UPDATE NOTICE No. 1

OS/78 User's Manual

Order No. AD-5748B-T1

September 1979

## NEW AND CHANGED INFORMATION

This update contains the Table of Contents and revised information for the OS/78 User's Manual.

Copyright © 1979 by Digital Equipment Corporation

## INSTRUCTIONS

Place the following pages in the OS/78 User's Manual as replacements for, or additions to, current pages. The changes made on replacement pages are indicated in the outside margin by change bars ( ▌ ) for additions and by bullets (•) for deletions.

| Old Page | New Page |
|---|---|
| Title Page/Copyright | Title Page/Copyright |
| -- | iii through xv |
| 2-9/2-10 | 2-9/2-10 |
| 2-15/2-16 | 2-15/2-16 |
| 2-17/2-18 | 2-17/2-18 |
| 2-23/2-24 | 2-23/2-24 |
| 2-25/2-26 | 2-25/2-26 |
| 2-27/2-28 | 2-27/2-28 |
| 2-29/2-30 | 2-29/2-30 |
| 2-31/2-32 | 2-31/2-32 |
| 3-33/3-34 | 3-33/3-34 |
| 3-35/3-36 | 3-35/3-36 |
| 6-15/6-16 | 6-15/6-16 |
| 6-25/6-26 | 6-25/6-26 |
| 6-61/6-62 | 6-61/6-62 |
| -- | 6-62.1/Blank |
| 6-67/6-68 | 6-67/6-68 |
| -- | 6-68.1/Blank |
| 6-83/6-84 | 6-83/6-84 |
| -- | 6-84.1/Blank |
| Index-1/Index-15 | Index-1/Index-15 |

# OS/78
# User's Manual

Order No. AD-5748B-T1

September 1979

This document is the user's manual for the OS/78 V3 operating system. Its purpose is to acquaint new users with the operating system designed for the DECstation 78/88 minicomputers. This manual presents the background material for getting on the air, and a detailed description of the OS/78 commands that are used to direct computer operations. It also describes the OS/78 Editor, PAL8 assembly language, and the two high-level languages supported by the system, BASIC and FORTRAN IV. Command examples and demonstration programs are contained throughout the manual.

| | |
|---|---|
| SUPERSESSION/UPDATE INFORMATION: | This manual supersedes previous editions, Order Numbers DEC-S8-OS78A-A-D, published 1977, and DEC-S8-OS78A-A-DN1, published 1978. This manual includes Update Notice No. 1. |
| OPERATING SYSTEM AND VERSION: | OS/78 V3 |

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, August 1979
Revised: September 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.

Copyright © 1979 by Digital Equipment Corporation

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 |
|---|---|---|---|
| PREFACE | | | iii |
| CHAPTER | 1 | INTRODUCTION TO THE SYSTEM | 1-1 |
| | 1.1 | DECSTATION 78/88 MINICOMPUTER SYSTEM HARDWARE | 1-1 |
| | 1.2 | OS/78 SOFTWARE | 1-2 |
| CHAPTER | 2 | GETTING ON THE AIR WITH OS/78 | 2-1 |
| | 2.1 | STARTING THE SYSTEM | 2-1 |
| | 2.1.1 | Loading the Diskette | 2-1 |
| | 2.1.2 | Loading the RL01K Cartridge Disk Pack | 2-3 |
| | 2.1.3 | Unloading the RL01K Cartridge Disk Pack | 2-5 |
| | 2.1.4 | Starting the System | 2-5 |
| | 2.1.4.1 | Setting Terminal Characteristics | 2-5 |
| | 2.1.4.2 | Bootstrapping OS/78 | 2-6 |
| | 2.1.5 | Using the Terminal | 2-7 |
| | 2.2 | MAKING A BACKUP COPY OF THE OS/78 SOFTWARE | 2-8 |
| | 2.2.1 | DECstations with RX01 and RX02 Disk Systems Only | 2-8 |
| | 2.2.2 | DECstations with RL01 Disk Systems Only | 2-8 |
| | 2.2.3 | DECstations with RX02 and RL01 Disk Systems | 2-9 |
| | 2.3 | DEMONSTRATION PROGRAMS | 2-10 |
| | 2.4 | DEVICE AND FILE NAMES | 2-10 |
| | 2.4.1 | Devices | 2-10 |
| | 2.4.2 | File Names and Extensions | 2-11 |
| | 2.5 | ENTERING MONITOR COMMANDS | 2-12 |
| | 2.5.1 | OS/78 Command Format | 2-13 |
| | 2.5.2 | Incorrect Commands | 2-14 |
| | 2.5.3 | Correcting Errors | 2-15 |
| | 2.5.4 | Using Input/Output Options | 2-15 |
| | 2.5.4.1 | Equal Sign Construction | 2-16 |
| | 2.5.4.2 | Slash Construction | 2-16 |
| | 2.5.4.3 | Parentheses Construction | 2-16 |
| | 2.5.4.4 | Square Bracket Construction | 2-16 |
| | 2.5.4.5 | General-Purpose Dash Options | 2-16 |
| | 2.5.5 | Remembering Previous Arguments | 2-17 |
| | 2.5.6 | Using Wildcards | 2-18 |
| | 2.5.6.1 | Wildcard Input File Specifications | 2-19 |
| | 2.5.6.2 | Wildcard Output File Specifications | 2-19 |
| | 2.5.7 | Indirect Commands (At Symbol (@) Construction) | 2-20 |
| | 2.5.8 | Asterisk (*) Prompting Symbol | 2-21 |
| | 2.6 | USING DEFAULTS | 2-21 |
| | 2.7 | WHERE TO GET MORE INFORMATION | 2-22 |
| | 2.8 | FILE-STRUCTURED DEVICES | 2-23 |
| | 2.8.1 | Using RX01 and RX02 Diskettes | 2-24 |
| | 2.8.2 | Using the RL01 Cartridge Disk Pack | 2-24 |
| | 2.8.3 | Using the VXA0 Extended-Memory Device | 2-25 |
| | 2.9 | FILES | 2-25 |

| | | | |
|---|---|---|---|
| | 2.9.1 | File Directories | 2-25 |
| | 2.9.2 | File Types | 2-25 |
| | 2.9.3 | ASCII File Format | 2-26 |
| | 2.10 | OS/78 COMMAND SUMMARY | 2-26 |
| CHAPTER | 3 | OS/78 COMMANDS | 3-1 |
| | 3.1 | ASSIGN COMMAND | 3-2 |
| | 3.2 | BASIC COMMAND | 3-3 |
| | 3.3 | BOOT COMMAND | 3-4 |
| | 3.4 | CANCEL COMMAND | 3-5 |
| | 3.5 | COMPARE COMMAND | 3-6 |
| | 3.5.1 | Using COMPARE's Options | 3-8 |
| | 3.6 | COMPILE COMMAND | 3-10 |
| | 3.7 | COPY COMMAND | 3-12 |
| | 3.7.1 | File Transfer | 3-12 |
| | 3.7.2 | Examples of COPY Commands | 3-12 |
| | 3.7.3 | Predeletion | 3-13 |
| | 3.7.4 | Postdeletion | 3-13 |
| | 3.7.5 | Considering the Date on a Transfer | 3-14 |
| | 3.7.6 | File Protection During a Transfer Operation | 3-14 |
| | 3.7.7 | Terminating COPY Operations | 3-15 |
| | 3.8 | CREATE COMMAND | 3-18 |
| | 3.9 | CREF COMMAND | 3-19 |
| | 3.10 | DATE COMMAND | 3-21 |
| | 3.11 | DEASSIGN COMMAND | 3-22 |
| | 3.12 | DELETE COMMAND | 3-23 |
| | 3.13 | DIRECT COMMAND | 3-25 |
| | 3.13.1 | Directory Listing Selected by Date | 3-26 |
| | 3.13.2 | Directory Format Selection | 3-27 |
| | 3.14 | DUPLICATE COMMAND | 3-28 |
| | 3.14.1 | Changing Devices Before and After Executing the DUPLICATE Command | 3-28 |
| | 3.14.2 | Performing a Read Check | 3-29 |
| | 3.14.3 | Transfer Without Checking for Identical Contents | 3-29 |
| | 3.14.4 | Check for Identical Contents Without Transferring | 3-30 |
| | 3.14.5 | Formatting Diskettes (RX02 Drives Only) | 3-30 |
| | 3.15 | EDIT COMMAND | 3-31 |
| | 3.16 | EXECUTE COMMAND | 3-34 |
| | 3.17 | FORMAT COMMAND | 3-35 |
| | 3.18 | GET COMMAND | 3-37 |
| | 3.19 | HELP COMMAND | 3-38 |
| | 3.20 | LIST COMMAND | 3-40 |
| | 3.21 | LOAD COMMAND | 3-42 |
| | 3.21.1 | PAL8 Absolute Binary Files | 3-42 |
| | 3.21.1.1 | Concatenated Programs (/S option) | 3-42 |
| | 3.21.1.2 | Loading Programs in Specified Areas (/ff and =ffnnnn options) | 3-42 |
| | 3.21.1.3 | Programs not Using Fields 0 and 1 (/8 and /9 options) | 3-43 |
| | 3.21.1.4 | Editing or Patching a SAVE File (/I option) | 3-43 |
| | 3.21.1.5 | Execution After Loading (/G option) | 3-43 |
| | 3.21.1.6 | Clearing Memory After Loading (/R option) | 3-44 |
| | 3.21.2 | FORTRAN Relocatable Binary Files | 3-44 |
| | 3.21.2.1 | Specifying Additional Files (/C option) | 3-44 |
| | 3.21.2.2 | Including System Symbols in the Symbol Map (/S option) | 3-44 |
| | 3.21.2.3 | Execution After Loading (/G option) | 3-45 |
| | 3.22 | MAP COMMAND | 3-46 |
| | 3.23 | MEMORY COMMAND | 3-50 |
| | 3.24 | ODT COMMAND | 3-52 |
| | 3.25 | PAL COMMAND | 3-53 |

| | | | |
|---|---|---|---|
| | 3.26 | QUEUE COMMAND | 3-54 |
| | 3.27 | R COMMAND | 3-55 |
| | 3.28 | RENAME COMMAND | 3-56 |
| | 3.29 | REQUEST COMMAND | 3-57 |
| | 3.30 | RUN COMMAND | 3-58 |
| | 3.31 | SAVE COMMAND | 3-59 |
| | 3.31.1 | Restrictions on Arguments of the SAVE Command | 3-60 |
| | 3.32 | SET COMMAND | 3-62 |
| | 3.32.1 | SET Command Forms | 3-62 |
| | 3.32.1.1 | ARROW | 3-62 |
| | 3.32.1.2 | COL n | 3-63 |
| | 3.32.1.3 | ECHO | 3-63 |
| | 3.32.1.4 | ESC | 3-64 |
| | 3.32.1.5 | HEIGHT m | 3-64 |
| | 3.32.1.6 | INIT cmd | 3-64 |
| | 3.32.1.7 | LC | 3-64 |
| | 3.32.1.8 | PAGE | 3-65 |
| | 3.32.1.9 | PAUSE | 3-65 |
| | 3.32.1.10 | READONLY | 3-65 |
| | 3.32.1.11 | SCOPE | 3-66 |
| | 3.32.1.12 | WIDTH = n | 3-66 |
| | 3.32.1.13 | HANDLER | 3-66 |
| | 3.33 | SQUISH COMMAND | 3-70 |
| | 3.34 | START COMMAND | 3-71 |
| | 3.35 | SUBMIT COMMAND | 3-72 |
| | 3.35.1 | Processing and Terminating a Batch Input File | 3-72 |
| | 3.35.2 | Spooling | 3-73 |
| | 3.36 | TERMINATE COMMAND | 3-75 |
| | 3.37 | TYPE COMMAND | 3-76 |
| | 3.37.1 | Using TYPE Options | 3-77 |
| | 3.38 | UA, UB, UC COMMANDS | 3-78 |
| | 3.39 | ZERO COMMAND | 3-79 |
| CHAPTER | 4 | THE SYMBOLIC EDITOR | 4-1 |
| | 4.1 | INTRODUCTION | 4-1 |
| | 4.2 | CALLING THE EDITOR | 4-1 |
| | 4.2.1 | Creating a New File -- The CREATE Command | 4-1 |
| | 4.2.2 | Editing an Existing File -- The EDIT Command | 4-2 |
| | 4.3 | MODES OF OPERATION | 4-2 |
| | 4.3.1 | Text Mode | 4-3 |
| | 4.3.2 | Command Mode | 4-5 |
| | 4.3.2.1 | Input Commands | 4-5 |
| | 4.3.2.2 | Listing Commands | 4-6 |
| | 4.3.2.3 | Output Commands | 4-7 |
| | 4.3.2.4 | Editing Commands | 4-9 |
| | 4.3.2.5 | Search Commands | 4-10 |
| | 4.3.2.6 | Special Command Mode Characters | 4-11 |
| | 4.4 | SEARCHING A TEXT | 4-13 |
| | 4.4.1 | Single-Character Search -- The S Command | 4-13 |
| | 4.4.2 | The Character String Search | 4-14 |
| | 4.4.2.1 | Intrabuffer String Search | 4-14 |
| | 4.4.2.2 | Interbuffer String Search -- J Command | 4-17 |
| | 4.5 | A SAMPLE EDITING JOB | 4-18 |
| | 4.6 | EDITOR OPTIONS | 4-21 |
| | 4.7 | EDITOR ERROR MESSAGES | 4-21 |
| | 4.8 | SUMMARY OF EDITOR COMMANDS AND SPECIAL CHARACTERS | 4-23 |
| CHAPTER | 5 | THE PAL8 ASSEMBLER | 5-1 |

| | | |
|---|---|---|
| 5.1 | INTRODUCTION | 5-1 |
| 5.2 | CREATING AND RUNNING A PAL8 PROGRAM | 5-4 |
| 5.2.1 | Creating a Program | 5-4 |
| 5.2.2 | Assembling a Program | 5-5 |
| 5.2.3 | Loading and Saving a Program | 5-6 |
| 5.2.4 | Executing the Program | 5-7 |
| 5.2.5 | Getting and Using a Cross-Reference Listing | 5-7 |
| 5.2.6 | Obtaining a Memory Map | 5-8 |
| 5.3 | PAL8 OPTIONS | 5-9 |
| 5.4 | CHARACTER SET | 5-10 |
| 5.5 | STATEMENTS | 5-11 |
| 5.5.1 | Labels | 5-11 |
| 5.5.2 | Instructions | 5-11 |
| 5.5.3 | Operands | 5-12 |
| 5.5.4 | Comments | 5-12 |
| 5.6 | FORMAT CHARACTERS | 5-12 |
| 5.6.1 | Form Feed | 5-12 |
| 5.6.2 | Tab | 5-12 |
| 5.6.3 | Statement Terminators | 5-13 |
| 5.7 | NUMBERS | 5-13 |
| 5.8 | SYMBOLS | 5-13 |
| 5.8.1 | Permanent Symbols | 5-14 |
| 5.8.2 | User-Defined Symbols | 5-14 |
| 5.8.3 | Current Location Counter | 5-14 |
| 5.8.4 | Symbol Table | 5-16 |
| 5.8.5 | Direct Assignment Statements | 5-16 |
| 5.8.6 | Symbol Instructions | 5-17 |
| 5.8.7 | Symbolic Operands | 5-18 |
| 5.9 | EXPRESSIONS | 5-18 |
| 5.9.1 | Operators | 5-18 |
| 5.9.2 | Special Characters | 5-21 |
| 5.9.2.1 | Period (.) | 5-21 |
| 5.9.2.2 | Double Quote (") | 5-22 |
| 5.9.2.3 | Parentheses ( ) and Brackets [ ] | 5-22 |
| 5.9.2.4 | Angle Brackets (< >) | 5-23 |
| 5.9.2.5 | Dollar Sign ($) | 5-24 |
| 5.9.3 | Other Characters | 5-24 |
| 5.10 | INSTRUCTION SET | 5-24 |
| 5.10.1 | Memory Reference Instructions | 5-24 |
| 5.10.2 | Microinstructions | 5-25 |
| 5.10.2.1 | Operate Microinstructions | 5-26 |
| 5.10.2.2 | Input/Output Transfer Microinstructions | 5-28 |
| 5.10.3 | Autoindexing | 5-28 |
| 5.11 | PSEUDO-OPERATORS | 5-29 |
| 5.11.1 | Indirect and Page Zero Addressing | 5-29 |
| 5.11.2 | Extended Memory | 5-29 |
| 5.11.2.1 | FIELD Pseudo-Operator | 5-29 |
| 5.11.2.2 | Specifying Data and Instruction Fields | 5-30 |
| 5.11.3 | Resetting the Location Counter | 5-33 |
| 5.11.4 | Reserving Memory | 5-33 |
| 5.11.5 | Relocation Pseudo-Operator | 5-34 |
| 5.11.6 | Suppressing the Listing | 5-34 |
| 5.11.7 | Controlling Page Format | 5-34 |
| 5.11.8 | Altering the Permanent Symbol Table | 5-35 |
| 5.11.9 | Conditional Assembly Pseudo-Operators | 5-36 |
| 5.11.10 | Radix Control | 5-36 |
| 5.11.11 | Entering Text Strings | 5-37 |
| 5.11.12 | End-of-File Signal | 5-37 |
| 5.11.13 | Use of DEVICE and FILENAME Pseudo-Operators | 5-37 |
| 5.12 | LINK GENERATION AND STORAGE | 5-38 |
| 5.13 | TERMINATING ASSEMBLY | 5-39 |
| 5.14 | DIAGNOSTIC ERROR MESSAGES | 5-39 |
| 5.15 | PAL8 PERMANENT SYMBOL TABLE | 5-41 |

| | | | |
|---|---|---|---|
| CHAPTER | 6 | BASIC | 6-1 |
| | 6.1 | OVERVIEW | 6-1 |
| | 6.1.1 | Writing a BASIC Program | 6-1 |
| | 6.1.2 | The BASIC Character Set | 6-1 |
| | 6.1.3 | Using BASIC | 6-3 |
| | 6.2 | BASIC COMMANDS | 6-3 |
| | 6.2.1 | Entering a New Program -- the NEW Command | 6-4 |
| | 6.2.2 | Calling for an Old Program -- the OLD Command | 6-4 |
| | 6.2.3 | Running a Program -- the RUN Command | 6-5 |
| | 6.2.4 | Displaying a Program -- the LIST Command | 6-5 |
| | 6.2.5 | Deleting Text -- the DELETE Command | 6-7 |
| | 6.2.6 | Substituting Text -- the EDIT Command | 6-7 |
| | 6.2.7 | Automatic Line Numbering -- the SEQUENCE Command | 6-8 |
| | 6.2.8 | Merging Text -- the WEAVE Command | 6-9 |
| | 6.2.9 | Storing a Program -- the SAVE Command | 6-10 |
| | 6.2.10 | Renaming a Program -- the NAME Command | 6-10 |
| | 6.2.11 | Erasing the Workspace -- the SCRATCH Command | 6-11 |
| | 6.2.12 | Leaving BASIC -- the BYE Command | 6-11 |
| | 6.2.13 | Resequencing a Program -- Calling RESEQ | 6-11 |
| | 6.2.14 | Key Commands | 6-12 |
| | 6.2.14.1 | Correcting Typing and Format Errors -- DELETE, CTRL/U | 6-12 |
| | 6.2.14.2 | Eliminating Program Lines -- RETURN | 6-12 |
| | 6.2.14.3 | Interrupting Program Execution -- CTRL/C | 6-12 |
| | 6.2.14.4 | Controlling Program Listings on the Terminal -- CTRL/S, CTRL/Q, and CTRL/O | 6-12 |
| | 6.3 | ELEMENTS OF BASIC | 6-13 |
| | 6.3.1 | Constants | 6-13 |
| | 6.3.1.1 | Numeric Constants | 6-13 |
| | 6.3.1.2 | String Constants | 6-14 |
| | 6.3.2 | Variables | 6-14 |
| | 6.3.2.1 | Numeric Variables | 6-15 |
| | 6.3.2.2 | String Variables | 6-15 |
| | 6.3.2.3 | Subscripted Variables | 6-16 |
| | 6.3.3 | Expressions | 6-17 |
| | 6.3.3.1 | Arithmetic Expressions | 6-17 |
| | 6.3.3.2 | Relational Expressions | 6-18 |
| | 6.3.3.3 | String Concatenation | 6-18 |
| | 6.4 | OS/78 COMMERCIAL ARITHMETIC | 6-19 |
| | 6.4.1 | BASIC String Numbers | 6-20 |
| | 6.4.2 | Truncation Versus Rounding Off | 6-21 |
| | 6.4.3 | Using Relational Operators with String Numbers | 6-21 |
| | 6.5 | FORMATTING BASIC STATEMENTS | 6-22 |
| | 6.6 | THE ASSIGNMENT STATEMENT -- LET | 6-23 |
| | 6.7 | THE DIMENSION STATEMENT | 6-24 |
| | 6.8 | THE COMMENT STATEMENT -- REMARK AND EXCLAMATION POINT | 6-26 |
| | 6.9 | INPUT AND OUTPUT | 6-26 |
| | 6.9.1 | Input | 6-26 |
| | 6.9.2 | READ, DATA, and RESTORE | 6-28 |
| | 6.9.3 | PRINT | 6-29 |
| | 6.9.3.1 | Printing Zones -- Format Control Characters | 6-30 |
| | 6.9.3.2 | Printing Numbers and Strings | 6-31 |
| | 6.9.3.3 | Printing with the TAB and PNT Functions | 6-31 |
| | 6.9.4 | PRINT USING | 6-32 |
| | 6.9.4.1 | Representing Numeric Fields | 6-33 |
| | 6.9.4.2 | Representing Special Functions | 6-34 |

6.9.4.3 Letters and Numbers 6-35
6.9.4.4 Printing Format String Characters 6-35
6.9.4.5 Numeric Field List Use 6-36
6.10 CONTROL STATEMENTS 6-36
6.10.1 Unconditional Transfer -- GOTO 6-36
6.10.2 Conditional Transfer -- IF GOTO 6-37
6.10.3 Semiconditional Transfer -- ON GOTO 6-38
6.10.4 Looping -- FOR, STEP, and NEXT 6-39
6.10.4.1 Nested Loops 6-40
6.10.5 Stopping -- END and STOP 6-41
6.10.6 Jumping to Subroutines -- GOSUB and RETURN 6-42
6.10.7 Semiconditional Jump to Subroutine -- ON GOSUB 6-43
6.11 FUNCTIONS 6-43
6.11.1 Numeric Functions 6-44
6.11.1.1 Calculating Sine -- SIN 6-45
6.11.1.2 Calculating Cosine -- COS 6-45
6.11.1.3 Calculating the Arctangent -- ATN 6-45
6.11.1.4 Calculating the Tangent -- TAN 6-46
6.11.1.5 Finding the Square Root -- SQR 6-46
6.11.1.6 The Exponential Function -- EXP 6-46
6.11.1.7 Calculating the Natural Logarithm -- LOG 6-47
6.11.1.8 The Integer Function -- INT 6-47
6.11.1.9 The Absolute Value Function -- ABS 6-48
6.11.1.10 The Sign Function -- SGN 6-48
6.11.1.11 Random Numbers -- RND 6-48
6.11.2 String Functions 6-49
6.11.2.1 Finding the Length of a String -- LEN 6-50
6.11.2.2 Finding a Substring -- POS 6-51
6.11.2.3 Returning a String -- SEG$ 6-52
6.11.2.4 Converting a Character to ASCII Code -- ASC 6-52
6.11.2.5 Converting ASCII Code to a Character -- CHR$ 6-53
6.11.2.6 Converting Numbers from String to Numeric Format 6-53
6.11.2.7 Converting a Number to a String -- STR$ 6-54
6.11.2.8 Converting Lower to Upper Case 6-54
6.11.2.9 Converting Octal to Decimal -- OCT 6-54
6.11.2.10 Converting Decimal to Octal -- OCS$ 6-55
6.11.2.11 Returning a String Integer -- FIX$ 6-55
6.11.3 Issuing Commands to the OS/78 Monitor -- CCL 6-55
6.11.4 The FNa Function and the DEF Statement 6-56
6.11.5 The Debugging Function -- TRC 6-57
6.11.6 Calling for the Date -- the DAT$ Function 6-57
6.11.7 Logical Functions 6-58
6.11.7.1 AND 6-58
6.11.7.2 IOR 6-58
6.11.8 The KEY$ Function 6-59
6.11.9 The CUR$ Function 6-59
6.11.10 The PMT$ Function 6-60
6.11.11 The COL Function 6-60
6.11.12 Printing Functions 6-60
6.11.12.1 The TAB Function 6-61
6.11.12.2 The PNT Function 6-61
6.12 FILE STATEMENTS 6-62
6.12.1 File Control 6-63
6.12.1.1 Opening a File 6-63
6.12.2 File I/O 6-64
6.12.2.1 Reading Data from a File -- INPUT# 6-65
6.12.2.2 Writing Data on a File -- PRINT# 6-65
6.12.2.3 Formatting Output to a File -- PRINT# USING 6-67
6.12.2.4 Resetting a File -- RESTORE# 6-67
6.12.2.5 Checking for Open Files -- IF OPEN# 6-68
6.12.2.6 Checking for End-of-File -- the IF END# Statement 6-68.1

| | | | |
|---|---|---|---|
| | 6.13 | DIRECT RECORD I/O | 6-69 |
| | 6.13.1 | Specifying Record-Size with FILEV# | 6-69 |
| | 6.13.2 | Opening a File for Record I/O | 6-70 |
| | 6.13.3 | Defining Record Fields | 6-70 |
| | 6.13.4 | Reading and Updating Records -- PUT# and GET# | 6-71 |
| | 6.13.5 | CLOSE# Statement | 6-71 |
| | 6.13.6 | Working with Record I/O | 6-72 |
| | 6.14 | SEGMENTING PROGRAMS -- THE CHAIN STATEMENT | 6-74 |
| | 6.15 | BASIC ESCAPE SEQUENCES | 6-76 |
| | 6.15.1 | Calling for ESCape Sequences with the PNT Function | 6-76 |
| | 6.15.2 | Using the KEY$ Function | 6-77 |
| | 6.15.3 | Defining a User Function | 6-78 |
| | 6.16 | CREATING A MEMORY-IMAGE OF A BASIC PROGRAM WITH COMPILE | 6-79 |
| | 6.17 | BASIC OPTIONS | 6-79 |
| | 6.18 | INTERACTION BETWEEN BASIC AND BATCH | 6-80 |
| | 6.19 | BASIC OVERLAYS -- BASIC.OV | 6-80 |
| | 6.20 | SUMMARY OF BASIC COMMANDS | 6-80 |
| | 6.21 | SUMMARY OF BASIC STATEMENTS | 6-81 |
| | 6.22 | SUMMARY OF BASIC FUNCTIONS | 6-83 |
| | 6.23 | ERROR MESSAGES | 6-86 |
| | 6.23.1 | Editor Error Messages | 6-86 |
| | 6.23.2 | Compiler Error Messages | 6-87 |
| | 6.23.3 | Run-Time System Error Messages | 6-87 |
| CHAPTER | 7 | FORTRAN IV | 7-1 |
| | 7.1 | OVERVIEW | 7-1 |
| | 7.1.1 | The COMPILER | 7-4 |
| | 7.1.1.1 | Compiler Options | 7-5 |
| | 7.1.1.2 | Compiler Error Messages | 7-5 |
| | 7.1.2 | The LOADER | 7-7 |
| | 7.1.2.1 | Specifying I/O Devices | 7-7 |
| | 7.1.2.2 | Running Subprograms | 7-8 |
| | 7.1.2.3 | LOADER Options | 7-10 |
| | 7.1.2.4 | Loader Error Messages | 7-11 |
| | 7.1.3 | The Run-Time System (FRTS) | 7-12 |
| | 7.1.3.1 | Run-Time System Options | 7-15 |
| | 7.1.3.2 | Run-Time System Error Messages | 7-17 |
| | 7.1.4 | The Library | 7-19 |
| | 7.2 | THE FORTRAN IV SOURCE LANGUAGE | 7-19 |
| | 7.2.1 | The FORTRAN Character Set | 7-20 |
| | 7.2.2 | Elements of a FORTRAN Program | 7-21 |
| | 7.2.2.1 | Statements | 7-21 |
| | 7.2.2.2 | Comments | 7-22 |
| | 7.2.3 | FORTRAN Lines | 7-22 |
| | 7.2.3.1 | Using a Text Editor | 7-22 |
| | 7.2.3.2 | Statement Label Field | 7-23 |
| | 7.2.3.3 | Comment Indicator | 7-23 |
| | 7.2.3.4 | Continuation Indicator Field | 7-24 |
| | 7.2.3.5 | Statement Field | 7-24 |
| | 7.2.3.6 | Identification Field | 7-25 |
| | 7.2.4 | Blank Lines | 7-25 |
| | 7.2.5 | Line Format Summary | 7-25 |
| | 7.3 | FORTRAN STATEMENT COMPONENTS | 7-26 |
| | 7.3.1 | Symbolic Names | 7-26 |
| | 7.3.2 | Data Types | 7-27 |
| | 7.3.3 | Constants | 7-27 |
| | 7.3.3.1 | Integer Constants | 7-28 |
| | 7.3.3.2 | Real Constants | 7-28 |
| | 7.3.3.3 | Logical Constants | 7-30 |
| | 7.3.3.4 | Octal Constants | 7-30 |

| | | |
|---|---|---|
| 7.3.3.5 | Hollerith Constants and Alphanumerical Literals | 7-31 |
| 7.3.4 | Variables | 7-32 |
| 7.3.4.1 | Data Type Specification | 7-33 |
| 7.3.4.2 | Default Data Types | 7-33 |
| 7.3.5 | Arrays | 7-33 |
| 7.3.5.1 | Array Declarations | 7-34 |
| 7.3.5.2 | Array Storage (Order of Subscript Progression) | 7-35 |
| 7.3.5.3 | Subscripts | 7-36 |
| 7.3.5.4 | Data Type of an Array | 7-37 |
| 7.3.5.5 | Array Reference Without Subscripts | 7-37 |
| 7.3.5.6 | Adjustable Arrays | 7-37 |
| 7.4 | EXPRESSIONS | 7-38 |
| 7.4.1 | Arithmetic Expressions | 7-38 |
| 7.4.1.1 | Rules for Writing Arithmetic Expressions | 7-39 |
| 7.4.1.2 | Evaluation Hierarchy | 7-40 |
| 7.4.1.3 | Data Type of an Arithmetic Expression | 7-40 |
| 7.4.2 | Relational Expressions | 7-41 |
| 7.4.3 | Logical Expressions | 7-42 |
| 7.4.3.1 | Logical Operator Hierarchy | 7-43 |
| 7.4.4 | Use of Parentheses | 7-44 |
| 7.5 | ASSIGNMENT STATEMENTS | 7-45 |
| 7.5.1 | Arithmetic Assignment Statement | 7-45 |
| 7.5.2 | Logical Assignment Statements | 7-46 |
| 7.6 | SPECIFICATION STATEMENTS | 7-46 |
| 7.6.1 | Type Declaration Statements | 7-47 |
| 7.6.2 | DIMENSION Statement | 7-47 |
| 7.6.3 | EXTERNAL Statement | 7-48 |
| 7.6.4 | COMMON Statement | 7-50 |
| 7.6.5 | EQUIVALENCE Statement | 7-52 |
| 7.6.5.1 | Making Arrays Equivalent | 7-53 |
| 7.6.5.2 | EQUIVALENCE and COMMON Interaction | 7-54 |
| 7.7 | DATA STATEMENTS AND BLOCK DATA SUBPROGRAMS | 7-54 |
| 7.8 | CONTROL STATEMENTS | 7-56 |
| 7.8.1 | GOTO Statements | 7-57 |
| 7.8.1.1 | Unconditional GOTO Statement | 7-57 |
| 7.8.1.2 | Computed GOTO Statement | 7-58 |
| 7.8.1.3 | ASSIGN and ASSIGNed GOTO Statement | 7-58 |
| 7.8.2 | IF Statements | 7-60 |
| 7.8.2.1 | Arithmetic IF Statement | 7-60 |
| 7.8.2.2 | Logical IF Statement | 7-62 |
| 7.8.3 | DO Statement | 7-62 |
| 7.8.3.1 | DO Iteration Control | 7-63 |
| 7.8.3.2 | Nested DO Loops | 7-64 |
| 7.8.3.3 | Control Transfers in DO Loops | 7-65 |
| 7.8.3.4 | Extended Range | 7-65 |
| 7.8.4 | CONTINUE Statement | 7-66 |
| 7.8.5 | PAUSE Statement | 7-67 |
| 7.8.6 | STOP Statement | 7-68 |
| 7.8.7 | END Statement | 7-68 |
| 7.9 | SUBPROGRAMS | 7-69 |
| 7.9.1 | Subprogram Arguments | 7-69 |
| 7.9.2 | User-Written Subprograms | 7-70 |
| 7.9.2.1 | Arithmetic Statement Functions (ASE) | 7-70 |
| 7.9.2.2 | FUNCTION Subprogram | 7-72 |
| 7.9.2.3 | SUBROUTINE Subprograms | 7-73 |
| 7.9.3 | CALL Statement | 7-74 |
| 7.9.4 | RETURN Statement | 7-75 |
| 7.9.5 | FORTRAN Library Functions | 7-76 |
| 7.10 | INPUT/OUTPUT STATEMENTS | 7-76 |
| 7.10.1 | Defining I/O Operations | 7-76 |
| 7.10.1.1 | Input/Output Devices and Logical Unit Numbers | 7-77 |

7.10.1.2 Format Specifiers 7-77
7.10.1.3 Input/Output Record Transmission 7-77
7.10.2 Input/Output Lists 7-77
7.10.2.1 Simple Lists 7-78
7.10.2.2 Implied DO Lists 7-78
7.10.3 Input/Output Forms 7-80
7.10.3.1 Unformatted Sequential Input/Output 7-80
7.10.3.2 Formatted Sequential Input/Output 7-80
7.10.3.3 Unformatted Direct Access Input/Output 7-80
7.10.4 READ Statements 7-80
7.10.4.1 Unformatted Sequential READ Statement 7-80
7.10.4.2 Formatted Sequential READ Statement 7-81
7.10.4.3 Unformatted Direct Access READ Statement 7-82
7.10.5 WRITE Statements 7-83
7.10.5.1 Unformatted Sequential WRITE Statement 7-83
7.10.5.2 Formatted Sequential WRITE Statement 7-84
7.10.5.3 Unformatted Direct Access WRITE Statement 7-85
7.10.6 Auxiliary Input/Output Statements 7-85
7.10.6.1 BACKSPACE Statement 7-86
7.10.6.2 DEFINE FILE Statement 7-86
7.10.6.3 END FILE Statement 7-87
7.10.6.4 REWIND Statement 7-87
7.11 FORMAT STATEMENTS 7-88
7.11.1 Field Descriptors 7-89
7.11.1.1 I Field Descriptor 7-90
7.11.1.2 F Field Descriptor 7-91
7.11.1.3 E Field Descriptor 7-92
7.11.1.4 G Field Descriptor 7-93
7.11.1.5 L Field Descriptor 7-94
7.11.1.6 A Field Descriptor 7-95
7.11.1.7 H Field Descriptor 7-96
7.11.1.8 Alphanumeric Literals 7-97
7.11.1.9 X Field Descriptor 7-97
7.11.1.10 T Field Descriptor 7-97
7.11.2 Scale Factor 7-99
7.11.3 Grouping and Group Repeat Specifications 7-100
7.11.4 Carriage Control 7-100
7.11.5 Format Specification Separators 7-101
7.11.6 Short Field Termination 7-102
7.11.7 Format Control Interaction with Input/Output Lists 7-103
7.11.8 Summary of Rules for Format Statements 7-103
7.12 LIBRARY FUNCTIONS AND SUBROUTINES 7-105
7.12.1 ABS (Single-Precision Absolute Value) 7-105
7.12.2 ACOS (Single-Precision Arc-Cosine Function) 7-105
7.12.3 AINT (Single-Precision Floating-Point to Integer) 7-106
7.12.4 ALOG (Single-Precision Natural Logarithm) 7-106
7.12.5 ALOG10 (Single-Precision Common Logarithm) 7-106
7.12.6 AMAX0 (Single-Precision Maximum Value) 7-106
7.12.7 AMAX1 (Single-Precision Maximum Value) 7-106
7.12.8 AMIN0 (Single-Precision Minimum Value) 7-106
7.12.9 AMIN1 (Single-Precision Minimum Value) 7-106
7.12.10 AMOD (Single-Precision A Modulo B) 7-106
7.12.11 ASIN (Single-Precision Arc-Sine) 7-107
7.12.12 ATAN (Single-Precision Arc-Tangent) 7-107
7.12.13 ATAN2 (Single-Precision Arc-Tangent of Two Arguments) 7-107
7.12.14 CGET (Character Get Subroutine) 7-107
7.12.15 CHKEOF (Check for End-of-File Subroutine) 7-108
7.12.16 CLOCK (Initialize Clock Subroutine) 7-108
7.12.17 COS (Single-Precision Cosine Function) 7-108

7.12.18 COSH (Single-Precision Hyperbolic Cosine Function) 7-108
7.12.19 CPUT (Character Put Subroutine) 7-109
7.12.20 DATE (OS/78 Date Subroutine) 7-109
7.12.21 DIM (Single-Precision Positive Real Difference) 7-109
7.12.22 EXP (Single-Precision Exponential Function) 7-110
7.12.23 FLOAT (Integer-to-Floating-Point Conversion) 7-110
7.12.24 IABS (Integer Absolute Value Function) 7-110
7.12.25 IDIM (Integer Positive Difference Function) 7-110
7.12.26 IFIX (Single-Precision Floating-Point-to-Integer Function) 7-110
7.12.27 INT (Single-Precision Floating-Point-to-Integer) 7-110
7.12.28 ISIGN (Integer Transfer of Sign Function) 7-110
7.12.29 MAX0 (Single-Precision Maximum Value) 7-111
7.12.30 MAX1 (Single-Precision Maximum Value) 7-111
7.12.31 MIN0 (Single-Precision Minimum Value Function) 7-111
7.12.32 MIN1 (Single-Precision Minimum Value Function) 7-111
7.12.33 MOD (Integer A Modulo B Function) 7-111
7.12.34 SIGN (Single-Precision Transfer of Sign) 7-111
7.12.35 SIN (Single-Precision Sine Function) 7-111
7.12.36 SINH (Single-Precision Hyperbolic Sign) 7-112
7.12.37 SQRT (Single-Precision Square Root Function) 7-112
7.12.38 TAN (Single-Precision Tangent Function) 7-112
7.12.39 TANH (Single-Precision Hyperbolic Tangent) 7-112
7.12.40 TIME (Read Time of Day) 7-112
7.13 FORTRAN LANGUAGE SUMMARY 7-112

CHAPTER 8 BATCH 8-1

8.1 BATCH PROCESSING UNDER OS/78 8-1
8.2 BATCH COMMANDS 8-2
8.3 THE BATCH INPUT FILE 8-4
8.4 BATCH ERROR MESSAGES 8-6
8.5 RESTRICTIONS UNDER OS/78 BATCH 8-8

CHAPTER 9 OCTAL DEBUGGING TECHNIQUE (ODT) 9-1

9.1 ODT FEATURES 9-1
9.2 CALLING AND USING ODT 9-1
9.3 ODT COMMANDS 9-3
9.3.1 Special Characters 9-3
9.3.1.1 Slash (/) 9-3
9.3.1.2 RETURN -- Close Location 9-3
9.3.1.3 LINE FEED -- Close Location 9-3
9.3.1.4 Semicolon (;) -- Change Location, Close It and Open Next 9-4
9.3.1.5 nnnn+ and nnnn- -- Open Location Relative to Another 9-4
9.3.1.6 Circumflex (^) -- Close Location, Take Contents as Memory Reference Instruction and Open Effective Address 9-4
9.3.1.7 Underline (_) -- Close Location, Open Indirectly 9-4
9.3.2 Illegal Characters 9-5
9.3.3 Control Commands 9-5
9.3.3.1 ffnnnnG -- Transfer Control to Program at Location nnnn of Field ff 9-5
9.3.3.2 ffnnnnB -- Set Breakpoint at Location nnnn of Field ff 9-5

9.3.3.3 B -- Remove Breakpoint 9-5
9.3.3.4 A -- Open (AC) Location 9-6
9.3.3.5 L -- Open C(L) Location 9-6
9.3.3.6 C -- Continue from a Breakpoint 9-6
9.3.3.7 nnnnC -- Continue nnnn+1 Times from Breakpoint 9-6
9.3.3.8 D -- Open Data Field 9-7
9.3.3.9 F -- Open Current Field 9-7
9.3.3.10 M -- Open Search Mask 9-7
9.3.3.11 M LF -- Open Lower Search Limit 9-7
9.3.3.12 M LF LF -- Open Upper Search Limit 9-8
9.3.3.13 nnnnW -- Word Search 9-8
9.4 ERRORS 9-8
9.5 PROGRAMMING NOTES 9-8
9.6 ODT COMMAND SUMMARY 9-9

APPENDIX A ASCII CHARACTER SET A-1

APPENDIX B USEFUL MATHEMATICAL SUBROUTINES B-1

B.1 UNSIGNED INTEGER MULTIPLICATION SUBROUTINE B-1
B.2 UNSIGNED FRACTIONAL MULTIPLICATION SUBROUTINE B-2
B.3 UNSIGNED INTEGER DIVISION SUBROUTINE B-3
B.4 UNSIGNED FRACTIONAL DIVISION SUBROUTINE B-4

APPENDIX C USER SERVICE ROUTINE C-1

C.1 CALLING THE USR C-1
C.1.1 Standard USR Call C-1
C.1.2 Direct and Indirect Sequence C-3
C.2 USR FUNCTIONS C-4
C.2.1 FETCH Device Handler C-4
C.2.2 LOOKUP Permanent File C-6
C.2.3 ENTER Output C-7
C.2.4 The CLOSE Function C-8
C.2.5 Call Command Decoder C-10
C.2.6 CHAIN Function C-10
C.2.7 Signal User ERROR C-11
C.2.8 Lock USR in Memory C-12
C.2.9 Dismiss USR from Memory C-13
C.2.10 Ascertain Device Information C-13
C.2.11 RESET System Tables C-14

APPENDIX D THE COMMAND DECODER D-1

D.1 COMMAND DECODER CONVENTIONS D-1
D.2 COMMAND DECODER ERROR MESSAGES D-2
D.3 CALLING THE COMMAND DECODER D-3
D.4 COMMAND DECODER TABLES D-4
D.4.1 Output Files D-4
D.4.2 Input Files D-4
D.4.3 Command Decoder Option Table D-4
D.4.4 Example D-6
D.5 SPECIAL MODE OF THE COMMAND DECODER D-8
D.5.1 Calling the Command Decoder Special Mode D-8
D.5.2 Operation of the Command Decoder in Special Mode D-8

APPENDIX E OS/78 DEFAULT FILE NAME EXTENSIONS E-1

APPENDIX F USING DEVICE HANDLERS F-1

F.1 CALLING DEVICE HANDLERS F-1
F.2 OS/78 DEVICE HANDLERS F-3

| | | | |
|---|---|---|---|
| | F.2.1 | Line Printer | F-3 |
| | F.2.2 | File-Structured Devices | F-4 |
| | F.2.3 | Terminal Handlers | F-5 |
| | F.2.4 | Multiple Input Files | F-5 |
| APPENDIX G | | ERROR MESSAGE SUMMARY | G-1 |
| | G.1 | SYSTEM HALTS | G-1 |
| | G.2 | ERROR MESSAGES | G-2 |
| APPENDIX H | | OS/78 MULTIFUNCTION OPERATION | H-1 |
| | H.1 | SYMBIONT COMMANDS | H-1 |
| | H.2 | SPOOLER COMMANDS | H-2 |
| | H.3 | CUSP CODE CONVENTION | H-2 |
| | H.4 | WRITING A SYMBIONT | H-3 |
| APPENDIX I | | FULL WORD COMMAND SWITCHES | I-1 |
| APPENDIX J | | DECstation HARDWARE CONFIGURATION SUMMARY | J-1 |


## FIGURES


| | | | |
|---|---|---|---|
| FIGURE | 2-1 | Handling the Diskette | 2-1 |
| | 2-2 | Loading the Diskette | 2-2 |
| | 2-3 | Loading the RL01K Disk Pack | 2-4 |
| | 7-1 | Main Program and Subprogram for Calculating the Volume of a Regular Polyhedron | 7-8 |
| | 7-2 | Array Representation | 7-35 |
| | 7-3 | Array Storage | 7-36 |
| | 7-4 | Equivalence of Array Storage | 7-53 |
| | 7-5 | Legal and Illegal Common Extensions | 7-54 |
| | 7-6 | Nesting of DO Loops | 7-64 |
| | 7-7 | Control Transfers and Extended Range | 7-66 |


## TABLES


| | | | |
|---|---|---|---|
| TABLE | 2-1 | Terminal Operating Parameters | 2-6 |
| | 2-2 | OS/78 Permanent Logical Device Names | 2-10 |
| | 2-3 | OS/78 Extensions | 2-12 |
| | 2-4 | General-Purpose Dash Options | 2-17 |
| | 2-5 | Storage Capacity of OS/78 File-Structured Devices | 2-23 |
| | 2-6 | OS/78 Command Summary | 2-27 |
| | 3-1 | COMPARE Options | 3-6 |
| | 3-2 | COMPARE Error Messages | 3-9 |
| | 3-3 | COPY Options | 3-16 |
| | 3-4 | COPY Error Messages | 3-17 |
| | 3-5 | CREF Options | 3-19 |
| | 3-6 | CREF Error Messages | 3-20 |
| | 3-7 | DELETE Options | 3-24 |
| | 3-8 | DELETE Error Messages | 3-24 |
| | 3-9 | DIRECT Options | 3-25 |
| | 3-10 | DIRECT Error Messages | 3-27 |
| | 3-11 | DUPLICATE Options | 3-30 |
| | 3-12 | DUPLICATE Error Messages | 3-31 |
| | 3-13 | FORMAT Error Messages | 3-36 |
| | 3-14 | OS/78 Help Command Arguments | 3-39 |

| | | |
|---|---|---|
| 3-15 | HELP Error Messages | 3-39 |
| 3-16 | LIST Options | 3-41 |
| 3-17 | LIST Error Messages | 3-41 |
| 3-18 | MAP Options | 3-48 |
| 3-19 | MAP Error Messages | 3-49 |
| 3-20 | RENAME Options | 3-56 |
| 3-21 | Job Status Word | 3-60 |
| 3-22 | Summary of SET Command Forms | 3-62 |
| 3-23 | OS/78 Device Handlers | 3-67 |
| 3-24 | SET Error Messages | 3-68 |
| 3-25 | SUBMIT Options | 3-72 |
| 3-26 | TYPE Options | 3-76 |
| 3-27 | TYPE Error Messages | 3-77 |
| 4-1 | Editor Key Control Commands | 4-3 |
| 4-2 | Editor Input Commands | 4-6 |
| 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 | 4-11 |
| 4-8 | Aborting Editor String Search Commands | 4-18 |
| 4-9 | Nonfatal Editor Error Messages | 4-22 |
| 4-10 | Major Editor Error Codes | 4-22 |
| 4-11 | Editor Command and Special Characters | 4-23 |
| 5-1 | PAL8 Options | 5-9 |
| 5-2 | Use of Arithmetic Operators | 5-19 |
| 5-3 | PAL8 Diagnostic Error Codes | 5-40 |
| 7-1 | FORTRAN IV Compiler Run-Time Options | 7-5 |
| 7-2 | Compiler Error Messages | 7-5 |
| 7-3 | Loader Run-Time Options | 7-10 |
| 7-4 | Loader Error Messages | 7-11 |
| 7-5 | Run-Time System Options | 7-16 |
| 7-6 | Run-Time System Error Messages | 7-17 |
| 7-7 | FORTRAN Statement Categories | 7-20 |
| 7-8 | FORTRAN Special Characters | 7-21 |
| 7-9 | Field Summary | 7-25 |
| 7-10 | Classes of Symbolic Name | 7-27 |
| 7-11 | FORTRAN Data Types | 7-27 |
| 7-12 | Arithmetic Operators | 7-39 |
| 7-13 | Base/Exponent Combinations | 7-39 |
| 7-14 | Binary Operator Evaluation Hierarchy | 7-40 |
| 7-15 | Relational Operators | 7-41 |
| 7-16 | Logical Operators | 7-42 |
| 7-17 | Logical Operator Hierarchy | 7-43 |
| 7-18 | Arithmetic IF Transfers | 7-61 |
| 7-19 | Effect of Data Magnitude on G Format Conversions | 7-93 |
| 7-20 | Character Storage | 7-95 |
| 7-21 | Carriage Control Characters | 7-101 |
| 8-1 | BATCH Run-Time Options | 8-2 |
| 8-2 | BATCH Commands | 8-3 |
| 8-3 | BATCH Error Messages | 8-7 |
| 9-1 | ODT Command Summary | 9-9 |
| C-1 | Summary of USR Functions | C-2 |

3. Type the following commands in response to the monitor's period (.) prompting symbol:

```
.FORMAT RL01
.ZERO RL1A:/Y
.ZERO RL1B:
.ZERO RL1C:
.COPY RL1A:<RL0A:*.*
```

   The system will format the disk pack, zero each device and copy all the system files from unit A of Drive 0 to unit A of Drive 1. When the operation is complete, the monitor takes control as indicated by the period (.).

4. Remove both disk packs and label the disk pack taken from Drive 1 appropriately. This is the working system disk pack.

5. Store the master disk pack in a safe place.

6. Place the working system disk pack in Drive 0.

7. Press the system's BOOT button, and you are ready to use OS/78.

### 2.2.3 DECstations With RX02 and RL01 Disk Systems

The following steps describe how to make a backup copy of your master disk pack.

1. Place the master disk pack in RL01 Drive 0. Make sure that the drive is not write protected (WRITE PROT indicator lamp is extinguished).

2. Press the BOOT button.

3. Place a blank diskette in RX02 Drive 0.

4. Type the following command to initiate the backup operation:

```
.SUBMIT BUILDX
```

5. When the monitor again displays the period (.), remove the disk pack from its drive, and place a blank disk pack in the drive.

6. Type the following command to complete the backup operation:

```
.SUBMIT BACKRL
```

7. When the monitor again displays the period (.), the disk pack in Drive 0 now contains a complete copy of the contents of the OS/78 system software that resides on Device A (RL0A) of your master disk pack. If you want to make an additional backup copy of the system disk, type the following command:

```
.BO/RX
```

   then perform steps 5 through 7 again.

8. Store the master disk pack in a safe place.

## 2.3 DEMONSTRATION PROGRAMS

A series of programs are provided to demonstrate some of the capabilities of OS/78. These programs are run by a self-explanatory BATCH file called DEMO.BI. To initiate this demonstration, start the system and type the following command in response to the monitor's period (.):

```
.SUBMIT DEMO/H/T
```

The batch stream controlling the demonstration repeats it every three minutes until you type a CTRL/C.

Once you become familiar with the system, you may wish to remove the files comprising the demonstration to make room for other programs. To do so, use the DELETE command and type:

```
.DEL DEMO??.*
```

## 2.4 DEVICE AND FILE NAMES

### 2.4.1 Devices

The system recognizes each of its devices by one of the logical names listed in Table 2-2. You use these names in command strings to the system programs to specify the devices that you want to use. Although the system can recognize all the names, no one DECstation can have all listed physical devices. The SET HANDLER command, described in Chapter 3, allows you to specify which devices are available.

If you refer to a device that does not physically exist, the system will enter a loop condition and will not print any error message. You must restart the system by pressing the START or BOOT button to resume OS/78 operation.

Table 2-2
OS/78 Permanent Logical Device Names

| Name | Description |
|---|---|
| DSK | Default output device; usually same as SYS |
| SYS | The diskette or disk pack inserted in Drive 0 where the monitor and system programs reside. |
| RL0A | Unit A of RL01 Drive 0 |
| RL0B | Unit B of RL01 Drive 0 |
| RL0C | Unit C of RL01 Drive 0 |
| RL1A | Unit A of RL01 Drive 1 |
| RL1B | Unit B of RL01 Drive 1 |
| RL1C | Unit C of RL01 Drive 1 |
| RXA0 | The diskette inserted in Drive 0 |
| RXA1 | The diskette inserted in Drive 1 |
| RXA2 | The diskette inserted in Drive 2 |
| RXA3 | The diskette inserted in Drive 3 |
| TTY | Keyboard/screen |

(continued on next page)

2. If the command is recognized, execution will begin immediately. If you see an error in the command line, press CTRL/C quickly to terminate the action and return control to the monitor. Whether this step prevents execution of the incorrect command depends on the type of command and the file(s) involved.

   You can destroy important files if you incorrectly issue any of the following commands:

   COPY SAVE SQUISH DELETE ZERO

   In fact, the error may actually destroy part of the software. Therefore, be extremely careful when using these commands. Any specific cautions that should be exercised when using these commands are fully described in Chapter 3. Always be sure to make a back-up copy of the system and all other important files that you have created (see Section 2.2).

Section 2.5.3 describes how to correct any errors that have been made in the command line prior to executing the command.

### 2.5.3 Correcting Errors

Until the RETURN key is pressed, the system will not process the command line. Therefore, the command line is still available for correction. Always check the command lines before pressing the RETURN key since a line with errors may cause undesirable results.

To correct typing mistakes, use the DELETE key. This key erases the last character typed. Each use of the DELETE key causes one more character to be erased. The correct character or characters can then be typed.

A command line may be deleted completely before it is entered by typing CTRL/U (produced by holding down the CTRL key and pressing the U key). This echoes as ^U and returns control to the monitor without processing the current line. The monitor's period (.) indicates that the system is ready to accept a new command. You can then retype the command line. You can use the LINE FEED key to verify the contents of a command that is being entered. The system will display the portion of the command line that you have entered.

### 2.5.4 Using Input/Output Options

In addition to specifying output and input files in the command line, you can specify various options to perform select certain functions. Options are numbers, alphanumeric characters, or their full-word equivalents. Numbers used as options are generally contained in the command line with the equal sign (=) or square brackets ([]) construction. The alphanumeric option characters are set off from the I/O specifications by the slash (/) character for single character options and parentheses for a string of single characters. The usage of the slash, parentheses, equal sign, and square brackets is explained below.

You will find an explanation of the single character options in the command descriptions in Chapter 3. Appendix I lists the full-word equivalents to the single character options. A group of general-purpose options known as dash options that you can use in most command lines are described below.

2.5.4.1 **Equal Sign Construction** - An equal sign (=) followed by an octal number may occur only once in a command line and must be followed by a separator character (comma or left-angle bracket), other options, or a line terminator (RETURN or ESCape). For example,

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

will list the directory of the system device on the screen in three columns.

2.5.4.2 **Slash Construction** - A single alphanumeric character is preceded by a slash and can occur anywhere in the command line (even in the middle of a file name) although the usual position is at the end of the line. For example:

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

means the same as

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

The option /T instructs the monitor to assign the current date to the output file.

2.5.4.3 **Parentheses Construction** - When two or more letter options are used, you can group them together inside parentheses. This construction is valid anywhere in the command line. For example,

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

means the same as

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

2.5.4.4 **Square Bracket Construction** - The square bracket construction can only occur immediately after an output file name and consists of a left square bracket ([), a decimal number between 1 and 255, and a right square bracket (]). The square bracket construction allows you to optimize file storage by specifing an upper limit on the number of blocks required by your output file. For example:

```
.PAL BINARY[19],LISTIN[200]<SOURCE
```

The output files are a file named BINARY on device DSK: having a maximum length of 19 blocks, and a file named LISTIN (only six characters are significant) on the device DSK: with a maximum length of 200 blocks. The input file is SOURCE on device DSK.

2.5.4.5 **General-Purpose Dash Options** - These options provide a shorthand method for specifying devices and files with certain commands. The form is:

```
-ex
```

where:

-ex is one of the options specified in Table 2-4.

In the following example, typing

```
*PAL TEST-T
```

will assemble TEST.PA, store the resulting binary program in TEST.BN on DSK, and display the program listing on the terminal. The -T option saves you from typing the specification:

```
,TTY:<
```

Table 2-4
General-Purpose Dash Options

| Option | Meaning |
|---|---|
| -FT | Selects the FORTRAN IV Compiler if you do not use the default file extension .FT (used with the COMPILE and EXECUTE commands). |
| -L | Send output to LPT. For example, typing<br>.DIR-L<br>will list the director of the system device on the line printer (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 file name is the same as the file name that immediately preceded the -LS option in the command string. |
| -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). |
| -PA | Selects the PAL8 Assembler if you do not use the default file extension .PA (used with the COMPILE and EXECUTE commands). |
| -T | Send output to terminal. |

### 2.5.5 Remembering Previous Arguments

The system remembers the arguments that you use with the CREATE, LOAD, PAL, EDIT, and EXECUTE commands by storing them in a temporary file. Thereafter, when you use any of these commands without arguments, the system will use the arguments that it stored. When you again use these commands with arguments, the system replaces the currently stored arguments with the new arguments. The storage area for the EDIT command is separate from the others and its contents will not change unless you issue an EDIT command with arguments.

The following example illustrates this feature. If you entered the command:

```
.EXECUTE TEST1.PA
```

you could compile TEST.PA by issuing the command:

```
.COMPILE
```

NOTE

> The system does not remember command arguments typed if you restart the system with the START button or the BOOT command.

### 2.5.6 Using Wildcards

The wildcard construction is a feature that allows you write generalized file specifications when manipulating files. Wildcards allow a file name or the extension specified in a command to be replaced totally with an asterisk or partially with a question mark. This allows you to designate classes of file names or extensions. The wild characters are particularly useful when doing multiple file transfers.

You can use the wildcard construction in the file name specifications of the COPY, DELETE, DIRECT, LIST, RENAME, and TYPE commands.

The asterisk (*) is used as a wild field to designate the entire file name or extension. This is illustrated in the following examples:

| | |
|---|---|
| TEST1.* | All files with the name TEST1 and any extension. |
| *.BN | All files with a BN extension and any file name. |
| *.* | All files (except when used with the DELETE command). |

The question mark (?) is used as a wildcard character to designate part of the file name or extension. A question mark is used for each unspecified character that is to be matched. For example, PR?? matches all files beginning with PR that are two to four characters long. Other examples are as follows:

| | |
|---|---|
| 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 from three to five characters long beginning with TES. |
| ??.?? | All files with file names of two characters or less. |

The asterisk and the question mark can be specified together in the same command line:

| | |
|---|---|
| ???.* | All files with file names of three characters or less. |

## 2.8 FILE-STRUCTURED DEVICES

A file-structured device is a random-access mass storage device that is logically divided into a number of blocks. This kind of device can read and write any desired block. A diskette and disk pack are file-structured devices, but a terminal is not.

All OS/78 file-structured devices are logically divided into 256-word blocks. Hence, blocks of 256 words are considered the standard size for OS/78. A file consists of one or more sequential (consecutively numbered) blocks. A minimum of one block per file is required, although a single file could occupy all of the blocks on a device.

By convention, OS/78 block numbers are specified in octal radix, while file lengths (number of blocks per file) are specified in decimal radix.

There are two types of file-structured devices, system devices and non-system (or files only) devices. A system device contains a copy of the OS/78 Monitor (also called the system head) on its medium and the OS/78 system programs; a non-system device does not.

The system device is also the disk drive that is accessed when you press the DECstation's START or BOOT button or use the BOOT command. If your system has an RX01 or RX02 diskette drive, the system device is Drive 0 (RXA0). If your system has RL01 cartridge disk drives only, the system device is device A of RL01 Drive 0 (RL0A).

Table 2-5 lists the OS/78 file-structured devices and their storage capacities.

Table 2-5
Storage Capacity of OS/78 File-Structured Devices

| Device | Type | Size (in blocks) |
|---|---|---|
| RLxA | System | 4018 |
| | Non-system | 4074 |
| RLxB | Non-system | 4074 |
| RLxC | Non-system | 2018 |
| RX01 | System | 431 |
| | Non-system | 487 |
| RX02 | System | 932 |
| | Non-system | 981 |
| VXA0* | Non-system | 128 |

* DECstation 88/97 systems only

### 2.8.1 Using RX01 And RX02 Diskettes

The RX01 and RX02 diskettes are physically identical. The RX01 is a single-density diskette and the RX02 is a double-density diskette. This means that an RX02 diskette has approximately twice the storage capacity of an RX01 diskette (see Table 2-4).

The RX01 diskette is supplied already formatted for single-density operation. You must, however, create a directory area on it with the ZERO command before copying files to it. If you are creating backup copies with the DUPLICATE command, the ZERO command is not necessary.

The RX02 diskette is a single-density RX01 diskette that you reformat for dual-density operation with the DUPLICATE command's /D option. You must, however, create a directory area on it with the ZERO command before copying files to it. If you are creating backup copies with the DUPLICATE command, the ZERO command is not necessary.

You should observe the following rules when using diskettes:

- An RX01 disk drive can use single-density diskettes only.
- An RX02 disk drive can use either single-density or double-density diskettes. The system automatically determines the density of the diskette.
- A system diskette that contains an old version of OS/78 can run on an RX01 drive only. You can use it on an RX02 drive if you want to use it for file storage only.

### 2.8.2 Using the RL01 Cartridge Disk Pack

The RL01 cartridge disk pack is a high-density mass storage device that has a storage capacity that is approximately ten times that of the RX02 double-density diskette (see Table 2-4).

Each disk pack consists of three logical OS/78 devices: device A, device B, and device C. Device A and B are the same size. Transfers between these devices are faster than transfers between either one of them and device C. This is because the device C is smaller than the other two and consequently occupies less of each track on the disk than the others. This increases the search time required for device C blocks.

You must format and zero each new blank disk pack with the FORMAT and ZERO commands before using it with OS/78. If you do not do this, OS/78 cannot successfully use the disk pack.

Each disk pack incorporates a bad block mapping scheme that allows OS/78 software to ignore disk blocks that become defective. Whenever a bad block error occurs, you can use the FORMAT command to make the block unavailable to OS/78. This allows you to prolong the life of your disk packs.

### 2.8.3 Using the VXA0 Extended-Memory Device

The VXA0 device enables you to use the extended memory above 32K provided in DECstation 88/97 systems as though it were a separate file-structured device. The VXA0 device provides high speed I/O performance similar to that of a fixed-head disk type of storage device. You refer to it in the same way as the other OS/78 devices. For example, this command

```
.COPY VXA0:SAMPLE<RXA0:SAMPLE
```

copies a program called SAMPLE into the area of memory above 32K. The command

```
.DIR VXA0:
```

produces a directory listing of the device.

## 2.9 FILES

A file is the fundamental storage unit of the OS/78 system. It is a uniquely named collection of data that is treated as a unit. The format of this data is unimportant; OS/78 can manipulate several standard formats, including ASCII files, binary files, and memory-image files. The important consideration is that the data forms a single and uniquely identifiable unit within the system.

### 2.9.1 File Directories

To maintain records of the files on a device, blocks 1 through 6 are reserved for the file directory of that device. Thus, file structured devices are also called directory devices. Entries in this directory inform the system of the name, size, anad location of each file, including all empty files and the tentative output file, if one exists. Six blocks are always allocated, though all are not necessarily active at any given time. Block zero is unused, except on the system device, in which case it contains the system bootstrap, a program that reads in the OS/78 Monitor and the system device handler (SYS).

### 2.9.2 File Types

Three types of files exist in the OS/78 system:

1. Empty File - An empty file is a contiguous area of unused blocks. Empty files are created when permanent files are deleted or when the ZERO command is used.

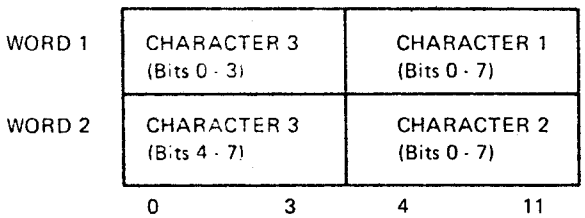
2. Tentative File - A tentative file is a file that is open to accept output and has not yet been closed. Only one tentative file can be open on any single device at one time. Tentative files never appear in directory listings.

3. Permanent File - A permanent file is a file that has been given a fixed size and is no longer expandable. A tentative file becomes permanent when it is closed.

To further understand file types, consider what occurs when a file is created. Normally, in creating a tentative file, the system first locates the largest empty file available and creates a tentative file in that space. That establishes the maximum space into which the file can expand. The user program then writes data into the tentative file. At the end of the data, the program tells the system to close the tentative file, making it a permanent file. When the file is closed, whatever space remains at the end of the file is available to contain a new file.

### 2.9.3 ASCII File Format

ASCII files are packed as three 8-bit characters into two words as follows:




## 2.10 OS/78 COMMAND SUMMARY

Table 2-6 is a summary of OS/78 commands. For each command, the table shows the command and its abbreviation printed in red ink, the format, and a brief functional description.

Table 2-6
OS/78 Command Summary

| Command | Format Example | Function |
|---|---|---|
| ASSIGN | AS permdev user-name | Associates a new user-defined device name with the permanent logical device name. |
| BASIC | BAS | Requests execution of the BASIC language editor to allow for program creation and modification. |
| BOOT | BO /dv | Initializes the system residing on the specified system device RL or RX. |
| CANCEL | CA | Terminates the symbiont task (see Appendix H) that is currently running and returns OS/78 to 16K, single task operation (DECstation 78 systems only). |
| COMPARE | COMPA dev:output.ex<<br>dev:file1.ex,dev:file2.ex | Compares two source files line by line and displays all their differences. |
| COMPILE | COM file.ex<file1.ex,.../options | Produces binary files and/or listings for specified program files. This command will chain to PAL8, BASIC, or FORTRAN IV depending on the file name extension. Under FORTRAN IV, only one input file can be specified. |
| COPY | COPY dev:file.ex<dev:file.ex,... | Transfers files from one OS/78 device to another. Up to five input files can be specified. |
| CREATE | CREA file.ex | Opens a new file for editing. |
| CREF | CREF file.ex | Assembles and produces a cross-reference listing from the source file. Defaults to LPT, thus the -T option must be used to output to terminal. |

(continued on next page)

Table 2-6 (Cont.)
OS/78 Command Summary

| Command | Format Example | Function |
|---|---|---|
| DATE | DA dd-mmm-yy | Enters the date into the system. If no argument is given, print current date on terminal or NONE if no date was specified. |
| DEASSIGN | DE | Eliminates all previous user-defined names. |
| DELETE | DEL dev:file.ex,.../options | Deletes one or more files. Up to five input files can be specified, separated by commas. All files must reside on the same device. |
| DIRECT | DIR dev:/options | Produces a listing of an OS/78 device directory. |
| DUPLICATE | DUP outdev:<indev:/options | Reproduces the contents of the input diskette onto the output diskette. |
| EDIT | ED file<br>or<br>ED file<file | Opens an already existing file for editing. Using the input/output construction allows a copy of the original file to be retained, while naming a new output file. |
| EXECUTE | EX file.ex,file.ex/options<br>or<br>EX file.ex<file.ex | Produces binary files and/or listings for the specified program files, loads the binary file, and executes the program. This command will chain to PAL8, BASIC or FORTRAN IV depending on the file name extension. Under FORTRAN IV, only one file can be specified. |
| FORMAT | FORMAT RL01 /options | Formats RL01 disk pack for OS/78 use. |

(continued on next page)

Table 2-6 (Cont.)
OS/78 Command Summary

| Command | Format Example | Function |
|---|---|---|
| GET | GE dev:file.ex | Loads memory image files (.SV format) into memory from a device. |
| HELP | HE command | Displays instructional information on the specified OS/78 command. |
| LIST | LI dev:file.ex,... | Lists the contents of the specified files on LPT. |
| LOAD | LO dev:file.ex | Runs one of the OS/78 loaders for either PAL8, BASIC or FORTRAN IV depending upon the extension of the filename. |
| MAP | MAP dev:file.ex<dev:file.ex | Produces a memory map of the specified input file. |
| MEMORY | MEM n | Sets the number of memory fields available. |
| | MEM | Displays both the number of fields actually being used by OS/78 and the number of fields available in the hardware. |
| ODT | OD | Loads and starts the ODT (Octal Debugging Technique) system. |
| PAL | PAL dev:binfile,dev:listfile, dev:creffile<dev:infile,... | Runs the PAL8 assembler and assembles the specified source file. |
| QUEUE | Q filename-list/options | Line printer spooler symbiont command (see Appendix H) to queue files to be printed (DECstation 78 system only). |

(continued on next page)

Table 2-6 (Cont.)
OS/78 Command Summary

| Command | Format Example | Function |
|---|---|---|
| R | R file.ex | Loads memory image files (except OS/78 system programs) from the system device (making it equivalent to a RUN SYS file.ex command except that the system scratch area is not affected) and starts it at the starting address. If .ex is omitted, .SV is assumed. |
| RENAME | REN dev:new.ex<dev:old.ex/options | Changes the name of the file from the input name to the output name. |
| REQUEST | REQ symbiont-name | Starts symbiont task (DECstation 78 systems only. See Appendix H). |
| RUN | RU dev:file.ex | Loads a memory image file (except OS/78 system programs), moves its Core Control Block to the system scratch area, and starts the program at its starting address. This command is equivalent to a GET and START command, and must be used to run a program that does not reside on SYS:. |
| SAVE | SA dev:file.ex options | Saves an image of the program currently in memory on the device specified. If .ex is omitted, .SV is assumed. |
| SET | | Changes device handler characteristics.<br>ter: = TTY:, SLUx, VLUx<br>any: = any device<br>NO = optional modifier to inhibit effect |
| | SET any: [NO] READONLY | Makes device read-only device. |
| | SET LPT: [NO] LC | Prints lower-case characters. |

(continued on next page)

Table 2-6 (Cont.)
OS/78 Command Summary

| Command | Format Example | Function |
|---|---|---|
| SET (Cont.) | SET LPT:WIDTH n | Sets column width to n. |
| | SET LPT:COL n | Sets number of columns displayed by DIRECT (ter: also). |
| | SET ter: [NO] ARROW | Sets terminal so that CTRL characters map as ^char rather than the actual code. |
| | SET ter:COL n | Sets number of columns displayed by DIRECT (LPT: also). |
| | SET ter: [NO] ECHO | Enables character echoing on input. |
| | SET TTY: [NO] ESC | Sets console terminal so that ESC code maps to a $ rather than 033(8). |
| | SET ter:HEIGHT m | Sets screen height to m. |
| | SET SYS: [NO] INIT | Executes command in INIT.CM when system is bootstrapped. |
| | SET SYS: [NO] INIT cmd | Executes command (cmd) when system is bootstrapped. |
| | SET ter: [NO] PAGE | Enables CTRL/Q and CTRL/S. |
| | SET ter: [NO] PAUSE n | Enables pause of n seconds during output. |
| | SET ter: [NO] SCOPE | Deletes character from screen rather than printing a \ when DELETE key is used. |
| | SET ter:WIDTH n | Sets column width to n. |
| | SET HANDLER old<new | Replaces handler currently in system (old) with new handler (new) called: handlername.HN. |

(continued on next page)

Table 2-6 (Cont.)
OS/78 Command Summary

| Command | Format Example | Function |
|---|---|---|
| SET (Cont.) | SET HANDLER/L | Lists names of handlers currently in system. |
| SQUISH | SQ dev:<dev: | Eliminates all embedded empty files on the device. |
| START | ST ffnnnn | Starts the program currently in memory at location nnnn in field ff. |
| | ST | Starts the program currently in memory at the starting address specified in the Core Control Block. |
| SUBMIT | SU dev:file.ex<dev:file.ex | Runs the BATCH program where the output is the optional spooling file and the input is the BATCH input file. |
| TERMINATE | TER | Causes OS/78 to terminate its operation and to run a user-created stand-alone program called TERMIN.SV. |
| TYPE | TY dev:file.ex,... | Displays the specified files on the terminal. |
| UA | UA (argument) | Remembers the argument where the argument must be a legal OS/78 command. UA with no argument executes the last remembered argument. |
| UB | UB (argument) | Similar to UA. |
| UC | UC (argument) | Similar to UA. |
| ZERO | ZERO dev:/options | Zeros the specified device directory, deleting any files that may exist or the device. |

The above example shows a sequence of commands in creating and editing of file MATCH.PA. The EDIT command is given without any argument since it remembers the file MATCH.PA specified with the CREATE command.

If the date has changed and a command line is typed containing the EDIT command without any arguments, the error message BAD RECOLLECTION will be displayed.

Example 2:

```
.EDIT A2.FT<A1.FT
*R
 .
 .
 .
*E
 .
 .
 .
.EDIT
```

Since the second EDIT command had no arguments, the previous argument up to the left-angle bracket (<) is recalled. Thus, the second EDIT command is equivalent to:

```
.EDIT A2.FT
```

This command executes the CCL.SV and EDIT.SV programs.

**EXECUTE**

## 3.16 EXECUTE COMMAND

The EXECUTE command performs the following functions:

1. assemble or compile, link, load and execute a BASIC, FORTRAN, or PAL8 source program
2. link, load and execute an already assembled or compiled program
3. execute an already linked and loaded program

Format:

```
EXECUTE indev:file.ex
```

To use the EXECUTE command with a source file, the input device must be specified. Then that source file is assembled or compiled, linked and loaded into memory and executed. The file extension used determines the compiler or assembler called.

If the file name extension is not a standard extension, specify the correct assembler or compiler by using one of general-purpose dash options in the -ex portion of the command line (see Table 2-4).

Example:

```
.EXE RXA1:NABS.IN-PA
```

If the EXECUTE command is used with an already assembled or compiled program, the input device must be specified. When the EXECUTE command is processed, the binary file is loaded into memory and executed. If the file name extension is not specified in the command line but is specified in the directory, the correct extension is automatically added.

The EXECUTE command can remember arguments from a previous COMPILE, LOAD, PAL, or EXECUTE command. When you restart the system with the START or BOOT button or with the BOOT command, the arguments are lost.

NOTE

For options and expanded command strings acceptable to the EXECUTE command, refer to the appropriate language/loader section.

This command executes CCL.SV and one or more of the following programs: PAL8.SV and ABSLDR.SV, F4.SV, FRTS.SV and LOAD.SV, or BCOMP.SV and BLOAD.SV.

**FORMAT**

## 3.17 FORMAT COMMAND

The FORMAT command verifies and formats RL01 cartridge disk packs. You must use FORMAT on each new disk pack before you can store any OS/78 files on it. You should also use FORMAT to reformat any disk pack that has blocks that the system cannot access to make them invisible to the system. (After using FORMAT, you must also use the ZERO command to create a new directory.)

NOTE

FORMAT does not alter the contents of a disk pack that has no bad blocks. If there are bad blocks, the data residing in the good blocks that follow a bad block will become unusable after reformatting, so be sure to make copies of the files that you want to save before using FORMAT.

Format:

```
FORMAT RL01/n/P
```

where:

/n specifies the drive number, 0 or 1. The default value is 1.

/P causes the system to pause before and after the operation so you can change disk packs. If you select drive 0, the pause is automatic. Type a CTRL/C to return to the monitor. If you are running under BATCH, FORMAT does not pause after it completes the operation.

Example:

```
.FORMAT RL01
```

This command formats the disk pack on drive 1.

The FORMAT command executes the FORMAT.SV program.

NOTE

FORMAT.SV is available on RL01 and RX02 media only.

Table 3-13
FORMAT Error Messages

| Message | Meaning |
|---|---|
| BAD COMMAND LINE | The command line contains a syntax error. Retype the line correctly. |
| BAD DISK | Disk pack cannot be reformatted. There are more than 63 bad blocks or the system area (blocks 0 - 70 octal) contains bad blocks. |
| CAUTION - THIS COMMAND DESTROYS CONTENTS OF SYSTEM DISK. CHANGE DISKS BEFORE PROCEEDING OR CTRL/C. | Drive 0 was selected for formatting. Be sure to replace your system disk pack with the one that you want to format. |
| READY. STRIKE CARRIAGE RETURN TO CONTINUE | The /P option was selected. Mount the disk pack to be formatted and type a carriage return to begin formatting. |

For example, the following statement stores the value 37 in the variable N.

```
10 LET N=37
```

If N already contains a value, the new value replaces it.

Once you have assigned a value to a variable, BASIC will use the value in any expression in which the variable appears, for example:

```
10 LET B=N*2
```

This statement evaluates the expression N*2 and stores the result in the variable B.

You may instruct BASIC to change the value of a variable any number of times during the execution of a program. BASIC always uses the most recently assigned value of a variable when performing calculations. The following section describes numeric variables, string variables, and subscripted variables.

6.3.2.1 **Numeric Variables** - A numeric variable name consists of a letter or a letter followed by a digit, for example:

| Acceptable Variables | Unacceptable Variables |
|---|---|
| M | 2C (A variable cannot begin with a digit.) |
| R2 | AB (A variable may contain only one letter.) |

Unless you specify otherwise, BASIC automatically sets all variables to zero before executing a program. However, if you wish to assign zero, it is good programming practice to do the initializing yourself at the beginning of the program. You can do this with a series of LET statements or by using READ and DATA statements. For example, this statement

```
10 LET A=0\B=0\C=5
```

assigns 0 to A and B and 5 to C.

6.3.2.2 **String Variables** - A string variable name consists of a letter -- or a letter and a digit -- followed by a dollar sign. A$ and A2$ are both legitimate string variable names; 2A$ and AA$ are not.

You may assign up to 18 characters to a string variable. To assign more you must first dimension the variable with a DIM statement. (See Section 6.7.)

6.3.2.3 **Subscripted Variables** - A subscripted variable consists of a string or numeric variable followed by a subscript in parentheses. A subscript may be a number, a numeric variable, or an expression or any two such elements separated by a comma. The following are all legal subscripted variables.

```
A(2)

A(K)

M(3,4)

M$(I,J)

G(K-1)

F(3.4)
```

If the subscript is not a whole number, BASIC uses only the whole number part. In the example above F(3.4) is the same as F(3). BASIC permits a subscript value of zero.

One subscript after a numeric variable serves as a pointer to a location in a one-dimensional array. For example, this subscripted numeric variable

```
X(3)
```

indicates the fourth position in the array.

```
X(0), X(1), X(2), X(3), X(4), X(5)
```

Note that all subscripted variables in an array share the same variable name.

Two subscripts appended to a numeric variable (and separated by commas) indicate a row and column number in two-dimensional array. This variable

```
A(3,5)
```

points to row 3 column 5.

One subscript appended to a string variable indicates the length of the string. Two subscripts indicate its position in a list and its length. For example, this variable

```
R$(35)
```

will hold a string constant 35 characters long. And this variable

```
R$(5,35)
```

indicates that the sixth position in a list (beginning with 0) holds a string 35 characters long.

You cannot create a two-dimensional array of string variables in a BASIC program.

To specify the length of a string, use a DIM statement in the form:

```
(line number) DIM X$(n)
```

where:

X$ is a string variable

n is the length of the string. A string may contain no more than 132 characters. All strings that exceed 18 characters in length must be dimensioned with a DIM statement.

To create a one-dimensional array of subscripted string variables, use the format

```
(line number) DIM X$(n,m)
```

where:

X$ is a string variable name

n specifies the number of strings in the array. (The number of strings is n + 1.)

m is the length of each string -- up to 132 characters

For example, this DIM statement describes one string 12 characters long.

```
10 DIM C$(12)
```

This statement describes 4 strings, each 20 characters long.

```
10 DIM D$(3,20)
```

This program will fill variables from a DATA list.

```
10 DIM D$(3,20)
20 FOR Y=0 TO 3
30 READ D$(Y)
40 NEXT Y
50 FOR Z=0 TO 3
60 PRINT D$(Z)
70 NEXT Z
80 DATA "ZERO","ONE","TWO","THREE"
```

Keep in mind the following features and rules concerning the DIM statement.

- Arrays are limited in size only by the amount of available memory -- that is, space not used by the monitor or the program statements.
- Subscripts n and m must be integer numbers. They may not be variables.
- A variable may not appear in a program with subscripts higher than the ones you have described in the DIM statement.
- BASIC assumes a string length of 18 characters or less unless you define the string variable with a DIM statement. If you wish to assign a string that is more than 18 characters long, you must DIMension the string variable.

- BASIC will not accept two-dimensional string variables.
- BASIC assigns a subscript of 0 to the first element in every array. Therefore, the number of elements in a one-dimensional array is n + 1, and the number of elements in a two-dimensional array is (n + 1)*(m + 1).
- You may define more than one array with a single DIM statement. For example, this statement dimensions both the one-dimensional array A and the two-dimensional array B.

```
10 DIM A(20), B(4,7)
```

## 6.8 THE COMMENT STATEMENT -- REMARK AND EXCLAMATION POINT

The REM statement or an exclamation point lets you document your source program with notes and comments, for example:

```
10 REM SUBROUTINE SWAPS VALUES A AND B
```

or

```
10 ! SUBROUTINE SWAPS A AND B
```

Use the exclamation point to add a comment to an executable line, for example:

```
100 A(I,J) = B(I,J) !  COPY B TO A
```

## 6.9 INPUT AND OUTPUT

BASIC provides you with three ways to supply a program with data.

- The INPUT statement lets you type in data at runtine.
- The READ, DATA, and RESTORE statements let you include data in a source program.
- BASIC file and record statements make it possible for you to store data in files and retrieve it under program control.

This section describes only the first two methods. See Sections 6.12 and 6.13 for information on file and record input and output.

The BASIC PRINT statement causes BASIC to display strings and the results of computations on the terminal.

### 6.9.1 INPUT

The INPUT statement allows you to enter data from the terminal during program execution.

Format:

```
(line number) INPUT x1, x2,...,xn
```

where:

x1 through xn represent numeric variables or string variables.

6.11.12.1 **The TAB Function** - he TAB function enables you to position characters on a line.

Format:

```
PRINT TAB(n)
```

where:

n is the number of a column.

For example, to print an upper-case "X" in column 10 of your screen, type

```
5 PRINT TAB(10);"X"
```

6.11.12.2 **The PNT Function** - The PNT function accepts an ASCII code number in decimal and returns an ASCII character. PNT is useful for transmitting non-printing characters and control codes to the terminal.

PNT works only in combination with a PRINT (or PRINT#) statement.

Format:

```
PRINT PNT(n)
```

where:

n is an ASCII code number expressed in decimal.

Use the PNT function to transmit control codes to the video terminal. For example, code 07 decimal (which you can specify by typing CTRL G) rings the bell on the terminal. To call for this action in a BASIC program, use the PNT function in a PRINT statement, specifying code 07 as the argument.

```
10 PRINT PNT(07)
20 END
```

This complete BASIC program rings the terminal bell.

The following control codes move the cursor on the screen.

PNT(08) moves cursor one space to the left

PNT(09) moves cursor to next tab stop

PNT(10) moves cursor down one line and scrolls if necessary

PNT(13) moves cursor to left margin of current line

To call for an escape sequence in a BASIC program, enter the ESCape code (27 decimal) with the PNT function, followed by the letter code for the sequence you wish to generate.

Format:

```
PRINT PNT(27);"X"
```

where:

27 is the code for ESCape

X is any capital letter that generates an escape sequence.

For example, the following statement transmits the sequence ESC H to the terminal. ESC H moves the cursor to the upper left corner of the screen.

```
10 PRINT PNT(27);"H"
```

For a list of additional ESCape Sequences that you can call for with the PNT function, see Section 6.14.

For a complete discussion of control codes, including ESCape Sequences, see the user's manual for your terminal: the DECscope User's Manual (EK-VT5X-OP-001) or the VT100 User Guide (EK-VT100-UG-PRE).

## 6.12 FILE STATEMENTS

BASIC file statements -- which are distinguished from other BASIC statements by the number sign (#) -- let you store data on peripheral devices for later use in any BASIC program. They include

- FILE# describes the file, assigns it a channel number from 1 to 5 (the number of files which BASIC can handle at one time), and opens it.
- PRINT# writes data on the file.
- PRINT# USING defines format of string numbers in a file.
- INPUT# reads data from the file.
- RESTORE# resets the pointer to the beginning of the file.
- CLOSE# closes the file and removes the channel number.
- IF END# tests for end-of-file.
- IF OPEN# determines if the file you specify is currently open.

In most operations, you open a file (FILE#) for output or input, write on it (PRINT#) or read from it (INPUT#), and close it (CLOSE#). You can open five files at a time -- excluding the terminal, which is always open and available for use -- on the system device plus 3 non-system devices. The ability to open and close files under program control gives you access to an unlimited number of files. That is, when you close a file, you may re-assign its channel number to a newly opened file.

BASIC treats files in the same way it treats terminal input and output. The INPUT statement causes BASIC to read a value that you enter on the terminal and assign it to a variable; the INPUT#

statement causes it to read a value from a file. The PRINT statement instructs BASIC to display data on the terminal; the PRINT# statement tells it to write data in a file.

will display:

```
1
2
3
4
5
6
```

6.12.2.3 Formatting Output to a File -- PRINT# USING - The PRINT# USING statement lets you define the format of string numbers that you want to write to a file.

The difference between PRINT# USING and PRINT USING is the same as the difference between PRINT# and PRINT: instead of causing BASIC to display formatted data on the terminal, PRINT# USING causes BASIC to place it in a file.

For a complete description of the procedure you follow to define the format of a string number, see the PRINT USING statement (Section 6.9.4).

Format:

(line number) PRINT #n: USING format string, list

where:

| | |
|---|---|
| n | is the channel number of a file opened to receive string output |
| format string | is a string or string variable describing the format of the string numbers you want to write to the file |
| list | is a series of string numbers, string variables, or string functions separated by commas or semicolons. |

For example, the following statement defines the format for any string number that you assign to the string variable A$ and places the formatted value in file #1.

```
20 PRINT #1: USING "$****.##",A$
```

6.12.2.4 Resetting a File -- RESTORE# - The RESTORE# statement resets the file back to the beginning so that the next INPUT# statement will cause BASIC to read the first item in the series.

Format:

(line number) RESTORE# n

where:

| | |
|---|---|
| n | is the channel number of the file to be reset or a variable representing the channel number. |

If n is 0, BASIC resets the DATA list to the beginning.

In the following program, RXA1:FILB.LM is a numeric input file containing the numbers 1 through 9. These instructions

```
100 FILEN#3:"RXA1:FILB.LM"
110 FOR I=1 TO 3
120 INPUT #3:Z
130 PRINT Z
140 NEXT I
150 RESTORE#3
160 INPUT#3 Z
170 PRINT Z
999 END
```

will display:

```
1
2
3
1
```

6.12.2.5 Checking for Open Files -- IF OPEN# - This statement lets you determine if a FILE statement has successfully opened a file or if a CLOSE statement has closed all files at the conclusion of a program. You can also use the IF OPEN statement to determine if a file exists.

Format:

(line number) IF OPEN #n THEN ln

where:

n — is the number of the file you wish to test; ln is the line number of the statement to which the program will jump if the file is open.

For example, the following program

```
10 DIM J$ (15)
20 FILEV #2: "CHECKF.FI"
30 PRINT #2: "CHECK FILE"
40 CLOSE #2
50 IF OPEN #2 THEN 80
60 PRINT "FILE NO.2 CLOSED"
70 GO TO 99
80 PRINT "FILE NO.2 OPEN"
99 END
```

will display

```
FILE NO.2 CLOSED
```

Without the CLOSE statement in line 40, BASIC would execute line 80, displaying the information that the file was open.

This statement provides a way to protect a data file in a long program, where you might accidentally omit the CLOSE statement.

You can use the IF OPEN statement to determine if a file that is called for actually exists. For example in the following sequence of statements:

```
10 FILE #1 "PROG.PA"
20 IF OPEN #1 THEN 50
30 PRINT "FILE NOT THERE"
40 STOP
50 ....
```

line 10 opens file PROG.PA. If PROG.PA does not exist, the program proceeds from the IF OPEN statement to line 30 and displays the message. If PROG.PA is open, and therefore exists, the program jumps to line 50.

6.12.2.6 **Checking for End-of-File -- the IF END# Statement** - The IF END# statement lets you detect the end of a string file.

Format:

(line number) IF END#n THEN m

where:

| | |
|---|---|
| n | is the channel number of the file in question or a variable representing the number. |
| m | is the number of the line in the program which BASIC will jump to if it has reached the end of the file. |

The IF END# statement works only on string files and must immediately follow the PRINT# or INPUT# statement for that file.

When you use the IF END# statement, you are asking BASIC to check if its last attempt to execute a PRINT# or INPUT# statement was successful. If it was unsuccessful -- if nothing was written or read -- BASIC jumps to line m.

| Statement | Function |
|---|---|
| PRINT# | Writes data to a file<br>Example: 180 PRINT#1:J |
| PRINT USING | Allows formatting of string numbers displayed on terminal<br>Example: 10 PRINT USING "##","3.7" |
| PRINT# USING | Allows formatting of string numbers written to a file<br>Example: PRINT #1: USING "$****.##",A$ |
| PUT# | Writes file records<br>Example: PUT#1:O,A$,B$,C$ |
| RANDOMIZE | Causes the RND function to produce a different set of numbers each time the program is run<br>Example: 10 RANDOMIZE |
| READ | Sets variables equal to the values in DATA statements<br>Example: 50 READ A$,B |
| REM | Inserts comments into the program<br>Example: 30 REM COMPUTE EARNINGS |
| RESTORE | Sets program READ statements back to the beginning of the DATA list<br>Example: 85 RESTORE |
| RESTORE# | Resets a file pointer back to the beginning of that file<br>Example: 130 RESTORE#3 |
| RETURN | Returns control from a subroutine (used with GOSUB)<br>Example: 115 RETURN |
| STOP | Terminates program execution<br>Example: 40 STOP |

## 6.22 SUMMARY OF BASIC FUNCTIONS

| Command | Function |
|---|---|
| ABS(X) | Returns the absolute value of an expression<br>Example: 10 LET X=ABS(-66)<br>will assign X a value of 66 |

| Command | Function |
|---|---|
| AND(A,B) | Performs a bitwise AND on two numbers entered in decimal |
| ASC(X$) | Converts a one character string to its code number<br><br>Example: 20 PRINT ASC("B") will display 2 |
| ATN(X) | Calculates the angle (in radians) whose tangent is given as the argument<br><br>Example: 30 LET X=ATN(.57735)<br><br>will assign X a value of 0.523598 |
| CAP$(X$) | Converts lower case to upper case |
| CCL(X$) | Causes BASIC to exit from currently running program, chain to CCL.SV, and pass the argument assigned to the function<br><br>Example: 100 X=CCL("SUBMIT BJOB.BI/T")<br><br>terminates BASK program and submits BATCH job BJOB.BI |
| CHR$(X) | Converts a code number to its equivalent character<br><br>Example: 40 PRINT CHR$(1) will display A |
| COL(N) | Returns the column number where the print head is positioned |
| COS(X) | Returns the cosine of an angle specified in radians<br><br>Example: 50 LET Y=COS(45*3.14159)/180<br><br>will assign Y a value of 0.707108 |
| CUR$(V,H) | Outputs a VT52 escape sequence |
| DAT$(X) | Returns the current system date<br><br>Example: 60 PRINT DAT$(X)<br><br>will display the system date, such as 07/20/77 |
| EXP(X) | Calculates the value of e raised to a power, where e is equal to 2.71828<br><br>Example: 30 IF Y>EXP(1.5) GOTO 70<br><br>will go to line 70 if Y is greater than 4.48169 |
| FIX$(X$) | Accepts a string number as an argument and returns the integer part<br><br>Example: 10 PRINT FIX$ ("4"/"3")<br><br>prints the value 1 |

| Command | Function |
| --- | --- |
| INT(X) | Returns the value of the nearest integer not greater than the argument<br><br>Example: 60 LET X=INT(34.67)<br><br>will assign X the value 34 |

# INDEX


0, FORTRAN carriage control character, 7-101
1, FORTRAN carriage control character, 7-101
/2, QUEUE command option, H-2
/8, LOAD command option, 3-43
/9, LOAD command option, 3-43
/=n, DIRECT command option, 3-25

A,
  Editor append command, 4-6
  field descriptor, 7-95
  ODT command, 9-6
ABS BASIC function, 6-48
ABSLDR.SV, 3-11, 3-34, 3-44
Absolute binary files, 3-42
Absolute value function, 6-48
Addition,
  BASIC operator (+), 6-17
  FORTRAN operator (+), 7-39
  PAL8 operator (+), 5-18
Addressing indirect and page zero, 5-29
AINT FORTRAN function, 7-106
ALOG FORTRAN function, 7-106
ALOG10 FORTRAN function, 7-106
Alphanumeric literals, 7-31
AMAX0 FORTRAN function, 7-106
AMAX1 FORTRAN function, 7-106, 6-106
AMIN0 FORTRAN function, 7-106
AMIN1 FORTRAN function, 7-106
AMOD FORTRAN function, 7-106
Ampersand (&),
  BASIC operator, 6-18, 6-33
  PAL8 operator, 5-18
AND,
  BASIC function, 6-58
  PAL8 AND symbol (&), 5-18
.AND. FORTRAN operator, 7-42
Angle brackets (<>),
  BASIC relations, 6-21
  in PAL8 comments, 5-23
  left (<),
    Editor command, 4-12
    monitor, 2-13
  PAL8 conditionals, 5-23
  right (>) Editor command, 4-12
Arctangent function, 6-45
Arguments system retention of, 2-17
Arithmetic expressions,
    BASIC, 6-17
    FORTRAN, 7-38
Arithmetic statements,
    FORTRAN, 7-45
    BASIC strings, 6-19
Arrays,
  BASIC, 6-24
  FORTRAN, 7-33, 7-47
ARROW SET command option, 3-62
ASC function, 6-52
ASCII,
  6-bit, 5-37
  character set, A-1
  conversion,
    BASIC function, 5-52
    PAL8, 5-22
  entering PAL8 text strings, 5-37
  file format, 2-26
ASF function, 7-70
ASIN FORTRAN function, 7-107
Assembling a PAL8 program, 5-5
  CREF command, 3-19
  PAL command, 3-53
Assembly language programs,
  Command Decoder use, D-1
  math subroutines, B-1
Assembler termination, 5-39
ASSIGN,
  command, 2-11, 3-2
  statement, 7-58
ASSIGNed GOTO, 7-58
Assigning logical device names, 3-2
Asterisk (*),
  BASIC, 6-33
  double (**),
    BASIC operator, 6-17
    FORTRAN operator, 7-39
  FORTRAN operator, 7-39
  prompting symbol, 2-21
  PAL8 location counter, 5-14
  PAL8 special character, 5-21
  wild character, 2-18
At (@) symbol indirect commands, 2-20
ATAN FORTRAN function, 7-107
ATAN2 FORTRAN function, 7-107
ATN function, 6-45
Autoindex registers, 5-28

B,
  Editor buffer command, 4-7
  ODT command, 9-5
/B,
  BASIC option, 6-79
  COMPARE option, 3-6
  COMPILE command option, 6-79

/B (cont.)
DIRECT command option, 3-25
Editor option, 4-21
PAL8 option, 5-9
.BA file name extension, 2-12
BACKSPACE statement, 7-86
BASIC, 6-1,
arithmetic expressions, 6-17
auto line numbering, 6-8
calling an old program, 6-4
changing program text, 6-7
character set, 6-2
command, 3-3
command summary, 6-80
commands, 6-3
commercial arithmetic, 6-19
compilation, 3-34
constants, 6-13
control statements, 6-36
creating a new program, 6-4
deleting text, 6-7
deleting program lines, 6-12
direct record I/O, 6-69
elements of the language, 6-13
ending a session, 6-11
erasing a program, 6-11
error messages, 6-86
execution, 3-34, 3-10
expression operators, 6-17
expressions, 6-17
file control, 6-62
file statements, 6-62
format control, 6-30
function summary, 6-83
functions, 6-43
interaction with BATCH, 6-80
interrupting program execution, 6-12
issuing monitor commands, 6-55
listing a program, 6-5
listing control, 6-12
logical functions, 6-58
merging text, 6-9
numeric,
constants, 6-13
field list use, 6-36
functions, 6-44
variables, 6-15
operating procedures, 6-3
overlay files, 6-80
overview, 6-1
printing format strings, 6-35
program termination, 6-41
program segmentation, 6-74
reading files, 6-65
relational expressions, 6-18
renaming a program, 6-10
resequencing a program, 6-11
running a program, 6-5
saving programs, 6-79
BASIC (cont.)
special print functions, 6-34
statements, 6-23
formatting, 6-22
summary, 6-81
storing a program, 6-10
string,
concatenation, 6-18
constants, 6-14
functions, 6-49
numbers, 6-20
relations, 6-22
variables, 6-15
subroutines, 6-42
subscripted variables, 6-16
user-defined functions, 6-78
using record I/O files, 6-72
variables, 6-14
writing files, 6-65
writing programs, 6-1
BASIC.SV, 3-3
BAT device handler, 3-67
Batch,
command processing, 8-3
commands, 8-2
error messages, 8-6
input file structure, 8-4
interaction with BASIC, 6-80
introduction, 8-1
SUBMIT command, 3-72
operating restrictions, 8-8
options, 8-2
BATCH.SV, 3-74
BCOMP.SV, 3-11,, 3-34
.BI file name extension, 2-12
BITMAP.SV, 3-48
Blank lines in FORTRAN programs, 7-25
BLOAD.SV, 3-34
BLOCK DATA statement, 7-54
.BN file name extension, 2-12
Boolean operators, PAL8, 5-18
BOOT,
command, 3-4
button, 2-6
BOOT.SV, 3-4
Bootstrapping OS/78, 2-6
Brackets ([]), 2-16
PAL8 special characters, 5-22
Breakpoints under ODT, 9-2
BTCHAx BATCH files, 3-74
Buffer size of Editor, 4-4
BYE command, 6-11

C,
Editor change command, 4-9
FORTRAN comment indicator, 7-23
ODT command, 9-7

/C,
  COMPARE option, 3-6
  COPY command option, 3-14
  DELETE command option, 3-24
  DIRECT command option, 3-25
  FORTRAN Loader option, 7-10
  FRTS option, 7-16
  LIST command option, 3-41
  LOAD command option, 3-44
  PAL8 option, 5-9
  RENAME command option, 3-56
  TYPE command option, 3-76
C:n/ QUEUE command option, H-2
CALL statement, 7-74
CANCEL command, 3-5, H-1
CAP$ function, 6-54
Caret - see circumflex
CCL.SV, 3-9, 3-3, 3-4
CDF instruction, 5-30, 5-43
CGET subroutine, 7-107
CHAIN,
  statement, 6-74
  USR function C-10
Chaining PAL8 programs, C-10
Character conversion function, 6-52
Character set, A-1
  BASIC, 6-2, 6-2
  FORTRAN, 7-20
  PAL8, 5-10
Characters,
  output by codes (BASIC), 6-60
  PAL8 formatting, 5-12
  wild, 2-18
CHKEOF FORTRAN subroutine, 7-108
CHR$ function, 6-53
CIF instruction, 5-30, 5-43
Circumflex (^) ODT command, 9-4
CLOCK FORTRAN subroutine, 7-108
CLOSE function (USR), C-8
CLOSE# statement, 6-64, 6-71
.CM file name extension, 2-12
COL,
  function, 6-60
  SET command option, 3-62
Comma (,),
  BASIC, 6-34
  FORTRAN use, 7-101
  PAL8 special character, 5-21
Command Decoder,
  calling from USR, C-1
  conventions, D-1
  calling, D-3
  error messages, D-2
  example of use, D-6
  option table, D-5
  special mode call, D-8
  tables, D-4
Commands, OS/78, 2-11, 3-1
  argument retention 2-17
Commands, OS/78 (cont.)
ASSIGN, 3-2
  BASIC, 3-3
  BOOT, 3-4
  CANCEL, 3-5
  COMPARE, 3-6
  COMPILE, 3-10, 7-4
  COPY, 3-12
  CREATE, 3-18
  CREF, 3-19
  DATE, 3-21
  DEASSIGN, 3-22
  DELETE, 3-23
  DIRECT, 3-25
  DUPLICATE, 3-28
  EDIT, 3-32
  EXECUTE, 3-34
  format, 2-13
  FORMAT, 3-35
  full word switches, H-1
  GET, 3-37
  HELP, 2-22, 3-38
  incorrect, 2-14
  LIST, 3-40
  LOAD, 3-42
  MAP, 3-46
  MEMORY, 3-50
  ODT, 3-52, 9-3
  PAL, 3-53
  QUEUE, 3-54
  R, 3-55
  RENAME, 3-56
  REQUEST, 3-57
  SAVE, 3-59
  SET, 3-62
  SQUISH, 3-70
  START, 3-71
  SUBMIT, 3-72, 8-2
  summary, 2-26
  TERMINATE, 3-75
  TYPE, 3-76
  UA/UB/UC, 3-78
  ZERO, 3-79
Comments,
  BASIC, 6-26
  FORTRAN, 7-23
  PAL8, 5-12
COMMON statement, 7-50
  interaction with EQUIVALENCE, 7-54
COMPARE command, 3-6
  error messages, 3-9
  option summary, 3-6
COMPILE command, 3-10
  option summary, 6-79, 7-5
Computed GOTO, 7-58
Concatenation, 6-18
Conditional,
  assembly operators, 5-36
  delimiters, 5-23

Conditional (cont.)
transfer,
BASIC, 6-37
FORTRAN, 7-60
Constants,
BASIC, 6-13
FORTRAN, 7-27
exponential, 7-29
integer, 7-28
Hollerith, 7-31
logical, 7-30
octal, 7-30
CONTINUE statement, 7-66
Control character,
representation, 2-7
uparrow (^) echo control, 3-63
Control statements,
BASIC, 6-36
FORTRAN, 7-56
COPY command, 3-12
option summary, 3-16
error messages, 3-17
Core Control Block, 3-55, 3-59, 3-71, 9-2
Correcting errors, 2-15
COS function, 6-45
COS FORTRAN subroutine, 7-108
COSH FORTRAN subroutine, 7-108
CPUT FORTRAN subroutine, 7-109
CREATE command, 3-18
Creating a PAL8 program, 5-4
CREF command, 3-19
error messages, 3-20
option summary, 3-19
CREFLS.TM file, 3-19
CREF.SV, 3-19, 3-19
Cross-reference listing, 3-19, 5-7
CTRL/C, 2-7, 2-15
BASIC use, 6-12
BATCH use, 8-5
COPY use, 3-15
EDIT use, 4-3
CTRL/G, 5-24
CTRL/L, 4-3
CTRL/O, 2-7
BASIC use, 6-6, 6-12
EDIT use, 4-3
ODT use, 9-11
CTRL/Q, 2-7
BASIC use, 6-12
CTRL/S, 2-7
BASIC use, 6-12
CTRL/U, 2-7
BASIC use, 6-12
FORTRAN use, 7-16
CTRL/Z,
FORTRAN use, 7-16
with device handlers, F-2, F-4
CUR$ function, 6-59
Current location counter, 5-21
changing with PAGE, 5-33
PAL8 symbol, 5-14
PAL8, 5-21
Current page literals, 5-22
Cursor function, 6-59

D,
Editor delete command, 4-9
ODT command, 9-7
/D,
DUPLICATE command option, 3-30
Editor option, 4-21
Dash options (general-purpose), 2-16
DAT$ function, 6-57
Data field register, 5-30
DATA statement, 6-28, 7-54
DATE,
command, 3-21
FORTRAN subroutine, 7-109
Date,
file transfer by, 3-14
function, 6-57
DEASSIGN command, 3-22
Debugging,
function (BASIC), 6-57
in octal, 3-52, 9-1
DECIMAL pseudo-operator, 5-36
Decimal to octal conversion, 6-55
Decimal point (.), 7-28
DECODE function (USR), C-10
DECstation System Hardware, 1-1
configuration summary, J-1
DEF statement, 6-56
Default,
file name extensions, E-1
file specifications, 2-21
DEFINE FILE statement, 7-86
DEFINE# statement, 6-70
DELETE,
BASIC command, 6-7
key, 2-7, 6-12, 7-16
OS/78 command, 3-23
error messages, 3-24
option summary, 3-24
Demonstration programs, 2-10
Device Control Word Table, C-15
Device handlers, 3-67
calling, F-1
description and use, F-1
FETCH function (USR), C-4
file-structured devices, 2-23, F-4
function control word, F-2
line printer, F-3
loading from PAL8 programs, C-4

Device handlers (cont.)
multiple input file restrictions, F-5
obtaining characteristics via USR, C-13
removal from memory, C-14
restrictions of use, F-3
terminals, F-5
Device names,
permanent, 2-10
reassigning, 2-10
DEVICE pseudo-operator, 5-37
Devices - see terminals also
file-structured, 2-23, F-4
FORTRAN I/O, 7-7
storage capacity, 2-23
using RX01/RX02 and RL01K, 2-24
using VXA0, 2-25
.DI file name extension, 2-12
DIM FORTRAN function, 7-109
Diagnostic messages,
FORTRAN, 7-5
PAL8, 5-39
DIMENSION statement,
BASIC, 6-24
FORTRAN, 7-47
Direct,
assignment statements, 5-16
record I/O, 6-69
DIRECT command, 3-25
option summary, 3-25
error messages, 3-27
DIRECT.SV, 3-27
Directory (file), 2-25
Disk pack - see RL01K,
Diskette - see RX01/RX02 also
bootstrap procedure, 2-5
making backup copies, 2-8
Dismissing USR, C-13
Division,
BASIC operator (/), 6-17
FORTRAN operator (/), 7-39
PAL8 operator (%), 5-18
DO statement, 7-62, 7-64
Document,
conventions, Preface
reference, Preface
dollar sign ($),
BASIC, 6-34
Editor symbol, 4-10
ESCape key echo, 3-64
FORTRAN field descriptor, 7-98
PAL8 special symbol, 5-24
Double,
asterisk (**), 6-17
quote ("), 5-22
Double-density diskette formatting, 3-30
DSK device handler, F-4
Dummy arguments in subprograms, 7-69
DUPLICATE command, 3-28
error messages, 3-31
option summary, 3-30

E,
Editor command, 4-7
FORTRAN field descriptor, 7-92
ODT command, 9-7
/E,
CREF command option, 3-19
DIRECT command option, 3-25
FRTS option, 7-16
PAL8 option, 5-9
SUBMIT command option, 3-72, 8-2
ECHO option (SET command), 3-62
EDIT command, 3-32
BASIC, 6-7
EDIT.SV, 3-18, 3-33
Editor - (see CREATE and EDIT commands also), 4-1
alteration commands, 4-9
append (A) command, 4-6
buffer (B) command, 4-7
buffer size, 4-4
calling procedure, 4-1
change (C) command, 4-9
changing existing file, 4-2
command mode, 4-2, 4-5
command summary, 4-23
command usage, 4-13 thru 4-18
control commands, 4-3
creating a new file, 4-1
delete (D) command, 4-9
Editor,exit (E) command, 4-7
error messages, 4-21,4-22
example session, 4-18
get (G) command, 4-6
insert (I) command, 3-6
interbuffer search (J) command, 4-10
kill (K) buffer command, 4-8
list (L) command, 4-6
list buffer (V) command, 4-9
listing commands, 4-6
next (N) command, 4-8
operating modes, 4-2
option summary, 4-21
output commands, 4-7
output current buffer (E) command, 4-7
output (P) command, 4-7
page (P) command, 4-4
quit (Q) command, 4-8
read (R) command, 4-5, 4-6
search character (S) command, 4-10

Editor (cont.)
search next (F) command, 4-10
special characters, 4-11
text mode, 4-2
text searches, 4-13
yank (Y) command, 4-10
EJECT pseudo-operator, 5-34
$END command (BATCH), 8-3
END statement, 6-41, 7-68
END# statement, 6-68.1
End-of-file,
BASIC, 6-68.1
FORTRAN ENDFILE statement, 7-87
PAL8, 5-37
ENTER function (USR), C-7
Exponentiation, 6-17
Equals sign (=), 2-16
BASIC, 6-21, 6-23
Editor command, 4-12
FORTRAN use, 7-45
PAL8 direct assignment, 5-16
PAL8 special character, 5-21
EQUIVALENCE statement, 7-52
interaction with COMMON, 7-54
.EQV. FORTRAN operator, 7-42
Error messages,
BASIC, 6-86
BATCH, 8-6
Command Decoder, D-2
COMPARE command, 3-9
COPY command, 3-17
CREF command, 3-20
DELETE command, 3-24
DIRECT command, 3-27
DUPLICATE command, 3-31
Editor, 4-21,4-22
FORMAT command, 3-36
FORTRAN Loader, 7-11
FRTS, 7-17
HELP command, 3-39
LIST command, 3-41
MAP command, 3-49
ODT, 9-8
OS/78 summary, G-1
PAL8, 5-39
SET command, 3-68
system halts, G-1
TYPE command, 3-77
ERROR function (USR), C-11
Errors, correcting keyboard, 2-15, 6-12
ESC SET command option, 3-62
ESCape key, 2-7
BATCH operation, 3-73
dollar sign ($) echo control, 3-64
Editor intrabuffer search command, 4-10
FORTRAN use, 7-14
terminator, 2-13
ESCape sequences in BASIC, 6-76
Exclamation point (!),
BASIC, 6-26
PAL8 operator, 5-18
EXECUTE command, 3-34
Executing PAL8 programs, 5-7
EXP function, 6-46
EXP FORTRAN function, 7-110
Exponential,
function, 6-46
operator (**), 7-39
real constant, 7-29
Expressions,
BASIC, 6-17
FORTRAN, 7-38
PAL8, 5-18
EXPUNGE pseudo-operator, 5-35
Extensions, file name 2-11, E-1
EXTERNAL statement, 7-48

F,
Editor search next command, 4-10
field descriptor, 7-91
/F,
COPY command option, 3-14
DIRECT command option, 3-25
PAL8 option, 5-9
F4.SV, 3-11, 3-34
/ff LOAD command option, 3-42
/ffnnnn, LOAD command option, 3-42
.FALSE. logical constant, 7-30
FETCH function (USR), C-4
FIELD pseudo-operator, 5-29
Fields,
BASIC numeric, 6-33
memory data and instruction, 5-30
FILE statement, 6-63
FILE# statement, 6-70
FILENAME pseudo-operator, 5-37
Files, 2-25
Files,
absolute binary, 3-42
ASCII format, 2-26
BASIC, 6-62
batch input, 3-72
changing ASCII, 4-2
closing, 6-71, C-8
creating ASCII, 4-1
deleting tentative, C-15
directory, 2-25
listing, 3-25
eliminating empty, 3-70
fixed length, 6-70
input table, D-4
memory-image,
creation, 3-59
loading and executing, 3-55, 3-71

Files (cont.)
  names, 2-10
    extensions, 2-12, E-1
  opening for record I/O, 6-70
  output spooling under BATCH, 3-73
  formatting BASIC output, 6-67
  output table, D-4
  permanent, C-6
  postdeletion, 3-13
  predeletion, 3-13
  protection during transfer, 3-14
  resetting BASIC, 6-67
  restrictions on multiple input, F-5
  specifications,
    using defaults, 2-21
    wildcard input, 2-19
    wildcard output, 2-19
  tentative, C-7
  testing for end, 6-68.1
  testing for open, 6-68
  transfer by date, 3-14
  transfer command COPY, 3-12
  types, 2-25
  USR search, C-6
  variable length string, 6-69
FILEV# statement, 6-69
FIX$ function, 6-55
FLOAT FORTRAN function, 7-110
FNa function, 6-56
FOR statement, 6-39
FORLIB.RL, 3-44, 7-19
FORM FEED, 5-12, 5-24
FORMAT command, 3-35
  error messages, 3-36
  option summary, 3-35
Format,
  ASCII files, 2-26
  BASIC statements, 6-22
  characters,
    PAL8, 5-12
  control,
    BASIC, 6-30
  statements,
    FORTRAN, 7-88
  OS/78 commands, 2-13
Formatted I/O (FORTRAN), 7-77
Formatting,
  double density diskettes, 3-30
  RL01K disk packs, 3-35
  output files in BASIC, 6-67
  single-density diskettes, 3-30
FORTRAN,
  arithmetic,
    assignment statements, 7-45
    conditionals, 7-60
  arrays, 7-33, 7-53
  assignment statements, 7-45

FORTRAN (cont.)
  auxiliary I/O statements, 7-85
  BLOCK DATA subprograms, 7-54
  carriage control, 7-100
  compilation, 3-34, 3-10
  constants, 7-27
  continuation lines, 7-24
  control statements, 7-56
  DATA statements, 7-54
  data types, 7-27, 7-27, 7-33
  direct access I/O, 7-80
  error message summary, 7-5
  execution, 3-34
  exponential constants, 7-29
  expressions, 7-38
    operators, 7-39
  field descriptors, 7-89
  FORMAT statements, 7-88
  format,
    control and I/O lists, 7-103
    specification separators, 7-101
  FRTS, 7-15
    error messages, 7-17
  grouping and group repeat specifications, 7-100
  Hollerith constants, 7-31
  I/O,
    devices, 7-7
    formatting, 7-80
    lists, 7-77
    statements, 7-76
  identification field, 7-25
  integer constants, 7-28
  introduction, 7-1
  label field, 7-23
  language, 7-19
    summary, 7-112
  library, 7-19
    functions, 7-76, 7-105
    subroutines, 7-105
  line format, 7-25
  literals, 7-97
  Loader, 7-7
    error messages, 7-11
    options, 7-10
  logical,
    assignment, 7-46
    conditionals, 7-62
    constants, 7-30
    expressions, 7-42
    operators, 7-42
    unit numbers, 7-77
  octal constants, 7-30
  operating procedures, 7-4
  program loading, 3-42
  real constants, 7-28
  relational expressions, 7-41
  relocatable binary files, 3-44

FORTRAN (cont.)
  runtime system (FRTS), 7-1, 7-12
    I/O assignment, 7-12
  scale factors, 7-99
  sequential I/O, 7-80
  special characters, 7-21
  specification statements, 7-46
  statements, 7-21
  storage declaration, 7-50
  subprograms, 7-69, 7-72
    declaration, 7-48
    user-written, 7-70
  subroutines, 7-73
  subscripts, 7-36
  symbols, 7-26
  variables, 7-31
FOTP.SV, 3-15, 3-23, 3-40, 3-56
FRTS, 7-12
FRTS.SV, 3-34
-FT dash option, 2-17
.FT file name extension, 2-12
FUNCTION statement, 7-72
Functions,
  BASIC, 6-43
  FORTRAN, 7-105
  user-defined, 6-78

G,
  Editor get command, 4-6
  field descriptor, 7-93
  ODT command, 9-5
/G,
  BASIC option, 6-79
  COMPILE command option, 6-79, 7-5
  FORTRAN Loader option, 7-10
  LOAD command option, 3-43, 3-45
  PAL8 option, 5-9
General-purpose dash options, 2-16
GET command, 3-37
GET# statement, 6-71
Global subroutine calls, 5-22
GOSUB statement, 6-42
GOTO statement, 6-36, 7-57

H field descriptor, 7-96
/H,
  PAL8 option, 5-9, 5-9
  SUBMIT command option, 8-2
Handler - see device handler,
HANDLER option (SET command), 3-62
Hardware configuration summary, J-1
HEIGHT option (SET command), 3-62
HELP command, 2-22, 3-38
  error messages, 3-39
X HELP.SV, 3-38
  .HN file name extension, 2-12
  Hollerith constants, 7-31

I,
  Editor insert command, 4-6
  field descriptor, 7-90
  PAL8 indirect addressing pseudo-operator, 5-29
/I LOAD command option, 3-43
I/O statements,
  FORTRAN, 7-76
I/O,
  BASIC, 6-26
  direct record (BASIC), 6-69
IDIM FORTRAN function, 7-110
IF END# statement, 6-68.1
IF GOTO statements, 6-37
IF OPEN# statement, 6-68
IF statement, 7-60
IFDEF pseudo-operator, 5-36
IFIX FORTRAN function, 7-110
IFNDEF pseudo-operator, 5-36
IFNZRO pseudo-operator, 5-36
IFZERO pseudo-operator, 5-36
Indirect,
  addressing, 5-29
  commands (@ symbol), 2-20
INIT option (SET command), 3-62
Input file table, D-4
INPUT# statemen6-65, 6-65
Input/output options, 2-15
INQUIRE function (USR), C-13
Instruction field,
  buffer, 5-30
  register, 5-30
Instructions,
  memory reference, 5-24
  PAL8, 5-11
INT function,
  BASIC, 6-47
  FORTRAN, 7-110
Integer function, 6-47
INTEGER, 7-27
IOR function, 6-58
IOT microinstructions, 5-28
ISIGN FORTRAN function, 7-110
Iteration,
  BASIC, 6-39
  FORTRAN, 7-62

J Editor command, 4-10
/J PAL8 option, 5-9
$JOB BATCH command, 3-72, 8-3
Job status word, 3-60

K Editor command, 4-8
/K PAL8 option, 5-9
KEY$ function, 6-59, 6-77
Keyboard errors, 2-15
KT8A Memory Management Control, 5-31

L,
  Editor list command, 4-6
  field descriptor, 7-94
  ODT command, 9-6
-L dash option, 2-17
/L,
  COMPILE command option, 7-5
  PAL9 option, 5-9
  QUEUE command option, H-2
  SET HANDLER command option, 3-67
LA34/38, 2-6
LA36, 2-6
LA12Ø, 2-6
Labels, 5-11
.LD file name extension, 2-12
LEN function, 6-5Ø
LET statement, 6-23
LINE FEED, 2-7
  Editor command, 4-12
  PAL8 special character, 5-24
  ODT command, 9-3
Link generation and storage, 5-38
LIST command, 3-4Ø
  BASIC, 6-5
LISTNH command, 6-6
Literals,
  FORTRAN, 7-97
  PAL8 current page, 5-22
  PAL8 storage, 5-39
LOAD command, 3-42
LOAD.SV, 3-11, 3-34
Loading PAL8 programs, 5-6
Loading,
  diskettes, 2-1
  RLØ1K disk packs, 2-3
Location counter - see current location counter
Locking USR in memory, C-12
LOG function function, 6-47
LOGICAL statement, 7-27
Logical,
  devices,
    assigning names, 3-2
    deassigning names, 3-22
    permanent names, 2-1Ø
  expressions, 7-42
  unit numbers, 7-77
LOOKUP,
  function (USR), C-6
Looping,
  BASIC, 6-39
  FORTRAN statements, 7-64
  nested (BASIC), 6-4Ø
Lower-case characters,
  conversion, 6-54
  line printer use, 3-64
  PAL8 restrictions, 5-11
LPT device handler, 3-67, F-3
LQP device handler, 3-67, F-3
-LS dash option, 2-17
.LS file name extension, 2-12

M command (ODT), 9-7, 9-8
/M,
  CREF command option, 3-19
  DIRECT command option, 3-25
  DUPLICATE command option, 3-3Ø
  SUBMIT command option, 3-72
Machine instruction set, 5-24
MAP command, 3-46
  option summary, 3-48
  error messages, 3-49
Mathematical subroutines, B-1
MAXØ function, 7-111
MAX1 function, 7-111
MEMORY command,, 3-5Ø
Memory,
  fields, 3-5Ø
  map, 3-46, 5-8
  page references, 5-38
  reference instructions, 5-24
  reserving with ZBLOCK, 5-33
Memory-image files,
  creation, 3-59
  execution, 3-71
  loading, 3-37,
  loading and executing, 3-55
Memory Management Control, 5-31
Microinstructions, 5-25
  IOT, 5-28
  operate, 5-26
MINØ function, 7-111
MIN1 function, 7-111
MIN12 function, 7-111
Minus (-),
  BASIC, 6-17, 6-34
  FORTRAN, 7-28, 7-39
  PAL8, 5-18
MOD function,, 7-111
Monitor,
  issuing commands from BASIC, 6-55
  period (.) prompting symbol, 2-6
-MP dash option, 2-17
.MP file name extension, 2-12

$MSG command (BATCH), 8-3
Multiplication,
  FORTRAN, 7-39
  PAL8, 5-18

N Editor next command, 4-8
/N,
  COMPILE command option, 7-5
  COPY command option, 3-13
  DELETE command option, 3-24
  DUPLICATE command option, 3-30
  PAL8 optioin, 5-9
  QUEUE command option, H-2

/n,
  FORMAT command option, 3-35
  MAP command option, 3-48
NAME command, 6-10
Natural logarithm, 6-47
-NB dash option, 2-17
Nested loops, 6-40
NEW command, 6-4
NEXT statement, 6-39
nnnn+ and nnnn- ODT commands, 9-4
Non-system device, 2-23
.NOT. FORTRAN operator, 7-42
Number sign (#), 6-33
Numbers,
  BASIC string, 6-20
  PAL8, 5-13
Numeric,
  constants, 6-13
  fields, 6-33, 6-36
  functions, 6-44
  variables, 6-15
Numeric-to-string conversion, 6-54

/O,
  DELETE command option, 3-24
  DIRECT command option, 3-25
  LIST command option, 3-41
  PAL8 option, 5-9, 5-30
  RENAME command option, 3-56
  TYPE command option, 3-76
OCS$ function, 6-55
OCT function, 6-54
OCTAL,
  pseudo-operator, 5-36
  statemnent, 7-27
Octal constants, 7-30
  3-52, 9-1
Octal Debugging Technique (ODT),
Octal-to-decimal conversion, 6-54
ODT command, 3-52, 9-3
  error messages, 9-8
ODT command (cont.)
  illegal characters, 9-5
  introduction, 9-1
  programming notes, 9-8
  setting breakpoints, 9-6
  setting search limits, 9-7
  special characters, 9-3
  summary, 9-9
  word searches, 9-8
OLD command, 6-4
ON GOTO statement, 6-38
ON GOSUB statement, 6-43
OPEN# statement, 6-68
Operands,
  as PAL8 symbols, 5-18
  PAL8, 5-12
Operate microinstructions, 5-26
Operators,
  PAL8 arithmetic and logical, 5-18
  PAL8 conditional assembly, 5-36
OR,
  BASIC function, 6-58
  PAL8, 5-18
.OR. FORTRAN operator, 7-42
OS/78,
  bootstrapping, 2-6
  command argument retention, 2-17
  Command Decoder, D-1
  commands, 2-11
    summary, 2-26
  devices,
    handlers, 3-6, F-3
    permanent names,
  error message summary, G-1
  file names and extensions, 2-10
  file-structured devices, 2-23
  full word command options, I-1
  hardware configurations, J-1
  introduction to, 1-1
  logical device names, 2-10
  making software backup, 2-8
  monitor commands from BASIC, 6-55
  startup procedure, 2-5
  system demonstration, 2-10
  User Service Routine, C-1
Output file table, D-4
Overlay files (BASIC), 6-80

P,
  Editor,
    output command, 4-7
    page command, 4-4
  FORTRAN scale factor symbol, 7-99

/P,
  CREF command option, 3-19
  DUPLICATE command option, 3-30
  FORMAT command option, 3-35
  LOAD command option, 3-44
-PA dash option, 2-17
.PA file name extension, 2-12
Page zero addressing, 5-29
PAGE,
  pseudo-operator, 5-33
  SET command option, 3-62
PAL command, 3-53
PAL8, 3-34, 3-53
  assembly, 3-10, 5-5
  chaining programs, C-10
  character set, 5-10
    restrictions, 5-11
  command string examples, 5-2
  comments, 5-12
  conditional,
    assembly operators, 5-36
    expressions, 5-23
  cross-reference listing, 5-7
  current location counter, 5-14
  current page literals, 5-22
  diagnostic messages, 5-39
  direct assignment statements, 5-16
  error messages, 5-10
  expressions, 5-18
  file extensions, 5-1
  FORM FEED character, 5-12
  formatting characters, 5-12
  instructions, 5-11
  introduction to, 5-1
  label, 5-11
  link generation and storage, 5-38
  listing control, 5-34
  literals, 5-39
  loading and saving a program, 5-6
  math subroutines, B-1
  memory map, 5-8
  numbers, 5-13
  off-page references, 5-38
  operands, 5-12
  option summary, 5-9
  page zero addressing
    pseudo-operator, 5-29
  permanent symbols, 5-14
  program,
    execution, 3-37
    loading, 3-42
  pseudo-operators, 5-29
  radix control, 5-36
  special characters, 5-21
  statements, 5-11, 5-13
  symbol table, 5-16
    alteration, 5-35
PAL8 (cont.)
    symbolic instructions, 5-17
    symbolic operands, 5-18
    symbols, 5-13
    TAB character, 5-12
    text strings, 5-37
    user-defined symbols, 5-14
    using Command Decoder, D-1
PAL8.SV, 3-19, 3-34, 3-53
Parentheses ( () ), 2-16
  FORTRAN, 7-44
  PAL8, 5-22
Patching a memory-image (SAVE) file, 3-43
PAUSE,
  pseudo-operator, 5-37
  SET command option, 3-62
  statement, 7-67
PDP-8 instruction set, 5-24
Percent (%), 5-18
Period (.),
  BASIC, 6-34
  Editor current line symbol, 4-11
  monitor prompting symbol, 2-6
  PAL8 special character, 5-21
Permanent file lookup, C-6
Permanent symbols, 5-14
PIP.SV, 3-70
Plus sign (+),
  BASIC operator, 6-17
  FORTRAN, 7-28
    carriage control character, 7-101
    operator, 7-39
  PAL8 operator, 5-18
PMT$ function, 6-60
PNT function, 6-31, 6-60, 6-76
POS function, 6-51
Postdeletion of files, 3-13
Predeletion of files, 3-13
Print head position function, 6-60
PRINT statement, 6-29
PRINT# statement, 6-65
PRINT USING statement, 6-32
PRINT# USING statement, 6-67
Program chaining, C-10
Prompt function, 6-60
Pseudo-operators,
  DECIMAL, 5-36
  DEVICE, 5-37
  EXPUNGE, 5-35
  EJECT, 5-34
  FIELD, 5-29
  FILENAME, 5-37
  IFDEF, 5-36
  IFNDEF, 5-36
  IFNZRO, 5-36
  IFZERO, 5-36
  OCTAL, 5-36

Pseudo-operators (cont.)
PAGE, 5-33
PAL8, 5-29
PAUSE, 5-37
RELOC, 5-34
TEXT, 5-36
XLIST, 5-34
ZBLOCK, 5-33
PUT# statement, 6-71

Q,
Editor quit command, 4-8
/Q,
COMPILE command option, 7-5
COPY command option, 3-15
DELETE command option, 3-24
LIST command option, 3-41
SUBMIT command option, 3-72, 8-2
TYPE command option, 3-76
Question mark (?) wild character, 2-18
QUEUE command, 3-54, H-2
Quote ("),
double - PAL8 special character, 5-22

R,
Editor read command, 4-5, 4-6
monitor command, 3-55
/R,
DIRECT command option, 3-25
DUPLICATE command option, 3-30
MAP command option, 3-48
.RA file name extension, 2-12
Radix control, 5-36
RALF.SV, 3-11
Random number function, 6-48
READ statement, 6-28, 7-80
READONLY option (SET command), 3-62
Real constants, 7-28
REAL statement, 7-27
Reassigning device names, 2-11
Record,
field definition, 6-70
size statement, 6-69
Reference Documents, Preface
Relational,
expressions,
BASIC, 6-18
FORTRAN, 7-41
operators,
BASIC strings, 6-22
RELOC pseudo-operator, 5-34
Relocatable binary files, 3-44
REM statement, 6-26
RENAME command, 3-56
RENAME command option summary, 3-56
REQUEST command, 3-57, H-1, H-2
RESEQ program, 6-11
RESET function (USR), C-14
Resetting BASIC files, 6-67
RESTORE statement, 6-28, 6-67
RETURN key, 2-7
BASIC, 6-12
ODT command, 9-3
PAL8 special character, 5-24
RETURN statement, 6-42, 7-75
REWIND statement, 7-87
.RL file name extension, 2-12
RL01K disk pack,
bootstrap procedure, 2-5
formatting, 3-35
making backup copies, 2-8
storage capacity, 2-23
use, 2-24
RLFMT.SV, 3-36
RLxx device handler, 3-67, F-4
RND function, 6-48
Rounding in BASIC, 6-21
RUN command, 6-5
RX01/RX02,
duplicating diskettes, 3-28
storage capacity, 2-23
use, 2-24
RXAx device handler, 3-68, F-4
RXCOPY.SV, 3-30

S Editor search character command, 4-10
/S,
BASIC option, 6-79
COMPARE option, 3-6
COMPILE command option, 6-79
DUPLICATE command option, 3-30
FORTRAN Loader option, 7-10
LOAD command option, 3-42, 3-44
MAP command option, 3-48
PAL8 option, 5-9
SAVE command, 3-59
BASIC, 6-10
Saving,
PAL8 programs, 5-6
BASIC programs, 6-79
SCOPE option (SET command), 3-62
SCRATCH command, 6-11
SEG$ function, 6-52
Semicolon (;) ODT command, 9-4
Semiconditional transfer, 6-38
SEQUENCE command, 6-8
SET command, 3-62
error messages, 3-68

SET command (cont.)
options, 3-62
Setting terminal characteristics, 2-5
SGN function 6-48,
SIGN FORTRAN function, 7-111
SIN,
BASIC, 6-45
FORTRAN, 7-111
Single-density diskette formatting, 3-30
SINH FORTRAN function, 7-112
Six-bit ASCII, 5-37
Slash (/), 2-16
BASIC operator, 6-17
BATCH command, 8-3
Editor symbol, 4-12
FORTRAN operator, 7-39
FORTRAN use, 7-101
ODT command, 9-3
PAL8 comments, 5-12
SLUx device handler, 3-68, F-5
Source file comparison, 3-6
Space character,
FORTRAN carriage control, 7-101
PAL8 operator, 5-18
PAL8 special character, 5-24
Spooler (symbiont) commands, H-2
Spooling, 3-73, 8-1, H-1
SPOOLR.SV, 3-54, H-1
SQR function, 6-46
SQRT FORTRAN function, 7-112
Square brackets ([]), 2-16, Preface
Square root function, 6-46
SQUISH command, 3-70
SRCCOM.SV, 3-9
START button, 2-6
START command, 3-71
Statement,
format,
BASIC, 6-22
FORTRAN, 7-24
terminators,
OS/78 monitor, 2-7
PAL8, 5-13
Statements,
FORTRAN, 7-21
PAL8, 5-11, 5-16
STEP statement, 6-39
STOP statement,
BASIC, 6-41
FORTRAN, 7-68
Storage capacity of devices, 2-23
STR$ function, 6-54
String,
arithmetic, 6-19
concatenation, 6-18
constants, 6-14
String (cont.)
integer function, 6-55
length function, 6-50
numbers, 6-20
to numeric conversion, 6-53
relations, 6-22
variables, 6-15
SUBMIT command, 3-72,
option summary, 8-2
Subprogram declaration, 7-48
SUBROUTINE statement, 7-73
Subroutines,
BASIC, 6-42
FORTRAN, 7-73
mathematical, B-1
PAL8 global calls, 5-22
semiconditional BASIC, 6-43
Subscripts,
BASIC, 6-16
FORTRAN, 7-36
Substring function, 6-51, 6-52
Subtraction,
BASIC, 6-17
FORTRAN, 7-39
PAL8, 5-18
.SV extension, 2-12, 3-59
Symbiont operation,
CANCEL command, 3-5
commands, H-1
CUSP coding conventions, H-2
REQUEST command, 3-57
spooler program, 3-54
writing your own, H-3
Symbol table,
PAL8, 5-16, 5-35, 5-41
Symbolic editor, 4-1
Symbols,
FORTRAN, 7-26
PAL8, 5-13, 5-41
use as instructions, 5-17
use as operands, 5-18
permanent, 5-14
user-defined, 5-14
SYS device handler, F-4
System Hardware, 1-1
System information,
additional documentation, Preface
HELP command, 2-22
System date, 3-21
System device, 2-23
System table reset via USR, C-14

T field descriptor, 7-97
-T TTY dash option, 2-17
/T,
COMPARE option, 3-6

/T (cont.)
  COPY command option, 3-14
  MAP command option, 3-48
  PAL8 option, 5-9
  RENAME command option, 3-56
  SUBMIT command option, 3-72, 8-2
TAB,
  BASIC function, 6-31
  character,
    PAL8 operator, 5-18
    PAL8 use, 5-12
  FORTRAN, 7-24
    function, 7-110
TAN BASIC function, 6-46
TANH FORTRAN function, 7-112
Tentative file lookup, C-7
Terminals,
  BASIC ESCape sequences, 6-76
  changing attributes, 2-5, 3-62
  control in BASIC, 5-59
  conventions, 2-7
  LA36, 2-6
  operating parameters, 2-6
  output formatting in BASIC, 6-32
  VT52, 2-6
  VT100, 2-5
TERMINATE command, 3-75
TEXT pseudo-operator, 5-37
TIME FORTRAN function, 7-112
.TM file name extension, 2-12
TRC function, 6-57
.TRUE. logical constant, 7-30
Truncation BASIC string
  arithmetic, 6-21
TTY device handler, F-5
TYPE command, 3-76
  option summary, 3-76
  error messages, 3-77

/U,
  COPY command option, 3-12
  CREF command option, 3-19
  DIRECT command option, 3-25
  SUBMIT command option, 3-72, 8-2
UA/UB/UC commands, 3-78
Unconditional transfer,
  BASIC, 6-36
  FORTRAN GOTO, 7-57
Underline (_) ODT command, 9-4
Unformatted FORTRAN I/O, 7-80
Unloading RL01K disk packs, 2-5
Uparrow (^),
  BASIC operator, 6-17
  echo control, 3-63
  ODT command, 9-4
  PAL8 operator, 5-18
Updating BASIC records, 6-71
Upper-case conversion, 6-54
User Service Routine (USR), C-1
User-defined functions in BASIC, 6-78
USR, C-1
  CHAIN function, C-10
  CLOSE function, C-8
  DECODE function, C-10
  ENTER function, C-7
  ERROR function, C-11
  FETCH function, C-4
  INQUIRE function, C-13
  LOOKUP function, C-6
  RESET function, C-14
  restrictions, C-3
  standard call, C-1
  summary of functions, C-1
  USRIN function, C-12
  USROUT function, C-13
USRIN function (USR), C-12
USROUT function (USR), C-13

V,
  Editor list buffer command, 4-9
/V,
  COPY command option, 3-12
  DELETE command option, 3-24
  DIRECT command option, 3-25
  LIST command option, 3-41
  RENAME command option, 3-56
  TYPE command option, 3-76
VAL function, 6-53
Value assignment, BASIC, 6-23
Variables,
  BASIC, 6-14
  FORTRAN, 7-32
  numeric (BASIC), 6-15
  string (BASIC), 6-15
  subscripted (BASIC), 6-16
VLUx device handler, 3-68, F-5
VT52 terminal, 2-6
VT100 terminal, 2-5
VXA0,device handler,
  characteristics, F-4
  installing with SET, 3-68
  storage capacity, 2-23
  use, 2-25

W ODT command, 9-8
/W PAL8 option, 5-9
WIDTH SET command option, 3-62
Wildcards, 2-18
WRITE statement, 7-83

X FORTRAN field descriptor, 7-97
/X,
  COMPARE command option, 3-6
  CREF command option, 3-19
XLIST pseudo-operator, 5-34
.XOR. FORTRAN operator, 7-42

Y Editor yank command, 4-10

Z PAL8 page zero addressing
  pseudo-operator, 5-29
ZBLOCK pseudo-operator, 5-33
ZERO command, 3-15, 3-79

# READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code or Country_____

Please cut along this line.

— — Do Not Tear - Fold Here and Tape — — — — — — — — — — — — — — — — — — — —

digital

No Postage Necessary if Mailed in the United States

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

RT/C SOFTWARE PUBLICATIONS ML 5-5/E45
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MASSACHUSETTS 01754

— — Do Not Tear - Fold Here — — — — — — — — — — — — — — — — — — — —