do not use the last page of core (excluding the Disk Monitor). You must be aware that if you write a program that uses the last page of core, BIN will be wiped out when that program runs on the computer. When this happens, you must load RIM and then BIN before you can load another binary tape.

When loading binary tapes, start on the leader-trailer code (Code 200), otherwise zeros may be loaded into core, destroying previous instructions.

Figure B-4 illustrates the procedure for loading binary tapes into core.

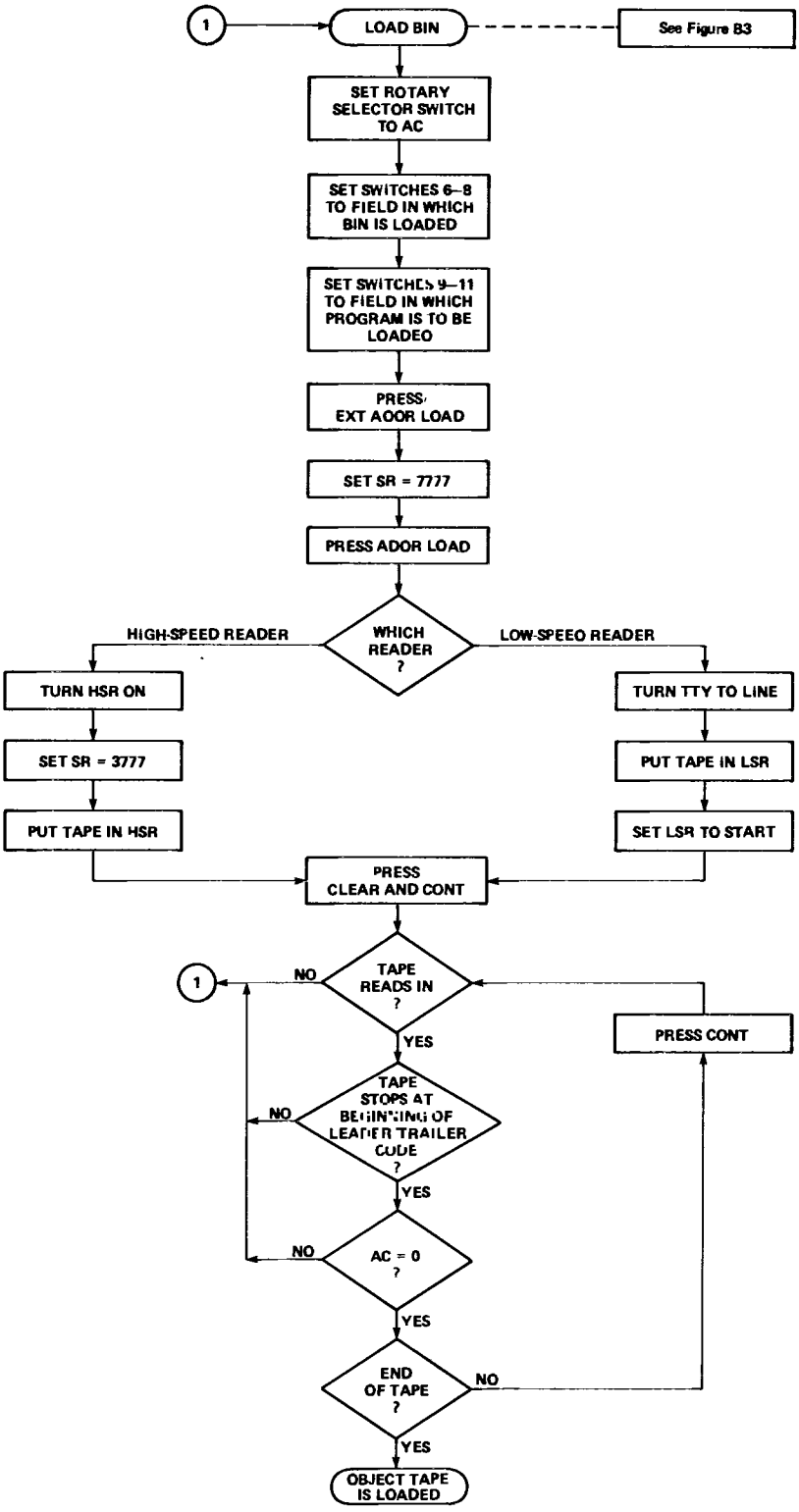


Figure B-4 Loading a Binary Tape Using BIN

APPENDIX C

OS/8 DEMONSTRATION RUN

The following pages present a demonstration of the use of the OS/8 system. The terminal output is set off by letters (to the left) that correspond to the textual explanations on the facing page. This demonstration illustrates the procedures involved and the use of many of the OS/8 system programs and commands.

A Use the CCL command to zero the DECtape on Unit 1, specifying one additional information word in the directory.

B You then type the DATE command to set the system date to April 10, 1974.

C Use the ASSIGN command to give DTA1 the additional name IN. All subsequent references to IN refer to DTA1.

D DIRECT is called to list the directory of DECtape Unit 1. A directory listing of DTA1 is produced.

E Use the Keyboard Monitor GET and SAVE commands to copy EDIT from the system device to DTA1.

F Run the FORTRAN compiler via the CCL command COMPILE to compile and execute the program TEST1 on the device DSK:. An output relocatable binary file named TEST1 is saved by SABR on DECtape Unit 1. The program has an error in it. Control is returned to the Keyboard Monitor after execution and the error message printed on the terminal.

G Use the program EDIT, located on DTA1, to correct the error in TEST1. Input the old program, TEST1, to the Editor, and the new (corrected) program, TEST2, is written by the Editor onto DTA1. The first page is yanked into core.

H You have noticed a misspelled word in FORMAT line 35 and used the string search feature of the Editor to correct it. An END statement is appended to the program.

A
```
.ZERO DTA1:=1
```

B
```
.DA 4/10/74
```

C
```
.AS DTA1 IN
```

D
```
.DIR IN:

 10-APR-74




 730 FREE BLOCKS
```

E
```
.GET SYS EDIT

.SAVE IN EDIT 0-5000;200=2001
```

F
```
.COMPILE IN:TEST2<IN:TEST2
/       CALL EXIT
                ^
NO END STATEMENT
```

G
```
.RUN IN EDIT
*IN:TEST2.FT<TEST1.FT

#Y
```

H
```
#/:0055
#$35    'L
35      FORMAT ('THE AVERATE IS' F20.2/)

#.S
35      FORMAT ('THE AVERAT/GE IS' F20.2/)


#.L
35      FORMAT ('THE AVERAGE IS' F20.2/)

#/L
        CALL EXIT
#A
        END
```

I You instruct the Editor to list the entire FORTRAN program.

J Note the use of implied DO loops in the READ and WRITE statements ...

K and device independent I/O. A file named ABCD.DA is opened on the default device DSK and data is written into it. When all the data is entered, the file is closed. Later, this file is again opened, and the data is read and used by the program.

L An S in column 1 of a FORTRAN line indicates that the line contains SABR code.

M Use CALL EXIT to return control to the Keyboard Monitor after execution.

N After listing the program, the E command to the Editor closes the file and returns control to the Keyboard Monitor.

```
#L
C     THIS PROGRAM  PRESENTS A FEW OF THE FEATURES
C     OF OS/8 FORTRAN; SPECIFICALLY IT INCLUDES IM-
C     PLIED DO LOOPS, DIRECT INSERTION OF SABR CODE
C     AND EXPANDED I/O.

C     THIS SECTION READS DATA FROM THE TTY AND WRITES
C     IT ONTO THE DSK AS AN ARRAY.

      DIMENSION A(10)
 5    CALL OOPEN ('DSK','ABCD')
      WRITE (1,10)
10    FORMAT ('ENTER 10 NUMBERS IN F6.2 FORMAT.')
      WRITE (1,11)
11    FORMAT ('FOLLOW EACH WITH A CARRIAGE RETURN:'//)
      READ (1,15) (A(N), N=1,10)
      WRITE (4,15) (A(N), N=1,10)
15    FORMAT (F6.2)
      CALL OCLOSE

C     THIS SECTION ADDS THE NUMBERS STORED ON THE DSK
C     AND AVERAGES THEM, PRINTING BOTH RESULTS ON
C     THE TELETYPE.

      SUM=0.0
      DO 20 I=1,10
20    A(I)=0.0
      CALL IOPEN ('DSK','ABCD')
      READ (4,15) (A(N), N=1,10)
      DO 25 N=1,10
      SUM=SUM+A(N)
25    CONTINUE
      WRITE (1,30) SUM
30    FORMAT (/'THE SUM IS' F20.2)
      AVR=SUM/10.
      WRITE (1,35)AVR
35    FORMAT ('THE AVERAGE IS' F20.2/)

C     THE SABR CODE FOLLOWING CHECKS FOR A CARRIAGE
C     RETURN CHARACTER TO INITIATE REPEATING THE
C     PROGRAM.  ANY OTHER CHARACTER TERMINATES THE
C     PROGRAM.

      WRITE (1,40)
40    FORMAT ('TO REPEAT, TYPE A CARRIAGE RETURN.'//)
SX,   KSF
S     JMP X
S     KRB
S     TAD MYES
S     SZA
S     JMP \50
      GO TO 5
SMYES, -215
50    WRITE (1,60)
60    FORMAT (/'PROGRAM DONE'//)
      CALL EXIT
      END
#E
```

O Use the ASSIGN command to change the assigned name of DTA1 from IN to OUT. The FORTRAN compiler is called again, and the program is loaded. An output relocatable binary file named TEST2 is saved by SABR on DECtape Unit 1.

P The FORTRAN program is executed via the CCL command EXECUTE. The /G, /I, and /O options cause automatic loading and execution of the program and the device independent I/O, and results are calculated and returned. Execution is not repeated.

Q Use the DEASSIGN command to delete all device names you have assigned. The ASSIGN command then gives the name X to DTA1.

R The CCL command DIR obtains a directory listing of DECtape Unit 1. TEST2.RL is the relocatable binary output file from the FORTRAN compilation.

S Next, use the CCL command DIR to print the directory of the system device on the line printer. ABCD.DA is the FORTRAN data file created in the preceding program.

T The CCL command DEL deletes the unwanted files PROG3 and PROG4 from the system device. Then the ASCII file TEST2 is copied from DECtape Unit 1 to the system device with the CCL command COPY.

O

```
.AS DTA1 OUT
.COMPILE OUT:TEST2<OUT:TEST2
```

P

```
.EXECUTE OUT:TEST2/G/I/O

ENTER 10 NUMBERS IN F6.2 FORMAT.
FOLLOW EACH WITH A CARRIAGE RETURN:


16.23
32.00
171.45
2.15
22.10
77.35
2.91
66.00
.46
27.50
THE SUM IS              418.15
THE AVERAGE IS                41.81

TO REPEAT, TYPE A CARRIAGE RETURN.



PROGRAM DONE
```

Q

```
.DEA

.AS DTA1 X
```

R

```
.DIR X:

 10-APR-74

EDIT  .SV  12 10-APR-74
TEST2 .FT   4 10-APR-74
TEST2 .RL   4 10-APR-74


710 FREE BLOCKS
```

S

```
.DIR LPT:<SYS:

 10-APR-74

ABSLDR.SV    5 15-JAN-74
CCL    .SV  17 26-FEB-74
DIRECT.SV    7 18-JAN-74
FOTP   .SV   8 18-JAN-74
PIP    .SV  11 18-JAN-74
LIB8   .RL  29 18-JAN-74
EDIT   .SV  10 18-JAN-74
PAL8   .SV  16 18-JAN-74
CREF   .SV  13 18-JAN-74
BITMAP.SV    5 18-JAN-74
FORT   .SV  25 18-JAN-74
SABR   .SV  24 18-JAN-74
LOADER.SV  12 18-JAN-74
SRCCOM.SV    5 18-JAN-74
BOOT   .SV   5 18-JAN-74
BUILD  .SV  33 18-JAN-74
EPIC   .SV  14 18-JAN-74
PIP10  .SV  17 18-JAN-74
RESORC.SV  10 18-JAN-74
DTFRMT.SV    7 18-JAN-74
TDFRMT.SV    9 18-JAN-74
RK8FMT.SV    9 18-JAN-74
RKEFMT.SV    6 18-JAN-74
CAMP   .SV   8 18-JAN-74
MCPIP  .SV  13 18-JAN-74
DTCOPY.SV    5 18-JAN-74
TDCOPY.SV    7 18-JAN-74
LIBSET.SV    5 18-JAN-74
CCL    .PA 130 26-FEB-74
TEST1  .BK   4 11-APR-74
TEST1  .RL   4
TEST1  .FT   4 10-APR-74
TEST2  .RL   4 10-APR-74
ABCD   .DA   1 10-APR-74
PROG3  .     1 10-APR-74
PROG4  .     1 10-APR-74

2295 FREE BLOCKS
```

T

```
.DEL PROG3,PROG4
FILES DELETED:
PROG3.
PROG4.

.COPY SYS:TEST2.FT<X:TEST2.FT
```

APPENDIX D

OS/8 FILE NAME EXTENSIONS

This appendix lists the file name extensions used in OS/8.

Extension	Meaning
.BA	BASIC source file (default extension for a BASIC input file)
.BI	Batch input file
.BK	Backup ASCII file (default extension for a TECO output file)
.BN	Absolute binary file (default extension for ABSLDR, BUILD, and BITMAP input files; also used as default extension for PAL8 binary output file)
.DA	Data file
.DC	Documentation file
.DI	Directory listing
.FT	FORTRAN language source file (default extension for FORT input files)
.HL	Help file (default extension for HELP input files)
.LD	F4 load mode (default assumed by run-time system, F4 loader)
.LS	Assembly listing output file (default extension for PAL8 and SABR)
.MA	Macro source file
.MP	File containing a loading map (used by the Linking Loader)
.PA	PAL8 source file
.RA	RALF assembly language file

Extension	Meaning
.RB	Relocatable binary source file
.RL	Relocatable binary file (default extension for a Linking Loader input file; also used as the default extension for an 8K SABR output file)
.SB	8K SABR source file
.SV	Core image file or SAVE file; appended to a file name by the R, RUN, SAVE, and GET Keyboard Monitor commands
.SY	System head
.TE	TECO macro file (default extension for a MUNG input file)
.TM	Temporary file generated by FORTRAN or SABR for system use (default extension for CREF input files and PAL8 output files)
.TX	Text files

APPENDIX E

OS/8 DEVICE HANDLERS

Most of the the device handlers supplied with the OS/8 system have simple operating characteristics that require no action from you. Some device handlers perform additional operations when you are performing I/O on a given device. This appendix gives a brief description of the OS/8 device handlers. See the OS/8 Software Support Manual (DEC-S8-OSSMB-A-D) for more detailed information concerning device handlers.

E.1 HIGH-SPEED READER/PUNCH

The device handler for the high-speed paper tape reader, before reading a tape, prints an uparrow (^) and waits for the user to type any single character at the keyboard. This gives you time to check the reader to ensure that the tape is loaded correctly, and it facilitates reading multiple tapes (e.g., a PAL8 source tape must be loaded three times for the three passes of the assembler). Characters are read from the paper tape and packed into an input buffer. The end of the paper tape or a full input buffer causes the buffer to be free for your program. Typing CTRL/C while the tape is moving causes a return to the Keyboard Monitor.

The handler for the high-speed paper tape punch unpacks characters from the output buffer and punches then on paper tape. Typing CTRL/C causes a return to the Keyboard Monitor. You must manually turn on the punch before trying to output to that device.

E.2 LOW-SPEED READER/PUNCH

In addition to the handler for the high-speed reader/punch, a similar handler is available for the ASR-33 Teletype low-speed reader/punch. If you do not have high-speed I/O, you can still read and punch binary format tapes by using this handler. (The standard TTY handler cannot be used for binary format tapes, because the binary format can appear as control characters to the handler.) The operation of this handler is exactly the same as that for the high-speed reader/punch except that the uparrow is not printed.

E.3 TTY HANDLERS

There are two TTY (console terminal) handlers available: a one-page handler and a two-page handler. Both handlers perform I/O transfers between the terminal keyboard and an input buffer, or between an output buffer and the terminal.

The one-page handler echoes all terminal input and performs a line feed operation after any typed carriage return. A CTRL/O typed while output is being printed terminates printing of the current output buffer. A CTRL/C typed at any time during input or output causes a return to the Keyboard Monitor. Typing CTRL/Z as input terminates input and gives an end-of-file indication to the calling program. You should not use the TTY handler to read binary tapes from the low-speed reader.

You may use the two-page TTY handler only to read or write ASCII files; results are unpredictable with non-ASCII files. In addition to the features included in the one-page handler, this handler has the ability to delete the previous character, through the RUBOUT key, and to echo it either as a backslash (\) or as the character rubbed out. Other features have the ability to delete the current line, through CTRL/U, and to output the correct number of spaces to bring the text to the start of the next tab stop (through the TAB key).

The two-page TTY handler also includes approximately 30 free locations so that you may conditionalize certain nonstandard features. See the OS/8 Software Support Manual for a complete list of these features.

E.4 LINE PRINTERS

The OS/8 line printer handler is a one-page handler for the LP08, LS8E, and LV8 line printers. The handler performs a form feed operation before beginning an output task. The characters are unpacked from the output buffer and printed. A form feed is also produced following the completion of an output task. Typing CTRL/C while the line printer is in operation causes a return to the Keyboard Monitor. A CTRL/Z found in the output buffer causes printing to terminate and a form feed to be produced. Tabs and line overflow are handled; nulls are ignored.

Relative location 0 of this handler specifies the width of the line printer. You may patch this location using the ALTER command in BUILD. Set the location to the one's complement of the width desired. Initially, set this location to 7573 (octal), which corresponds to a 132-column printer. For example, to indicate an 80-column printer, set location 0 to 7657 (octal).

E.5 VR12 SCOPE

The VR12 scope handler for OS/8 (running on a PDP-12) displays characters on the VR12 scope on both channels. When the scope is full, the handler stops reading characters from the buffer and displays what is known as a scope page. The screen is considered full whenever the end of the buffer is reached, a CTRL/Z is encountered in text, or when the number of lines displayed become equal to the maximum number you specify. You can advance to the next scope page by typing any character other than CTRL/C.

When you type CTRL/C, control returns to the Keyboard Monitor. Control does not return to the calling program until a character is typed at a point when the handler is displaying the last scope page of a particular buffer load.

To use the VR12 handler, set the number of lines desired in a single scope page via the switch register (right switches). Set the switch register to the negative of the number of lines to be displayed in a scope page. When text reaches the right margin of the scope face, it is continued on the next physical line of the scope.

A line feed or form feed character causes succeeding text to continue on the next physical line. Carriage return characters have no effect on the display.

E.6 CARD READER

The device handler for the card reader reads cards in alphanumeric format from either a punched card reader or an optical mark card reader. Card format can have up to 80 characters per card; trailing blanks are deleted from each card. Blank cards cause a carriage return/line feed to be entered into the data stream. Typing CTRL/C while cards are being read terminates reading and returns control to the Keyboard Monitor. Typing CTRL/Z terminates further reading and performs as though an end-of-file card was read. (An end-of-file card contains an arrow character in column 1 (0-8-5 punch) with the remaining columns blank. Either CTRL/Z or the end-of-file card is necessary to terminate reading.) It is not possible to RUN or GET a program from the card reader because these commands assume a directory device.

E.7 DECTAPES

You may interrupt any DECtape other than the system device (if the system is a DECtape system) with a CTRL/C, returning control to the Keyboard Monitor. You must never WRITE LOCK DECtape unit 0 on a DECtape system while it is operating OS/8.

E.8 MAGNETIC TAPE

The handler for magnetic tape reads and writes either 7- or 9-channel magnetic tape with odd parity at 800 bpi. This handler is non-file structured, but you may alter it to read and write files. CTRL/C returns control to the Keyboard Monitor, but its use is not recommended since it leaves the tape without an end-of-file indicator.

E.9 CASSETTES

The cassette handler performs character I/O transfer between the cassettes and the buffer. It treats cassettes as non-file structured devices. Data appears on cassette in 192-byte records. Typing CTRL/C returns control to the Keyboard Monitor.

E.10 BATCH HANDLER

The OS/8 batch handler is used from a BATCH job to read from the BATCH stream. This is a one-page handler for read-only, non-file structured devices. If you use this handler when BATCH is not running, it generates a fatal error. The BATCH handler reads characters from the BATCH stream, ignoring line feeds, and creating a line feed after a carriage return. When the handler encounters a line beginning with a dollar sign, it pads the buffer with CTRL/Z and nulls, and takes the end-of-file return.

E.11 DSK AND SYS

The DSK and SYS device handlers work automatically without any user intervention.

APPENDIX F

OBTAINING OS/8 PROGRAM VERSION NUMBERS

When you receive new OS/8 software or when you wish to report problems with the software, you must know the version number of the OS/8 program in question. Most OS/8 system programs have version numbers that you can obtain by typing a command to the OS/8 Command Decoder * or to the called program. Some system programs print the version number at the beginning of the output listing. The following table shows how to obtain version numbers for most OS/8 system programs.

Program	How to Obtain Version Number
ABSLDR	Internal only.
BASIC	Printed in program heading.
BATCH	Type /V in BATCH command string.
BITMAP	Printed at top of output listing.
BOOT	Type VE to the / printed by BOOT.
BUILD	Type VE to the $ printed by BUILD.
CAMP	Type VE to the # printed by CAMP.
CCL	Type VER to the Keyboard Monitor.
Command Decoder	Internal only.
CREF	Printed at end of CREF output listing.
DIRECT	Type /W to the * printed by DIRECT.
EDIT	Type # to the # printed by EDIT.
EPIC	Internal only.
F4 Compiler	Printed in heading of output listing.
F4 Loader (LOAD)	Printed in heading of loading map.
FLAP	Printed in heading of output listing.
POTP	Type /W to the * printed by FOTP.
FRTS	Type /V to the * printed by FRTS (to be implemented later).
Keyboard Monitor	Type VER to the Keyboard Monitor.
MCPIP	Type /V to the * printed by MCPIP.
ODT	Internal only.
PAL8	Printed in heading of output listing.
PIP	Type /V to the * printed by PIP.
PIP10	Printed in heading of directory listing.
RALF	Printed at heading of output listing.
RESORC	Type /V to the * printed by RESORC.
SRCCOM	Printed in heading of output listing.
TECO	Type CTRL/V to the * printed by TECO.

INDEX

Aborting a program,
  CTRL/C, 3-10
Angle bracket (<), usage,
  command decoder, 1-45, 5-1
Assembly instructions,
  BITMAP, 7-3
  EPIC, 15-11
  SRCCOM, 26-1
ASSIGN command, keyboard
    monitor, 3-11
Asterisk (*) usage,
  command decoder, 5-1
  wild card, 3-7
At sign (@) construction, CCL,
    1-56, 3-8

BACKSPACE command, CCL, 3-12
BASIC command, 3-13
BATCH, 6-1
  demonstration program, 6-13
  error messages, 6-9
  input file, 6-2
  loading and saving, 6-18
  monitor commands, 6-4 to 6-6
  output file, 6-2
  restrictions, 6-12
  running from punched cards,
    6-11
  run-time options, 6-3
  transferring software from
    cassette, 6-19
BITMAP utility program, 7-1
  assembly instructions, 7-3
  error messages, 7-3
  hardware/software requirements,
    7-1
  loading, 7-1
  options, 7-2
  output, 7-2
BOOT command, CCL, 3-14
BOOT (bootstrap utility program),
    8-1
  mnemonics, 8-2
Breakpoints, 19-4
BUILD, (system generation
    program), 9-1
  cassette device handlers, 9-3
  commands, 9-7 to 9-19
  DECtape device handlers, 9-3
  device handler format, 9-21
  device handlers, 9-1
  editing characters, 9-7
  error messages, 9-20
  paper tape device handlers,
    9-3

CAMP (Cassette and Magnetic
    Tape Positioner program),
    10-1
  commands, 10-1 to 10-4
  error messages, 10-5
Cassette and Magnetic Tape
    Positioner -- see CAMP
Cassette transfer program, 18-1
CCL (Concise Command Language),
    3-3
  command, 3-15
Character deletion, 3-10
Character search, Symbolic
    Editor, 4-12
Character string search,
    Symbolic Editor, 4-13 to
    4-16
Command Decoder, 5-1
  called from BATCH, 6-6
  error messages, 5-3
  file specifications, 5-1
  input string, 5-1
Command mode, Editor, 4-4
Commands,
  CCL, 3-4
  keyboard, 3-1, 3-11 to 3-67
  monitor, 3-1
Command string examples,
    Command Decoder, 5-1
Command string format, BATCH,
    6-2
Command summary, ODT, 19-8
COMPARE command,
  CCL, 3-16
  EPIC, 15-8
COMPILE command, CCL, 3-19
COPY command, CCL, 3-22
CREATE command, CCL, 3-23
CREF command, CCL, 3-25
Cross-Reference Program (CREF),
    11-1
  error messages, 11-6
  options, 11-1
  output, 11-3
  pseudo-op handling, 11-3
  restrictions, 11-5
CTRL/C, 3-10
CTRL/U, 3-10
Current location, ODT, 19-7

DATE command, 3-26
DEASSIGN command, 3-27
DECtape copy and format
    programs, 3-1

DECtape file for BATCH input, 6-1
DECtape systems, BUILD, 9-1
DELETE command, CCL, 3-27
Deletion of characters, 3-10
Demonstration program,
  BATCH, 6-13
Descriptor block, BUILD, 9-22
Device codes for paper tape,
  EPIC, 5-4
Device Control Block (DCB)
  word, BUILD, 9-23
Device entry points, 9-24
Device handlers, OS/8, 9-1
Device handlers, RESORC, 22-1
Device names,
  assignment of, 3-11
  deassignment of, 3-29
  permanent, 2-1
Device types, RESORC, 22-1
DIRECT command, 3-30
DIRECT utility program, 12-1
  error messages, 12-5
  examples, 12-3
  options, 12-2
  wild card construction, 3-7
Disk file for BATCH input, 6-1
Dollar sign ($),
  BATCH usage, 6-4
DOT (.) character,
  monitor response, 3-1
DTCOPY, 13-10
DTFRMT, 13-1
DUMP utility program, 14-1
Duplicate command, 3-32

EDIT command, CCL, 3-33
EDIT editing program, 4-1
Editing characters, BUILD, 9-7
Editing commands, EPIC, 15-4
Edit, Punch and Compare (EPIC)
  utility program -- see EPIC
Entry point offset, BUILD, 9-24
EOF command, CCL, 3-34
EPIC (Edit, Punch and Compare)
  utility program, 15-1
  assembly instructions, 15-11
  command format, 15-2
  compare commands, 15-8
  editing commands, 15-5
  error conditions, 15-4
  error messages, 15-8
  loading, 15-1
  loading from paper tape, 15-11
  low-speed I/O, 15-4
  paper tape format, 15-10
Equal sign (=),
  octal number options, 3-5
Error conditions,
  EPIC, 15-4
  ODT, 19-7
Error messages,
  BATCH, 6-9
  BITMAP, 7-3
  BUILD, 9-20
  CAMP, 10-5
  Command Decoder, 5-3
  CREF, 11-6
  DIRECT, 12-5
  Editor, 4-18
  EPIC, 15-8
  FOTP, 16-11
  MCPIP, 18-4
  Monitor, 3-1
  PIP, 20-8
  PIP10, 21-3
  RESORC, 22-6
  SRCCOM, 24-6
EXECUTE command, CCL, 3-35
Extension for BATCH input file,
  6-2
Extensions, CCL compiler-
  assembler, 3-19
Extensions to file names, key-
  board monitor, 2-2

File names, 2-2
File Oriented Trasnfer Program
  (FOTP), 16-1
  error messages, 16-11
  input specifications, 16-1
  options, 16-7
  output specifications, 16-3
File specifications,
  Command Decoder, 5-1
File transfers, DECsystem-10,
  21-1
FOTP -- see File Oriented
  Transfer Program
FUTIL utility program, 17-1

GET command, 3-35

Handlers -- see Device handlers
Header block, BUILD, 9-22
HELP command, CCL, 3-37
Hyphen construction in BUILD,
  9-8

Indirect commands, CCL, 3-8
Indirect references, ODT, 19-1
Input files,
  BATCH, 6-2
  Editor, 4-1
Input/output,
  low speed, with EPIC, 15-4
Input/output specifications,
  CCL, 3-1 to 3-10
  Command Decoder, 5-1
  DIRECT, 12-1
  FOTP, 16-1 to 16-3
  RESORC, 22-1
Input string, Command Decoder, 5-1
Inter-buffer character string search, Editor, 4-16
Intra-buffer character string search, Editor, 4-13

Keyboard commands, 3-11 to 3-67

LOAD command, CCL, 3-39
Loading,
  BATCH, 6-18
  BITMAP, 7-1
  EPIC, 15-1
  EPIC from paper tape, 15-11
  SRCCOM, 26-1
Low-speed I/O with EPIC, 15-4

Magnetic tape file names, 18-4
Magtape/Cassette Peripheral Interchange Program (MCPIP), 18-1
  error messages, 18-4
  options, 18-2
MAP command, CCL, 3-41
MCPIP -- see Magtape/Cassette Peripheral Interchange Program
Memory command, 3-41
Mnemonics for devices, BOOT, 8-2
MUNG command, CCL, 3-44

Octal Debugging Technique (ODT), 19-1
  commands, 19-2
  errors, 19-7
  special characters, 19-2
  techniques, 19-4
ODT -- see Octal Debugging Technique
ODT command, keyboard monitor, 3-45
Options, 3-4 to 3-6
Output,
  BITMAP, 7-2
  CREF, 11-1
  SRCCOM, 26-2
Output files,
  BATCH, 6-2
  Editor, 4-6
Output specifications,
  DIRECT, 12-2
  FOTP, 16-1
  RESORC, 22-1

Paper tape system loading, 1-20
Period (.) character -- see DOT (.) character
Peripheral Interchange Program, (PIP), 20-1
  error messages, 20-8
  examples of specification commands, 20-6
  options, 20-1
Permanent device names, 2-1
PIP -- see Peripheral Interchange Program
PIP10 utility program, 21-1
  error messages, 21-3
  options, 21-2
Postdeletion, FOTP, 16-7
Predeletion, FOTP, 16-7
PRINT command, 3-47
Pseudo-op handling, CREF, 11-3
Punch and Compare program -- see EPIC
PUNCH command, CCL, 3-48
Punched cards, 6-11

Question mark,
  wild character, 3-7

RENAME command, CCL, 3-50
RES command, CCL, 3-51
RESORC utility program, 22-1
  device types, 22-3
  error messages, 22-6
  handlers, 22-3
  options, 22-2
REWIND command, CCL, 3-52
RKLFMT disk formatter program, 23-1

RUN command, keyboard monitor, 3-53
RXCOPY utility program, 24-1

SET utility program, 25-1
Single character search, Editor, 4-12
SKIP command, 3-57
Source Compare Utility Program (SRCCOM), 26-1
  error messages, 26-4
  loading, 26-1
  options, 26-2
  output, 26-2
Spool device files, BATCH, 6-2
Square brackets ([]) characters, Command Decoder, 3-5
SQUISH command, CCL, 3-59
SRCCOM -- see Source Compare Utility program
START command, keyboard monitor, 3-58
Stopping execution, CTRL/C, 3-10
SUBMIT command, CCL, 3-60
System conventions, keyboard monitor, 7-1
System devices, 2-1

TDCOPY, 13-10
TDFRMT, 13-5
Terminate command, 3-62
Text mode, Editor, 4-3
TYPE command, CCL, 3-63

UA, UB, UC commands, CCL, 3-63
UNLOAD command, CCL, 3-65

VERSION command, 3-66

Wild card construction, 3-7

ZERO command, CCL, 3-67

OS/8
System Reference
Manual
AA-H607A-TA

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

Please cut along this line.

FOLD HERE

DO NOT TEAR – FOLD HERE AND STAPLE

digital

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
SOFTWARE DOCUMENTATION
146 MAIN STREET – ML5-5/E39
MAYNARD, MASSACHUSETTS 01754