

Digital Equipment Corporation
Maynard, Massachusetts

digital

Programmer's Reference Manual

PDP-15 Utility Programs



PDP-15

UTILITY PROGRAMS

For additional copies, order No. DEC-15-YWZA-D from Program Library, Digital Equipment Corporation, Maynard, Massachusetts 07154 Price \$6.00

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing October 1969

Copyright © 1969 by Digital Equipment Corporation

The following are registered trademarks of Digital
Equipment Corporation, Maynard, Massachusetts:

DEC
FLIP CHIP
DIGITAL

PDP
FOCAL
COMPUTER LAB

CONTENTS

| | Page |
|---|------|
| CHAPTER 1 EDITOR | |
| SECTION 1 INTRODUCTION | |
| SECTION 2 FUNCTIONAL DESCRIPTION | |
| 2.1 Control Modes | 2-1 |
| 2.2 Data Modes | 2-1 |
| 2.2.1 Line-By-Line Data Mode | 2-1 |
| 2.2.2 Block Data Mode | 2-1 |
| 2.3 Data Files | 2-2 |
| 2.3.1 Using Monitor I/O | 2-2 |
| 2.3.2 Input and Subsidiary Files | 2-3 |
| 2.3.3 Output Files | 2-4 |
| 2.4 Using the Break (CNTRL P) Character | 2-5 |
| 2.5 Using the Erase and Kill Characters | 2-5 |
| SECTION 3 EDITING OPERATIONS | |
| 3.1 Modifying an Existing File | 3-1 |
| 3.2 Creating a New File | 3-2 |
| 3.3 Input/Edit Modes | 3-2 |
| 3.4 Block Mode | 3-2 |
| 3.5 Closing the New File | 3-3 |
| 3.6 Error-Handling Conventions | 3-3 |
| 3.6.1 Command String Errors | 3-3 |
| 3.6.2 Premature End-Of-File | 3-4 |
| 3.6.3 Read Errors and Line Overflow | 3-4 |
| 3.6.4 Block-Mode Buffer Overflow | 3-5 |
| 3.6.5 File-Naming and Calling Errors | 3-5 |
| 3.7 File Renaming and Deletion | 3-7 |
| SECTION 4 EDITOR COMMANDS | |
| 4.1 OPEN | 4-1 |
| 4.2 READ | 4-2 |
| 4.3 WRITE | 4-2 |

CONTENTS (Cont)

| | Page |
|------------------|------|
| 4.4 CLOSE | 4-2 |
| 4.4.1 ICLOSE | 4-2 |
| 4.4.2 SCLOSE | 4-2 |
| 4.5 NEXT [N] | 4-3 |
| 4.6 PRINT [P] | 4-3 |
| 4.7 FIND [F] | 4-3 |
| 4.8 LOCATE [L] | 4-3 |
| 4.9 DELETE [D] | 4-4 |
| 4.10 BOTTOM [B] | 4-4 |
| 4.11 RETYPE [R] | 4-4 |
| 4.12 INSERT [I] | 4-4 |
| 4.13 INSERT [I] | 4-4 |
| 4.14 GET [G] | 4-4 |
| 4.15 CHANGE [C] | 4-5 |
| 4.16 TOP [T] | 4-5 |
| 4.17 VERIFY [V] | 4-5 |
| 4.18 OVERLAY [O] | 4-6 |
| 4.19 APPEND [A] | 4-6 |
| 4.20 BRIEF | 4-6 |
| 4.21 BLOCK | 4-7 |
| 4.22 SIZE [S] | 4-7 |
| 4.23 EXIT | 4-7 |
| 4.24 OUTPUT | 4-8 |
| 4.25 CALL RENAME | 4-8 |
| 4.26 CALL DELETE | 4-8 |
| 4.27 RENEW | 4-8 |
| 4.28 KEEP | 4-8 |

SECTION 5 RECOVERY PROCEDURES

SECTION 6 EXAMPLES OF EDITING REQUESTS

APPENDIX A SUMMARY OF EDITING COMMANDS

APPENDIX B 339 DISPLAY EDITOR

CONTENTS (Cont)

| | Page |
|--|------|
| CHAPTER 2 PATCH UTILITY PROGRAM | |
| SECTION 1 INTRODUCTION | |
| 1.1 General Information | 1-1 |
| SECTION 2 CALLING PATCH | |
| 2.1 .DAT SLOT Assignments | 2-1 |
| 2.2 Calling PATCH | 2-1 |
| SECTION 3 COMMANDS | |
| 3.1 Command Input Format | 3-1 |
| 3.2 Commands | 3-1 |
| 3.2.1 Select Commands | 3-1 |
| 3.2.2 The List Command | 3-2 |
| 3.2.3 The READ Command | 3-5 |
| 3.2.4 The EXIT Command | 3-6 |
| 3.3 Error Recovery | 3-6 |
| 3.3.1 Error Messages | 3-6 |
| APPENDIX A PATCH PROGRAM FORMAT | |
| APPENDIX B EXAMPLE OF OPERATION | |
| CHAPTER 3 LIBRARY UPDATE UTILITY PROGRAM | |
| SECTION 1 INTRODUCTION | |
| SECTION 2 UPDATE COMMANDS | |
| 2.1 File Specification Commands | 2-1 |
| 2.1.1 Options | 2-1 |
| 2.1.2 File Name | 2-2 |
| 2.1.3 Command Terminator | 2-2 |
| 2.2 Update Action Commands | 2-2 |
| 2.2.1 DELETE (D) Command | 2-3 |

CONTENTS (Cont)

| | Page |
|---------------------------------|------|
| 2.2.2 REPLACE (R) Command | 2-3 |
| 2.2.3 INSERT (I) Command | 2-4 |
| 2.2.4 END (E) Command | 2-4 |
| 2.3 UPDATE Termination Commands | 2-5 |
| 2.3.1 CLOSE (C) Command | 2-5 |
| 2.3.2 KILL (K) Command | 2-5 |

SECTION 3 DEVICE (.DAT) SLOT ASSIGNMENTS

SECTION 4 ERROR CONDITIONS AND RECOVERY PROCEDURES

| | |
|---------------------------------------|-----|
| 4.1 Recoverable Errors | 4-1 |
| 4.1.1 Unintelligible Commands | 4-1 |
| 4.1.2 Command Function/Option Errors | 4-1 |
| 4.1.3 Improper Name in Action Command | 4-2 |
| 4.1.4 Incorrect Input Source | 4-2 |
| 4.2 Unrecoverable Errors | 4-2 |

SECTION 5 UPDATE EXAMPLE

| | |
|------------------|-----|
| 5.1 Update FILEA | 5-1 |
| 5.2 Update BCDIO | 5-1 |

CHAPTER 4 LINKING LOADER

SECTION 1 INTRODUCTION

SECTION 2 DESCRIPTION

| | |
|-------------------------------|-----|
| 2.1 Operation | 2-1 |
| 2.2 FORTRAN COMMON Statements | 2-1 |

SECTION 3 INFORMATION UNITS

SECTION 4 IDENTIFICATION CODES

CONTENTS (Cont)

| | Page |
|--|------|
| SECTION 5 PROGRAM UNIT ORGANIZATION | |
| 5.1 Main Program and Subprogram Organization | 5-1 |
| 5.2 Block Data Subprogram Organization | 5-1 |
| SECTION 6 DEFINITIONS | |
| SECTION 7 LINKING LOADER OPERATING PROCEDURES | |
| 7.1 Basic I/O Monitor Environment | 7-1 |
| 7.1.1 Structure of System Library | 7-2 |
| 7.1.2 Loader Memory Map | 7-2 |
| 7.1.3 Error Messages | 7-3 |
| 7.2 Advanced Monitor Environment | 7-4 |
| SECTION 8 MEMORY MAPS | |
| 8.1 Introduction | 8-1 |
| 8.2 I/O Monitor Environment | 8-2 |
| 8.3 Advanced Monitor Environment | 8-4 |
| APPENDIX A SYMBOL CONCATENATION - RADIX 50 ₈ FORMAT | |
| APPENDIX B LOADER SYMBOL TABLE | |
| CHAPTER 5 CHAIN AND EXECUTE SYSTEM PROGRAMS | |
| SECTION 1 INTRODUCTION | |
| 1.1 Program Description | 1-1 |
| 1.2 Calling Sequence | 1-1 |
| SECTION 2 CHAIN SYSTEM PROGRAM | |
| 2.1 Functional Description | 2-1 |
| 2.2 Operating Procedures | 2-2 |
| 2.3 Error Condition Typeouts | 2-4 |
| 2.4 Memory Allocation Typeout | 2-4 |

CONTENTS (Cont)

| | Page |
|--|------|
| 2.5 Memory Map at CHAIN Build Time | 2-5 |
| SECTION 3 EXECUTE SYSTEM PROGRAM | |
| 3.1 Functional Description | 3-1 |
| 3.2 Operating Procedures | 3-1 |
| 3.2.1 Special Operational Characteristics | 3-2 |
| 3.3 Memory Map at Execute Time | 3-2 |
| CHAPTER 6 PIP | |
| SECTION 1 INTRODUCTION | |
| SECTION 2 DEVICE ASSIGNMENTS | |
| 2.1 Basic I/O Monitor System | 2-1 |
| 2.2 Advanced Monitor | 2-1 |
| SECTION 3 PIP COMMAND STRING: GENERAL | |
| 3.1 Operation Character | 3-2 |
| 3.2 Device Name | 3-2 |
| 3.3 Filename and Extension | 3-3 |
| 3.4 Switch Options | 3-3 |
| 3.4.1 Data Modes | 3-3 |
| 3.4.2 Subsidiary Operations | 3-3 |
| SECTION 4 PIP FUNCTIONAL DESCRIPTION | |
| 4.1 Operations Under the Basic I/O Monitor | 4-1 |
| 4.1.1 Transfer File (T) | 4-1 |
| 4.1.2 Verify File (V) | 4-1 |
| 4.1.3 Segment File (S) | 4-1 |
| 4.2 Switch Options Under the Basic I/O Monitor | 4-2 |
| 4.2.1 Image Alphanumeric (I) | 4-3 |
| 4.2.2 IOPS Binary (B) | 4-3 |
| 4.2.3 IOPS ASCII (A) | 4-3 |
| 4.2.4 Bad Parity Correction (G) | 4-3 |

CONTENTS (Cont)

| | Page |
|--|------|
| 4.2.5 Tab to Space Conversion (E) | 4-4 |
| 4.2.6 Space to Tab Conversion (C) | 4-4 |
| 4.2.7 Segment File (Y) | 4-4 |
| 4.2.8 Combine Files (W) | 4-4 |
| 4.2.9 Insert Form Feed (F) | 4-4 |
| 4.2.10 Delete Trailing Spaces (T) | 4-4 |
| 4.2.11 Delete Sequence Numbers (Q) | 4-5 |
| 4.3 Operations Under the Advanced or Background/Foreground Monitor | 4-5 |
| 4.3.1 List Directory (L) | 4-5 |
| 4.3.2 New Directory (N) | 4-5 |
| 4.3.3 Delete File (D) | 4-5 |
| 4.3.4 Rename File (R) | 4-5 |
| 4.3.5 Copy Mass Storage Unit (C) | 4-6 |
| 4.3.6 Block Copy (B) | 4-6 |
| 4.4 Switch Options Under the Keyboard Monitor | 4-6 |
| 4.4.1 Image Binary (H) | 4-7 |
| 4.4.2 Dump Mode (D) | 4-7 |
| 4.4.3 New Directory (N) | 4-7 |
| 4.4.4 New Directory With † QAREA (S) | 4-8 |

SECTION 5 PIP COMMAND STRING

| | |
|---------------------------------|-----|
| 5.1 Transfer File (T) | 5-1 |
| 5.1.1 Copying Files | 5-1 |
| 5.1.2 Creating Files | 5-2 |
| 5.1.3 Listing Files | 5-2 |
| 5.1.4 Using the G Switch | 5-2 |
| 5.1.5 Using the C or E Switches | 5-3 |
| 5.1.6 Using the N or S Switch | 5-3 |
| 5.1.7 Using the W Switch | 5-4 |
| 5.1.8 Using the Y Switch | 5-4 |
| 5.2 Verify File (V) | 5-5 |
| 5.3 Segment File (S) | 5-6 |
| 5.4 List Directory (L) | 5-7 |
| 5.5 New Directory (N) | 5-7 |

CONTENTS (Cont)

| | Page |
|--------------------------------|------|
| 5.6 Delete File (D) | 5-8 |
| 5.7 Rename File (R) | 5-8 |
| 5.8 Copy Mass Storage Unit (C) | 5-8 |
| 5.9 Block Copy (B) | 5-9 |

SECTION 6 CORRECTION PROCEDURES

| | |
|--|-----|
| 6.1 tP (CTRL Key P) | 6-1 |
| 6.2 Rubout (RO) | 6-1 |
| 6.3 tU (CTRL Key U) | 6-1 |
| 6.4 PIP Error Detection and Correction | 6-2 |

APPENDIX A SUMMARY OF PIP COMMANDS

CHAPTER 7 DSKPTR/DSKSAV UTILITY PROGRAMS

SECTION 1 INTRODUCTION

SECTION 2 DSKPTR OPERATING PROCEDURE

SECTION 3 DSKSAV UTILITY PROGRAM

CHAPTER 8 PUNCH UTILITY PROGRAM

SECTION 1 INTRODUCTION

| | |
|----------------------------|-----|
| 1.1 Equipment Requirements | 1-1 |
| 1.2 Software Requirements | 1-1 |
| 1.2.1 Resident Programs | 1-2 |
| 1.3 Program Organization | 1-2 |

SECTION 2 OPERATING PROCEDURES

| | |
|----------------------------------|-----|
| 2.1 Load and Punch | 2-1 |
| 2.2 Error Detection and Recovery | 2-2 |

CONTENTS (Cont)

| | Page |
|---|------|
| SECTION 3 EXAMPLES | |
| 3.1 Producing an Executable User Program Tape | 3-1 |
| 3.2 System Program Changes | 3-1 |
| 3.3 Modifying User .DAT (Device Assignment Table) Slots | 3-2 |
| 3.3.1 Using an Alternate I/O Library Handler | 3-2 |
| 3.3.2 Using a Non-Standard I/O Handler | 3-3 |
| APPENDIX A I/O MONITOR SKIP CHAIN | |
| APPENDIX B LINKING LOADER IOC TABLE | |
| APPENDIX C RADIX 50 ₈ VALUES | |
| CHAPTER 9 DUMP UTILITY PROGRAM | |
| SECTION 1 INTRODUCTION | |
| 1.1 Operating Procedures | 1-1 |
| 1.1.1 Calling Procedure | 1-1 |
| 1.1.2 General Command Characters | 1-1 |
| 1.1.3 Command String | 1-1 |
| 1.2 Error Conditions | 1-2 |
| 1.3 Restart Procedures | 1-2 |
| 1.4 Example | 1-2 |
| CHAPTER 10 DDT | |
| SECTION 1 INTRODUCTION | |
| 1.1 General Information | 1-1 |
| 1.2 Terminology Used | 1-1 |
| SECTION 2 DEBUGGING WITH DDT | |
| 2.1 Loading the Program | 2-1 |
| 2.2 Using the Breakpoints | 2-1 |
| 2.3 Examination and Modification | 2-3 |

CONTENTS (Cont)

| | Page |
|-----------------------------------|------|
| 2.4 Type-Out Modes | 2-4 |
| 2.4.1 Address Modes | 2-5 |
| 2.5 Starting and Restarting | 2-6 |
| 2.6 Searching Operations | 2-6 |
| 2.7 Special Locations Used by DDT | 2-7 |
| 2.8 Symbol Definitions | 2-7 |
| 2.9 Patch File Output | 2-8 |
| 2.10 Patch File Input | 2-8 |
| 2.11 Co-resident Subroutines | 2-9 |
| 2.12 Indirect Address References | 2-9 |
| 2.13 Miscellaneous Features | 2-9 |

APPENDIX A SUMMARY OF COMMANDS

APPENDIX B MNEMONIC INSTRUCTION TABLE

APPENDIX C PATCH FILE FORMAT

CHAPTER 11 SRCCOM, SOURCE COMPARE UTILITY PROGRAM

SECTION 1 INTRODUCTION

| | |
|---------------------------------------|-----|
| 1.1 General Description | 1-1 |
| 1.1.1 Software Operating Environment | 1-1 |
| 1.1.2 Minimum Equipment Configuration | 1-1 |
| 1.2 Reference Material | 1-2 |
| 1.3 Special Symbols | 1-2 |

SECTION 2 OPERATION

| | |
|---|-----|
| 2.1 Loading Procedure | 2-1 |
| 2.2 Device Assignments | 2-1 |
| 2.3 Operating Sequence | 2-1 |
| 2.4 Command String | 2-2 |
| 2.5 Using Nonfile-Oriented Input Device | 2-3 |
| 2.6 File-Oriented SRCCOM Listing | 2-3 |

CONTENTS (Cont)

| | Page |
|--|------|
| SECTION 3 OUTPUT FORMATS | |
| 3.1 M Switch On | 3-1 |
| 3.1.1 Lines Inserted | 3-1 |
| 3.1.2 Lines Deleted | 3-1 |
| 3.1.3 Lines Changed | 3-2 |
| 3.2 M Switch Off | 3-3 |
| 3.2.1 Lines Inserted | 3-3 |
| 3.2.2 Lines Deleted | 3-3 |
| 3.2.3 Lines Changed | 3-4 |
| SECTION 4 ERROR RECOVERY | |
| 4.1 Operator Errors | 4-1 |
| 4.2 Software Errors | 4-1 |
| 4.3 Device Not Enabled | 4-2 |
| CHAPTER 12 SGEN SYSTEM GENERATOR | |
| SECTION 1 INTRODUCTION | |
| SECTION 2 GENERAL OPERATING PROCEDURES | |
| 2.1 Formation of Skip Chains | 2-1 |
| 2.1.1 DECTape or DECTape/Disk Systems | 2-2 |
| 2.1.2 Negative Skips | 2-3 |
| 2.2 Formation of Device Assignment Table (.DAT) | 2-3 |
| APPENDIX A SYSTEM GENERATION, STEP-BY-STEP PROCEDURE | |
| APPENDIX B PIP ERROR MESSAGES | |

CONTENTS (Cont)

Page

ILLUSTRATIONS

CHAPTER 1

SECTION 2

| | | |
|-----|--|-----|
| 2-1 | Schematic of Line Processing In Block and Normal Modes | 2-2 |
|-----|--|-----|

SECTION 6

| | | |
|-----|---|-----|
| 6-1 | Sample Input File | 6-2 |
| 6-2 | Input File Listing Marked for Correction | 6-3 |
| 6-3 | Hard-Copy Output of Editing Session (Sheet 1) | 6-4 |
| 6-3 | Hard-Copy Output of Editing Session (Sheet 2) | 6-5 |
| 6-4 | File Resulting From Editing Session | 6-6 |

CHAPTER 7

SECTION 1

| | | |
|-----|-------------------------|-----|
| 1-1 | Paper Tape Block Format | 1-1 |
|-----|-------------------------|-----|

CHAPTER 10

APPENDIX C

| | | |
|-----|--|-----|
| C-1 | | C-1 |
|-----|--|-----|

TABLES

CHAPTER 1

SECTION 2

| | | |
|-----|---|-----|
| 2-1 | Standard .DAT Assignments for Text Editor | 2-3 |
| 2-2 | Output File Conventions for the Text Editor | 2-4 |

CONTENTS (Cont)

Page

TABLES (Cont)

CHAPTER 6

SECTION 2

| | | |
|-----|-----------------------------------|-----|
| 2-1 | I/O Monitor .DAT Slot Assignments | 2-1 |
| 2-2 | Initial .DAT Slot Assignments | 2-2 |

SECTION 3

| | | |
|-----|--------------------------|-----|
| 3-1 | PIP Operation Characters | 3-2 |
| 3-2 | PIP Device Names | 3-2 |

SECTION 4

| | | |
|-----|---|-----|
| 4-1 | Legal Operation/Switch Combinations | 4-2 |
| 4-2 | Legal Switch Combinations for Transfer File (T) | 4-3 |
| 4-3 | Legal Operation/Switch Combinations | 4-6 |
| 4-4 | Legal Switch Combinations for Transfer File | 4-7 |

CHAPTER 8

SECTION 3

| | | |
|-----|---|-----|
| 3-1 | Standard Paper Tape .DAT Slot Assignments | 3-2 |
| 3-2 | Loader - I/O Correspondence Table | 3-3 |

CHAPTER 12

APPENDIX A

| | | |
|-----|--------------------------|-----|
| A-1 | Query/Response Procedure | A-1 |
|-----|--------------------------|-----|

PREFACE

This manual contains individual descriptions of the PDP-15 Utility programs provided as standard software for the majority of PDP-15 Systems. Each utility program is described in a complete, self-contained chapter of this manual. A quick-reference locator is provided.

OVERALL PDP-15 DOCUMENTATION STRUCTURE

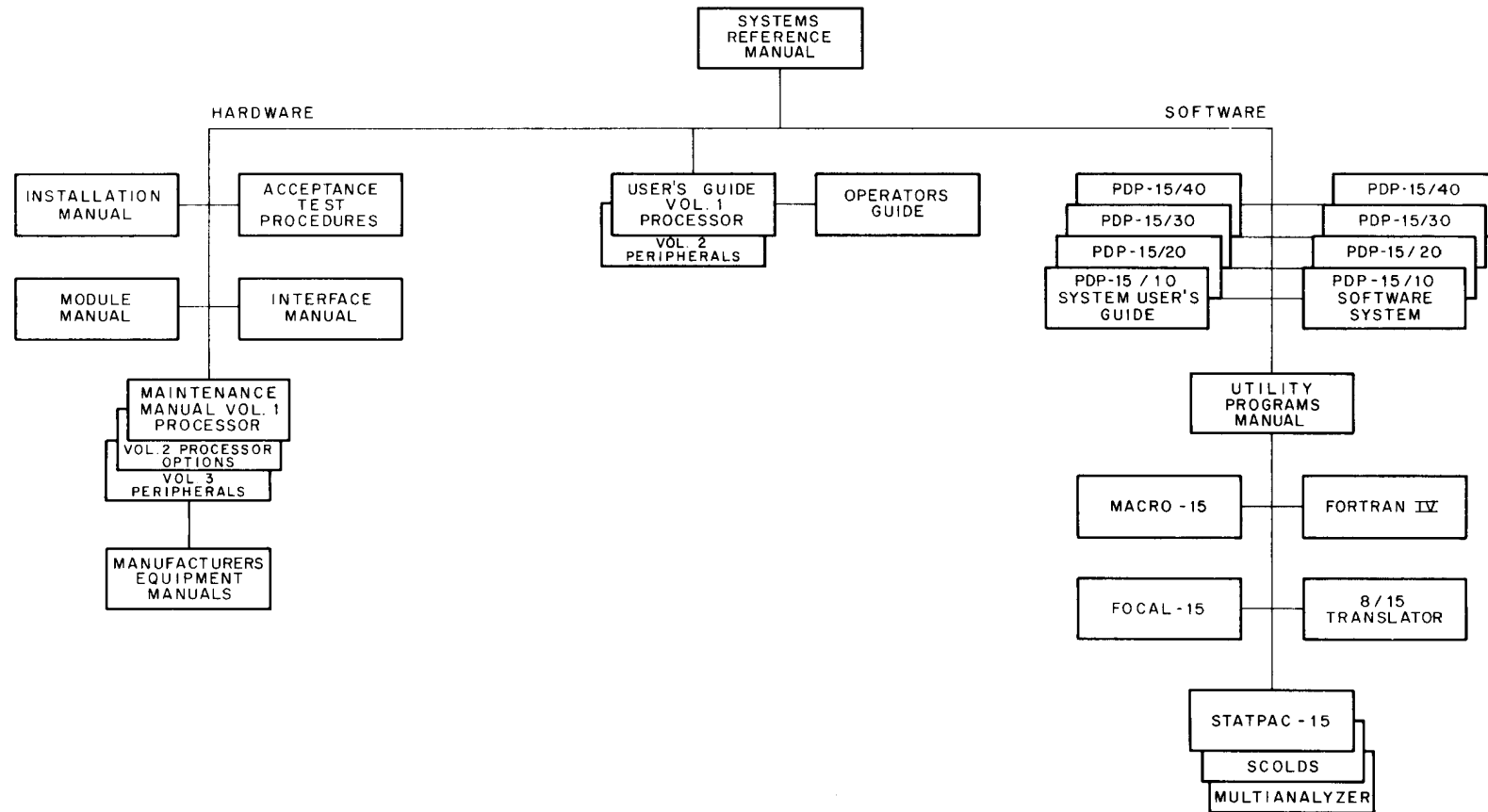
A block diagram of the overall PDP-15 Family of Manuals is illustrated on the following page, accompanied by a brief description of the contents and the order number of each manual.

ORGANIZATION OF PDP-15 SOFTWARE MANUALS

There are two basic categories of PDP-15 software manuals:

- a. Unique, single-system manuals, which contain information concerning only one of the four available PDP-15 Systems. This category consists of detailed software system descriptive manuals, each with an associated operational command summary. An example of this class of manual is the PDP-15/10 Software System manual and its associated PDP-15/10 User's Guide.
- b. Common, multisystem manuals that describe utility, language, application, and other PDP-15 programs which can be used in one or more of the four available PDP-15 Systems. Examples of this type of manual are the PDP-15 Utility, MACRO-15 Assembler, and STATPAC manuals.

PDP-15 FAMILY OF MANUALS



SYSTEM REFERENCE MANUAL — Overview of PDP-15 hardware and software systems and options, instruction repertoire, expansion features and descriptions of system peripherals. (DEC-15-GRZA-D)

USERS GUIDE VOLUME 1, PROCESSOR — Principle guide to system hardware includes system and subsystem features, functional descriptions, machine-language programming considerations, instruction repertoire and system expansion data. (DEC-15-H2DA-D)

VOLUME 2 PERIPHERALS — Features functional descriptions and programming considerations for peripheral devices. (DEC-15-H2DA-D)

OPERATOR'S GUIDE — Procedural data, including operator maintenance, for using the operator's console and the peripheral devices associated with PDP-15 Systems. (DEC-15-H2CA-D)

PDP-15/10 SYSTEM USER'S GUIDE — COMPACT and BASIC I/O Monitor operating procedures. (DEC-15-GG1A-D)

PDP-15/20 SYSTEM USER'S GUIDE — Advanced monitor system operating procedures. (DEC-15-MG2A-D)

PDP-15/30 SYSTEM USER'S GUIDE — Background/Foreground monitor system operating procedures. (DEC-15-MG3A-D)

PDP-15/40 SYSTEM USER'S GUIDE — Disk-oriented background/foreground monitor system operating procedures. (DEC-15-MG4A-D)

PDP-15/10 SOFTWARE SYSTEM — COMPACT software system and BASIC I/O Monitor system descriptions. (DEC-15-GR1A-D)

PDP-15/20 ADVANCED Monitor Software System — ADVANCED Monitor System descriptions; programs include system monitor and language, utility and application types; operation, core organization and input/output operations within the monitor environment are discussed. (DEC-15-MR2A-D)

PDP-15/30 BACKGROUND/FOREGROUND Monitor Software System — Background/Foreground Monitor

description including the associated language, utility and applications programs. (DEC-15-MR3A-D)

PDP-15/40 Disk-Oriented BACKGROUND/ FOREGROUND Monitor Software System — Background/Foreground Monitor in a disk oriented environment is described; programs include language, utility, and application types. (DEC-15-MR4A-D)

MAINTENANCE MANUAL VOLUME 1, PROCESSOR — Block diagram and functional theory of operation of the processor logic. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

VOLUME 2, PROCESSOR OPTIONS — Block diagram and functional theory of operation of the processor options. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

VOLUME 3, PERIPHERALS (Set of Manuals): — Block diagram and functional theory of operation of the peripheral devices. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

INSTALLATION MANUAL — Power specifications, environmental considerations, cabling and other information pertinent to installing PDP-15 Systems. (DEC-15-H2AA-D)

ACCEPTANCE TEST PROCEDURES — Step by step procedures designed to insure optimum PDP-15 Systems operation.

MODULE MANUAL — Characteristics, specifications, timing and functional descriptions of modules used in PDP-15 Systems.

INTERFACE MANUAL — Information for interfacing devices to a PDP-15 System. (DEC-15-H0AA-D)

UTILITY PROGRAMS MANUAL — Utility programs common to PDP-15 Monitor systems. (DEC-15-YWZA-D)

MACRO-15 — MACRO assembly language for the PDP-15. (DEC-15-AMZA-D)

FORTRAN IV — PDP-15 version of the FORTRAN IV compiler language. (DEC-15-KFZA-D)

FOCAL-15 — An algebraic interactive compiler level language developed by Digital Equipment Corporation. (DEC-15-KJZA-D)

SECTION 1 INTRODUCTION

The Text Editor (EDIT) is a powerful context-editing program that allows the modification and creation of symbolic source programs and other ASCII text material.[†] By means of commands issued from the teletype, the Editor is directed to bring a line, or group of lines, from the input file to an internal buffer. The user can then, by means of additional commands, examine, delete, and change the contents of the buffer, and insert new text at any point in the buffer. When the line, or block of lines, has been edited, it is written into a new file on the output device.

The Editor is most frequently used to modify MACRO and FORTRAN IV source programs, but it can also be used to edit any symbolic text.

The Editor operates in the ADVANCED Software System with either the I/O or Keyboard Monitor and can be used with all standard peripheral devices. In systems with greater than 8K, the additional memory (except that reserved for the Monitor and the required device handlers) is utilized for block mode buffers.

[†]The Editor reads and writes standard IOPS ASCII lines. The characteristics of IOPS ASCII text are described in the applicable Software System description manual (refer to Preface for a list of applicable manuals.)

SECTION 2

FUNCTIONAL DESCRIPTION

2.1 CONTROL MODES

The Editor operates in one of two control modes; in Edit (or Command) Mode the program accepts and acts upon control word and data strings to open and close files; to bring lines of text from an open file into the work area; to change, delete, or replace the line currently in the work area; and to insert single or multiple lines after the line in the work area. In Input (or Text) Mode, lines from the teletype are interpreted as text to be added to the open file. Commands are available for conveniently changing control mode.

2.2 DATA MODES

Data from the input file is made available for editing in two ways: in Line-By-Line Mode or in Block Mode.

2.2.1 Line-By-Line Data Mode

In Line-By-Line Data Mode, a single line is the unit of the input file available to the user for modification at any point. The line currently available is specified by a pointer, which can be thought of as moving sequentially through the file, starting at the first line, in response to typed editing commands. When a file is opened at the beginning of an editing session, the first line of that file is brought into the work area and is available for modification. This line remains in the work area until the user requests that a new line be brought in. The pointer then moves down the file until the line requested is encountered. That line is brought to the work area and, as the "current line," can be modified. Lines previously skipped over are no longer available for editing by the user, but are written in the output file. Thus, at any point in a single edit run in line-by-line mode, the user is able to modify only the portion of the input file consisting of the current line and all lines between the current line and the end of the file (i.e., the current line and all lines below it).

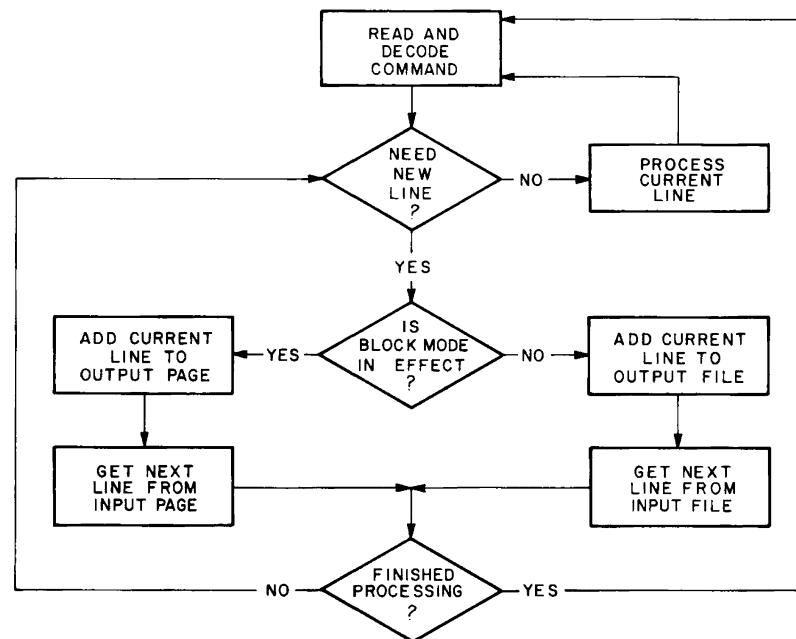
2.2.2 Block Data Mode

In Block Data Mode, a user-specified portion of the input file is held in a core buffer for editing until the user requests that the contents of the buffer be added to the output file. A group of Editor commands is available for use in Block Mode only (see Section 4) in addition to the commands used in line-by-line editing.

When the user is operating in Block Mode, commands to the Editor are honored only with respect to that portion of the input file currently occupying the buffer. The lines of text in the buffer are made available for modification through the use of normal locative requests and can be reaccessed until the buffer is emptied by the user.

Unless deleted, lines passed over in Block Mode are not lost to the user (as in Line-By-Line Mode) until the contents of the buffer are written in the output file. Consider, for example, the editing request to search for and bring in a specified line. In Line-By-Line Mode, the result is a scan of (possibly) the entire file below the pointer. The same request in Block Mode provides a search of the entire buffer below the pointer, but no further.

Block Mode has another advantage: rapid correction of editing command errors. If the user finds that he has typed the wrong command, he can immediately correct it, because the buffer has not been added to the output file. In Line-By-Line Mode, a command error may cause the program to bypass a line in which a change is needed. The user must then output a new input file and begin editing again.



150065

Figure 2-1 Schematic of Line Processing in Block and Normal Modes

2.3 DATA FILES

2.3.1 Using Monitor I/O

The Editor makes use of the Monitor Input/Output Programming System for I/O transfers and communicates with IOPS by way of entries in the Device Assignment Table. Entries in DAT which are required by the Editor are given in Table 2-1. Methods of modifying DAT are described in the applicable Software System manual (refer to the Preface for a list of manuals).

Table 2-1
Standard .DAT Assignments for Text Editor

| .DAT Entry Number | Used For |
|---|--|
| -3 | Teleprinter output; messages to user |
| -2 | Keyboard input; text and commands |
| -14 | File input |
| -15 | Scratch or edit file output [†] |
| -10 | Subsidiary file input |
| [†] The use of the scratch device is described under Output Files. | |

2.3.2 Input and Subsidiary Files

The Editor will accept file input from a maximum of two devices, in addition to input from the keyboard. The first device normally holds a previously prepared file upon which changes are to be carried out. The second, the subsidiary file device, is usually the medium through which additional, previously prepared, text is inserted in the object file. Either one, or both, of these devices may be ignored by the user, in which case the Editor assumes that all data will come from the keyboard.

Care must be taken in the specification of the subsidiary input device to ensure that the data of interest residing thereon was recorded in nonfile-structured fashion. This is the only recording mode for the paper tape and card reader. The user has the choice of writing data in either a file-structured, or a nonfile-structured, manner for other devices (e.g., DECtape).^{††} The characteristics of the subsidiary input device are determined when the Editor is first loaded. If that device can be file-structured, the comment

SECONDARY INPUT DEVICE IS FILE-ORIENTED

is printed on the teletype to warn the user that disastrous results will occur if the data to be read from the device is file-structured. Note, however, that if the data to be read was recorded in nonfile-structured fashion, then the requested device is a legal one for secondary input. Accordingly, the Editor then asks the question,

DO YOU WISH TO CONTINUE?

The user's answer indicates the nature of the data on the secondary input device. If the user's response is

YES)

^{††} For a discussion of data-handling conventions in file-structured and nonfile-structured input/output modes, see the applicable Software System manual (refer to Preface for a list of applicable manuals).

then the program reads data from the device in the normal (nonfile-structured) way. If the user's answer is NO (or anything except YES) file-structured data is assumed, and return is made (via .EXIT) to the Monitor.

2.3.3 Output Files

The Editor attempts to determine whether or not the input and scratch devices are file-structured immediately on receiving control after having been loaded. If either one of the devices is not file-structured, then the scratch device (DAT entry -15) is assigned as the final output device. If both devices are file-structured, the scratch device is assigned an intermediate function and the input device is used as the final output device.

The intent, in all cases, is to allow replacement of the input file by the edited output file. This is possible only when the input and output devices can be both read and written. If replacement can be accomplished (both devices are file-structured), the following sequence of events takes place when the files are closed after editing.

- a. The intermediate output file is read from the scratch device and written on the input device under a temporary name.
- b. The old input file is deleted from the input device.
- c. The intermediate output file is deleted from the scratch device.
- d. The intermediate output file, temporarily named and now residing on the input device, is given the name previously assigned to the old (now deleted) input file.
- e. The output file is closed and immediately becomes available for use.

If no replacement can be accomplished, no change is ever made to the input file. If the output device is file-oriented, the new edited file is properly entered in the file directory for that device under the name given in the OPEN or CLOSE command sequences.

The possible destinations of the new edited file are summarized in Table 2-2.

Note that in the process of file housekeeping, there is always at least one copy of the output file available on one, or both, of the devices. Further, the original input file is not deleted until the new file has been successfully written and closed. A system failure, therefore, can never result in total loss of data. Recovery procedures to be used in case of difficulty are outlined in Chapter 5.

Table 2-2
Output File Conventions for the Text Editor

| Input Device | Scratch or Output Device | Edited File appears on: | Input File is: |
|------------------|--------------------------|-------------------------|----------------|
| File-oriented | File-oriented | Input Device | Deleted |
| File-oriented | Nonfile-oriented | Output Device | Unchanged |
| Nonfile-oriented | File-oriented | Output Device | Unchanged |
| Nonfile-oriented | Nonfile-oriented | Output Device | Unchanged |

2.4 USING THE BREAK (CNTRL P) CHARACTER

Frequently the user, having made a mistake in his command line, wishes to stop processing and re-issue his request. The user, for example, may have asked erroneously for a line which is absent from the input file. When the Editor begins its search for the requested line, it will not give up until that line is found, or until the end of the input file is encountered. The user, meanwhile, has noticed his typing mistake. Control must now be transferred from the command processor to the command decoder.

The Editor's break, or quit, character provides the mechanism for the orderly accomplishment of the transfer. When the user types the quit character (CNTRL P) during command processing, the normal instruction sequence is interrupted when processing of the current line has been completed, Edit Mode is reentered, and the program reads a new edit command from the keyboard. Nothing is lost from the output file. Depending on the command being serviced when CNTRL P was typed, the pointer is left in one of two positions. In the first (usual) case, the pointer indicates the line which was being processed when the break character appeared. This line is now the current line, and is dealt with in the normal way. In the second case, the pointer is left between two lines. The current-line area is empty, and some locative request (e.g., NEXT) must be issued to move a line into the work area.

The break character results in program restart when the Editor is waiting for a command. In Input Mode, the break character results in a control mode change.

2.5 USING THE ERASE AND KILL CHARACTERS

The Monitor allows the use of two keyboard characters for correction of the line currently being typed by the user. The Rubout key (Erase character) results in the deletion of the immediately preceding character. The Monitor echoes a backslash (\) for each Rubout typed. CNTRL U (Kill Line character) results in the deletion of the entire line as typed to the CNTRL U. The Monitor echoes a commercial "at" sign (@) for each CNTRL U typed.

CNTRL U has a second function when used during output from the Editor to the teletype. When the user types CNTRL U while a line is being printed, output is immediately terminated, and a carriage return is issued. CNTRL U functions in this case as the user's means of overriding his previous request for the output of tediously long lines.

SECTION 3

EDITING OPERATIONS

The Editor always begins in Edit Mode and assumes that the user wishes to modify some (named or unnamed) file. When first loaded, or when restarted for a new file, the program types

EDITOR V_nx where n = version number
 x = modification level

on the teleprinter and waits for the user's first command.

3.1 MODIFYING AN EXISTING FILE

If the input device is file-structured (disk, drum, magnetic tape, or DECtape), the first command to the Editor must be

```
OPEN filename ext)
```

where "filename" is the primary name[†] of the wanted file residing on the input device and "ext" is its extension.^{††} Ext can be omitted and, if so, is assumed to be SRC. If the file specified is not found in the directory, the program assumes that the user wishes to create a file named "filename ext." Accordingly, when it has been determined that the named file is absent from the input device, the Editor types

```
FILE filename ext NOT FOUND.
INPUT
```

Input Mode is entered and subsequent lines from the teletype are inserted in a new temporarily named file on the output device.

If the specified file is present on the input device, an intermediate temporarily named file is opened for writing on the output device, and the input file is opened for reading. The user may then proceed to make the necessary changes in the input file.

If the input device is not file-structured (e.g., paper tape reader, card reader), the user's first command after program initialization can be any edit request. The OPEN command is not required for nonfile-structured devices.

[†]Maximum of six characters permitted for "filename".

††Maximum of three characters permitted for "ext".

3.2 CREATING A NEW FILE

When the user wishes to create a new file, he need only issue a carriage return, thereby entering Input Mode. If the output device is file-structured, a temporarily named file is opened for writing and text lines from the teletype are added to it as they appear. If the output device is not file-structured, the file-naming conventions are bypassed.

When both input and output devices are file-structured, the user can issue the OPEN command, followed by the name he wishes to assign to his new file. Since a file of the name given is guaranteed not to be found (if the user has properly chosen his new name), Input Mode is immediately entered, following the standard error message. The name specified is assigned to the final output file if no other name is given in the CLOSE command.

3.3 INPUT/EDIT MODES

To enter text from the teletype, the Editor must be in Input Mode. To carry out an edit function on the current line, the Editor must be in Edit Mode.

Control mode may be changed at any time by typing a line of zero length (a line consisting of a carriage return only). The Editor command INSERT (without arguments) also causes a mode change. After the user changes control modes, the Editor types INPUT or EDIT, indicating the control mode in effect.

3.4 BLOCK MODE

The Editor recognizes several commands which are designed to be useful in the Block, or Page, Mode. In Block Mode, a user-specified portion of the input file is held in a core buffer until the user indicates his satisfaction with the current state of that portion. Block Mode is entered via the control word BLOCK, followed by the parameter ON. When in Block Mode, the user can take advantage of all the locative and manipulative commands (FIND, LOCATE, CHANGE, etc.) and, in addition, can employ the TOP command to re-examine portions of the text buffer.

Line-By-Line Mode is re-entered by use of the BLOCK OFF command. If the BLOCK OFF command is issued before the buffer is empty, the following comment is printed:

BUFFER NON-EMPTY

The user must empty the buffer to terminate Block Mode.

3.5 CLOSING THE NEW FILE

When the user, after modifying his input file, is satisfied that all needed changes have been carried out, he is required to close out the input and output files. The edit command

CLOSE filename ext)

initiates the sequence of events described above (Paragraph 2.3.3).

Neither "filename" nor "ext" need be specified if previously given in the OPEN command. If filename and ext are present in the command string, they override the names given in the OPEN command.

Both filename and ext are ignored if the output device is nonfile-oriented.

3.6 ERROR-HANDLING CONVENTIONS

3.6.1 Command String Errors

All mistakes in the use of Edit Mode control words result in a common complaint by the Editor. Although the possible errors in usage fall into a number of distinct categories, the program makes no attempt to differentiate among error types. The reasons for this common treatment lie in the requirement that the Editor take some cognizance of its memory allocation (relatively obscure error types need as much memory for recognition and response as do the more usual mistakes) and in the fact that the treatment rendered makes the error self-explanatory.

Command string errors, then, all result in the single typed comment,

NOT A REQUEST:

followed, on the next line, by the request line with which the Editor had trouble.

Usual types of command string errors include the following:

- a. The edit control word issued was not among those in the program's repertoire.
- b. A SIZE command was issued with a missing argument or an argument of "1."

When the BRIEF Mode (see Paragraph 4.20) is ON, the Editor comment and the command line in error are replaced by a single typed question mark:

?)

3.6.2 Premature End-Of-File

During the processing of some commands, it occasionally happens that a read is attempted which moves the pointer below the last line of a logical (or physical) group. Consider, for example, the effect of a numeric argument in the GET n command line. The program reads successive lines from the subsidiary input device until exactly n lines have been read. If, in the process of reading, it is discovered that fewer than n lines are physically present on the secondary input medium (paper tape, say), then a premature end-of-file condition is said to exist. An improperly formulated FIND request (the character string typed is absent from the file) results in a similar condition.

Depending on the character of the incoming group of lines (block buffer, secondary input medium, or input file), the appearance of an unexpected end-of-file causes a comment to be typed to inform the user of the difficulty. The form of the message is:

END OF $\left\{ \begin{array}{c} \text{BUFFER} \\ \text{MEDIUM} \\ \text{FILE} \end{array} \right\}$ REACHED BY:

followed, on the next line, by the edit request which caused the problem.

A premature end-of-file causes the pointer to be left below the final line of the group being read.

3.6.3 Read Errors and Line Overflow

The Editor recognizes two types of errors which may occur during the processing of the input file. Both errors result in an appropriate printed comment and immediate transfer of control to the command decoder. The line in error is printed and left in the work area for modification by the user.

The first type of error occurs when the input file device handler detects either incorrect parity or a faulty checksum in the incoming line. The printed comment is:

READ ERROR:

followed by the line in which the error was encountered.

The second difficulty results from the appearance of a line which is too long to be contained in the program's internal buffers. Any line of more than 90₁₀ characters (not including terminator) results in the comment:

TRUNCATED:

followed by the first (leftmost) 90 characters of the long line. The remaining right-end characters are discarded.

The user has the choice, after either type of error, of modifying the line which caused the complaint (via any manipulative request) or of allowing the line to stand as is in the output file (via any locative request).

3.6.4 Block-Mode Buffer Overflow

When Block Mode is in effect, it is possible for an attempted addition of a line to the Block-Mode Buffer to exceed the buffer's capacity. This might occur, for example, during the processing of a READ request if the buffer length (previously defined by a SIZE command) is too great to be accommodated by the memory available. When the capacity of the buffer is exceeded, the program types the comment:

BUFFER CAPACITY EXCEEDED BY:
(offending line)

To eliminate this error condition, the operator must delete the excess (offending) line. The user should control carefully the size specification of the buffer and lines entered to ensure that this error condition is avoided.

3.6.5 File-Naming and Calling Errors

Errors in filename usage can be classified in three general groups. Either the named file cannot be found, or a name has not been given to the file at a point where one is needed, or a name has been given which cannot be used.

3.6.5.1 Absent File - If the file named in the OPEN request line cannot be found on the device associated with DAT slot-14, the assumption is made that the user wishes to create a new file with the name given. The program prints the comment:

FILE [filename ext] NOT FOUND.

and changes to Input Mode.

3.6.5.2 Absent File Name - If no filename is given either in an OPEN request line or as an argument to the CLOSE command, the program, after attempting to process the CLOSE request, prints:

NO FILE NAME GIVEN.

The next edit request must be another CLOSE naming the file.

If no OPEN command is issued (a new file is being created), any locative request (FIND, NEXT) will result in the comment:

NO INPUT FILE PRESENT.

3.6.5.3 Identically-Named Files - The problem of duplicate file names is apparent on two levels. In the first case, it is possible for a previous edit run to have been aborted with one of the Editor's temporary files (normally .TFIL1 EDT) closed on the output device. The closing of the temporary file created during the current edit run results in the deletion of the like-named file from the previous run. To enable the retrieval of prior work, the Editor types the comments:

FILE .TFIL1 EDT IS PRESENT ON OUTPUT DEVICE.
PLEASE RENAME IT OR IT WILL BE DELETED.

If the user wishes to preserve the contents of .TFIL1 EDT, he must rename it using the CALL request (see Paragraph 3.7).

At the second level, it may happen that the file name given in a CLOSE sequence is identical to that of another file on the (current) output device. In most cases, the program types:

PLEASE USE ANOTHER NAME.

A second CLOSE request (with a unique name) can then be issued. If file processing has proceeded to a point at which recovery, as described above, is impossible, the Editor recognizes a priority scheme when file-name difficulties are encountered. An attempt is made, first, to ensure that the new (modified) version of the file being edited is left on .DAT slot -14 and properly named. If that is impossible, the program tries to leave the new file (again, properly named) on .DAT slot -15. If that cannot be done either, then the new file is left on .DAT slot -15 and is named .TFIL1 EDT. The Editor then reports the nature of the difficulty, the final destination of the file, and its current name, thus:

FILE [filename ext] IS PRESENT ON OUTPUT DEVICE.
YOUR EDITED FILE IS ON .DAT-14 (OR -15) AS [newfile ext]
ORIGINAL FILE DELETED.

The user now knows the residence of his edited file (.DAT-14 or .DAT-15) and the name under which it can be accessed.

3.6.5.4 Nothing in File – The following error message can result from issuing CLOSE command prior to WRITE command when Block Mode is ON, or by having OUTPUT turned off when a WRITE or CLOSE command is issued:

NOTHING IN FILE

In any event, the control returns to the Editor. The contents of the buffer remain unchanged. In the case of file-oriented input and output, the input file is left unchanged.

3.7 FILE RENAMING AND DELETION

The CALL function allows user access to the file renaming and deletion facilities in the Editor. This command cannot be issued while any file (either input or output) is open. That is, it can only be used immediately after the Editor is loaded or restarted.

Format:

$$\text{CALL RENAME} \left\{ \begin{array}{c} \text{INPUT} \\ \text{OUTPUT} \end{array} \right\} \text{FILE1 EX1 FILE2 (EX2)}$$

The file FILE1 EX1 on .DAT-14 (if INPUT is specified) or .DAT-15 (if OUTPUT is specified) is given the new name FILE2. EX2 need not be given. If it is not, EX1 is used; if it is given, EX2 is used.

$$\text{CALL DELETE} \left\{ \begin{array}{c} \text{INPUT} \\ \text{OUTPUT} \end{array} \right\} \text{FILNAM EXT}$$

The file named FILNAM EXT is deleted from the device specified (INPUT or OUTPUT). EXT need not be given and, if it is not, SRC is used.

SECTION 4

EDITOR COMMANDS

When Edit Mode is in effect, the following commands result in the specified activity. Abbreviations for most commands consist of the initial characters of those commands. Legal abbreviations are given in square brackets. Optional arguments are given in parentheses.

Certain commands (e.g., FIND, RETYPE) require the presence of arguments. Others (DELETE, NEXT) can take explicit arguments at the option of the user. Each command must be separated from its argument string by a single blank character. This blank delimiter is considered by the Editor to be a part of the command itself, not part of the argument string which follows the command. Thus, the command

```
RETYPE □ /COMMENT )
```

results in the following line:

```
/COMMENT
```

If more than one blank appears between the command and its argument string, all blanks except the first are taken as part of the argument. Thus,

```
FIND □ □ □ /COMMENT )
```

results in a search for the line which begins with the character string

```
□ □ /COMMENT
```

4.1 OPEN (filename (ext)))

The file, the name of which is "filename" and the extension of which is "ext", is searched for on the input device. If a file of this name is not found, a message is printed on the teletype and the mode is changed to Input Mode. An intermediate write file is opened on the output device, and lines from the keyboard are written into it as they are completed. "Ext", if not given, is assumed to be SRC.

If the file specified is found on the input device, it is opened for reading. Subsequent typed lines are interpreted as Editor commands.

Neither file name nor extension need be given if the input device is nonfile-oriented.

4.2 READ ↵

Read sequential lines from the input file, inserting them in the buffer as they are encountered, until the number of lines in the buffer is equal to the argument specified in the SIZE request. The pointer is set to the first line of the buffer when the operation is complete.

The READ request will not be accepted if any lines remain in the current buffer. The buffer must have been cleared by DELETE requests or a WRITE command.

The READ request is treated as illegal if Block Mode is off. The READ request must be used if input device is not file-structured.

4.3 WRITE ↵

Add the current contents of the block buffer to the output file regardless of the position of the pointer within the buffer, and clear the buffer. Nothing is output if the buffer is empty. This command is illegal if Block Mode is OFF.

4.4 CLOSE (filename (ext)) ↵

If an input file is present, all lines in that file falling below the current line are appended to the output file, and the output file is closed. If no input file is present, the current line is added to the output file, and the output file is closed. No further editing is permitted.

If the extension is omitted, and none was assigned in the OPEN command line, the extension is assumed to be SRC. If no filename is given, the name assigned in the OPEN command line is used.

Neither filename nor ext need be given for nonfile-oriented output devices.

4.4.1 ICLOSE ↵

The ICLOSE command effects the closing of the current input file only. The output file remains open. A new input file can be referenced after the ICLOSE request by issuing an OPEN command. ICLOSE provides a facility for combining source files during an editing run.

4.4.2 SCLOSE ↵

This command permits the placement of an edited file onto the current output device without the .DAT-14 to .DAT-15 recopy process. It is particularly useful in closing long files which have only minor changes.

In employing the command SCLOSE, always use a filename different from that used with the OPEN command given for the file. Files closed in this manner are normally left on .DAT-15.

4.5 NEXT [N] (\sqcap n)

The pointer is moved past the next n lines, beginning with the line currently in the work area. Line n + 1 is brought into the work area for modification. Lines skipped over are added to the output file. If omitted, n is assumed to be 1. If the command results in the pointer moving past the last line of the file (or buffer, if Block Mode is on), the following error message is printed:

```
END OF { FILE  
        BUFFER } REACHED BY:  
NEXT n
```

4.6 PRINT [P] (\sqcap n)

This command causes n lines from the input file (or buffer, in Block Mode), including the current line, to be printed on the teletype. The pointer is left at the last line printed; n is assumed to be 1 if omitted.

If, as a result of the command, the pointer moves past the last line of the file, the error message is printed.

```
END OF { FILE  
        BUFFER } REACHED BY:  
PRINT n
```

4.7 FIND [F] \sqcap string

The input file or buffer is searched, beginning with the line following the current line, for the next occurrence of a line which begins with the character group "string." If the search is successful, the line beginning with "string" is brought into the work area. If the search is unsuccessful (pointer moves past end of file), the end-of-file error message is printed.

"String" may contain any number of characters.

4.8 LOCATE [L] \sqcap string

The input file is searched, beginning with the line following the current line, for the next occurrence of a line which contains the character group "string." If the search is successful, the line which satisfies the search is brought to the work area. If the search is unsuccessful, the end-of-file message is printed, and the pointer is moved to the top of the file. "String" may contain any number of characters.

4.9 DELETE [D] (n)

This command causes n lines, including the current line, to be deleted from the input file. The line following the last line deleted becomes the current line. If n is omitted, only the current line is deleted. If n is large enough to cause the pointer to move past the end of the file, the end-of-file error message is printed.

4.10 BOTTOM [B]

The pointer is moved to the final line in the input file (or buffer) which then becomes the current line. Lines skipped over in the process of moving the pointer are added to the output file.

4.11 RETYPE [R] line

The character string "line" replaces the current line. The new line is left in the work area and can be subsequently modified.

4.12 INSERT [I] line

The current line is added to the output file and the character string "line" is taken as the current line. Note that insertions are always made below the current line. The program remains in Edit Mode when command processing is completed.

4.13 INSERT [I]

The current line is added to the output file and the mode is changed from Edit to Input. Subsequent lines are interpreted as text to be added to the output file.

4.14 GET [G] (n)

This command causes n lines from the subsidiary input device to be added to the output file. New lines are added below the current line. When command processing is complete, the nth line read is left in the work area as the current line. If n is omitted, it is assumed to be 1.

If an end-of-medium condition is encountered on the subsidiary input device before n lines are read, the error message

END OF MEDIUM REACHED BY:

GET n

is printed. The pointer remains at the last line read.

4.15 CHANGE [C] \sqsubset q string1q string2q \succ

In the current line, the first character group (string1) which matches that occurring between the first pair of quote characters (q's, in this case) is replaced by the character group (string2) appearing between the second pair of quote characters. The quote character chosen by the user may be any graphic (including blank) which does not appear in either of the character strings quoted. Both string1 and string2 can contain any number of characters, including zero. If Verify Mode is in effect, the program prints the new current line on the teletype when the requested change has been accomplished. Examples of change requests:

| | | | |
|--|-------------------------------------|--------------|------------------|
| Current line: | NXTLIN | JMS TYP OUT | /PRNT THE LINE. |
| a. In the comment, spell "PRINT" properly. | | | |
| Request: | CHANGE \sqsubset /RN/RIN/ \succ | | |
| New line: | NXTLIN | JMS TYP OUT | /PRINT THE LINE. |
| b. Make the "JMS" a "JMP*" | | | |
| Request: | CHANGE \sqsubset XSXP*X \succ | | |
| New line: | NXTLIN | JMP* TYP OUT | /PRINT THE LINE. |
| c. Delete the "T" in the tag. | | | |
| Request: | C \sqsubset /T// \succ | | |
| New line: | NXLIN | JMP* TYP OUT | /PRINT THE LINE. |

4.16 TOP [T] \succ

Move the pointer to the beginning of the edited file or buffer. The first line of the file becomes the current line.

4.17 VERIFY [V] \sqsubset $\begin{Bmatrix} \text{ON} \\ \text{OFF} \end{Bmatrix}$ \succ

Set Verify Mode according to the parameter. When Verify Mode is on, text lines are printed in response to certain editing commands, for example:

- The line brought into the work area as a result of a FIND or LOCATE request is printed.
- The last line of the file, brought in by the BOTTOM request, is printed.
- The new line resulting from a CHANGE request is printed.

When Verify Mode is off, only error messages are printed. After the Editor is loaded initially, Verify Mode is on.

The command

VERIFY [V])

(without arguments) is equivalent to

VERIFY [V] _ ON)

4.18 OVERLAY [O] (_ n)

Starting with the current line, n lines (or the current line only, if n is omitted) are deleted from the input file. Control mode is changed to Input with the normal typed program response,

INPUT

Subsequent typed lines are interpreted as text intended to replace the lines so OVERLAYed.

4.19 APPEND [A] _ string)

"String" is added to the current line following the last data character and preceding the terminating carriage return. Thus, to add a comment to the current line

JMS GETNUM

the command might be

APPEND _ → /GET DECIMAL ARGUMENT.)

The new current line would be

JMS GETNUM → /GET DECIMAL ARGUMENT.

If "string" is absent, the current line is unchanged.

4.20 BRIEF _ { ON } { OFF }

Set Brief Mode according to the ON/OFF parameter. Brief Mode results in the abbreviated printing of the current line during the servicing of some commands. An attempt is made to print only the tag, operation code, and address fields of lines brought in as a result of the FIND, LOCATE, and BOTTOM commands. In addition, the printing of the new line resulting from a CHANGE request is terminated at the last newly-inserted character.

Brief Mode is set to OFF initially. The setting of the brief mode indicator is of no consequence when Verify Mode is off.

The command

BRIEF ↵

(without arguments) is equivalent to

BRIEF ␣ ON ↵

4.21 BLOCK ␣ $\begin{cases} \text{ON} \\ \text{OFF} \end{cases}$ ↵

Set Block Mode according to the parameter. When Block Mode is ON, the editing commands READ, WRITE, and MOVE are accepted by the program; these commands are treated as illegal if Block Mode is off. When Block Mode is in effect, the program treats several lines as a subfile, retaining them internally in a block buffer. In Block Mode, editing commands which move the pointer reference only those lines currently residing in the buffer. The contents of the buffer are saved until a WRITE command is encountered or until, by way of the DELETE command, it is emptied. A buffer emptied by deletions can be filled by a READ request.

When Block Mode is OFF, sequential lines in the input file are moved singly to the word area and are not available for re-examination after the pointer has been moved to a later line.

When the Editor is initially loaded, Block Mode is set to ON if either the input or the scratch device is nonfile-oriented. If both devices are file-oriented, Block Mode is set OFF.

The command

BLOCK ↵

(without arguments) is equivalent to

BLOCK ␣ ON ↵

4.22 SIZE [S] ␣ n ↵

Set the total number of lines which will occupy a buffer (in Block Mode) to n. The SIZE command can be issued at any time and takes effect when the next group of lines is inserted in the buffer via a READ command. The value of n is initially set to 55₁₀; it must always be greater than 1.

4.23 EXIT ↵

Control is transferred from the Editor to the Monitor. This command is illegal if any file is open for reading or writing when it is issued, i.e., it may only be given as the first command after Editor initialization and the message

EDITOR
>

4.24 OUTPUT $\sqcup \begin{Bmatrix} \text{ON} \\ \text{OFF} \end{Bmatrix} \rhd$

After the Editor is loaded into core, in either I/O or Keyboard Monitor environments, OUTPUT is initially set to ON. If OUTPUT is set OFF, the user is allowed to examine any part of his program. No output will result after a WRITE or CLOSE command is issued and the NOTHING IN FILE message is typed out. If the input is file-oriented, the input remains unchanged.

4.25 CALL \sqcup RENAME $\sqcup \begin{Bmatrix} \text{INPUT} \\ \text{OUTPUT} \end{Bmatrix} \sqcup \text{OLDNAM} \sqcup \text{EXT} \sqcup \text{NEWNAM} \sqcup \text{EXT} \rhd$

This command can only be used before any other commands are issued. No abbreviation is allowed in the command string.

Example:

```
EDITOR V6A
>CALL  $\sqcup$  RENAME  $\sqcup$  INPUT  $\sqcup$  OLDNAM  $\sqcup$  SRC  $\sqcup$  NEWNAM  $\sqcup$  SRC  $\rhd$ 
will change the name of the file on the input device (.DAT slot-14) from
OLDNAM SRC to NEWNAM SRC.
```

4.26 CALL \sqcup DELETE $\sqcup \begin{Bmatrix} \text{INPUT} \\ \text{OUTPUT} \end{Bmatrix} \sqcup \text{FILNAM} \sqcup \text{EXT} \rhd$

This command can only be used before any other commands are issued. No abbreviation is allowed in the command string.

Example:

```
EDITOR V6A
>CALL  $\sqcup$  DELETE  $\sqcup$  OUTPUT  $\sqcup$  FILNAM  $\sqcup$  SRC  $\rhd$ 
will delete the file FILNAM SRC from the directory on the output device
(.DAT SLOT-15).
```

4.27 RENEW \rhd

This command initiates the same operations as those performed by a WRITE command followed immediately by a READ command. The use of this command is permitted only in Block Mode.

4.28 KEEP \rhd

This command causes the original file appearing on .DAT-14 to be preserved for back-up purposes. The form of this command is:

KEEP savnam (ext))

where "savnam" is a unique name to be assigned to the file to be preserved and "ext" is its extension. If ext is not given, SRC is assumed. As many KEEP requests as are needed can be issued. Each savnam, however, must be different from any other filename appearing on -14. If a unique name is not given to the file to be saved, the name SAVFIL EDT is assigned. The KEEP command can be issued only while the input file is on .DAT-14, i.e., the number of TOPs issued must be even (0,2,4...).

Example of Usage:

| | |
|---------------|-----------------------------|
| EDITOR V6A | |
| > OPEN THSFIL | |
| EDIT | |
| > KEEP BKUP01 | |
| . | |
| > TOP | /Normal TOP |
| > NEXT | /Some locative request must |
| . | be issued here to allow |
| . | physical file transfer at |
| . | second TOP. |
| > TOP | /To return file to -14. |
| > KEEP BKUP02 | |
| > CLOSE | |

Resulting Directory on -14:

| | |
|------------|--------------------------|
| THSFIL SRC | /New edited file |
| BKUP01 SRC | /First copy--original |
| BKUP02 SRC | /Second copy--after TOPs |

SECTION 5

RECOVERY PROCEDURES

In case of a hardware or system failure, the user can recover at the point at which the last complete version of the edited output file was closed. The Editor, in preparing intermediate files, assigns them temporary names. Thus, in the event of disaster, one (or both) of the following files may be found.

.TFIL1 EDT and .TFIL2 EDT both contain the version of the edited file extant at the point at which the crash occurred. No editing is lost. If neither of these files is present, the file specified in the OPEN command contains the version of the file extant at the time the latest TOP command was issued. All editing taking place after the TOP command is lost. If neither .TFIL1 EDT nor .TFIL2 EDT is found and if no file name was given in the OPEN command, no recovery is possible.

SECTION 6

EXAMPLES OF EDITING REQUESTS

This section contains illustrations of one complete iteration through the modification process using the Editor.

Figure 6-1 shows the assembly listing of a sample input file.

Figure 6-2 shows the same listing marked for correction.

Figure 6-3 (Sheet 1, Sheet 2) shows the hard-copy output of the editing session. The sequence numbers at the right margin are not program generated, but were added later for reference.

Figure 6-4 is the assembly listing of the new, edited file showing the results of the editing run.

```

                                /SUBROUTINE PACK, 7-BIT CHARS TO IOPS ASCII.
                                /CALL: JMS PACK
                                /      FROM
                                /      TO
                                /      PACK 0
                                LAC PACK          /GET FROM ADDRESS.
                                DAC PFROM        /GIVE TO FROM POINTER.
                                LAC* PACK        /GET ADDRESS OF TO ARRAY.
                                DAC PTO         /GIVE TO OUTPUT POINTER.
                                DAC PLRH        /SAVE AS START ADDRESS.
                                ISZ PACK        /BUMP TO RETURN.
                                PL00P1 LAW 17773 /SET UP
                                DAC PK5CHR
                                PL00P2 LAC* PFROM /GET NEXT WORD IN INPUT ARRAY.
                                SAC (-1         /TERMINATOR?
                                SKP            /NO, SKIP.
                                JMP PCLOS      /YES, GO CLOSE OUTPUT ARRAY.
                                ISZ PFROM      /POINT TO NEXT WORD.
                                DAC PWRD3     /POINT TO NEXT WORD.
                                PL00P7 JMS PRL7  /SET UP TO ROTATE.
                                ISZ PK5CR      /5 CHARS IN,0
                                JMP LP00P2     /NO, GET ANOTHER.
                                LAC PWRD2     /WORD PAIR COMPLETE.
                                RAL          /CLEAR PAIR BIT 35.
                                DAC PWRD2
                                LAC PWRD1
                                RAL:CLL      /GET FIRST WD OF PAR..
                                DAC* PTO     /BIT 0 OF WD 2.
                                ISZ PTO      /INSERT FIRST WD IN OUT ARRAY.
                                JMP PL00P1   /BUMP OUT ADDRESS.
                                LAW 17773    /GO SET UP NEXT PAIR.
                                SAD:PK5CHR   /MAKE SURE PAIR IS COMPLETE.
                                JMP PL00P7
                                CLA:CMA     /INCOMPLETE PAIR.
                                TAD PLRH    /FORM WORD PAIR COUNT
                                CMA        /START ADDRESS.
                                TAD PTO     /LESS END ADDRESS.
                                JMP* PACK   /RETURN TO CALLER.
                                .END
00000 R 000000 A
00001 R 200000 R
J 00002 R 040044 R
J 00003 R 220000 R
U 00004 R 040052 R
J 00005 R 040047 R
J 00006 R 440000 R
J 00007 R 777773 A PL00P1
J 00010 R 040045 R
J 00011 R 220044 R PL00P2
LU 00012 R 000000 R
00013 P 741000 A
J 00014 R 600033 R
J 00015 R 440044 R
J 00016 R 440044 R
J 00017 R 040055 R
J 00020 R 100051 R PL00P7
U 00021 P 440046 R
U 00022 P 600043 R
J 00023 R 200054 R
J 00024 P 740010 A
J 00025 R 040054 R
U 00026 R 200053 R
J 00027 P 744010 A
J 00030 P 060052 R
U 00031 R 440052 R
U 00032 P 600050 R
J 00033 R 777773 A PCLOS
J 00034 P 540045 R
J 00035 R 600020 R
J 00036 R 750001 A
U 00037 R 340047 R
J 00040 R 740001 A
J 00041 R 340052 R
J 00042 R 620000 R
00057 R 777777 A *LIT

```

Figure 6-1 Sample Input File

LEFT-ADJUSTED NON-HEADERED

.GLOBAL PACK, PRACT, PWRD1, PWRD2, PWRD3

/SUBROUTINE PACK, 7-BIT CHARS TO IOPS ASCII.

/CALL: JMS PACK

FROM /START OF INPUT ARRAY.

TO /START OF OUTPUT ARRAY.

ON RETURN, AC HOLDS TOTAL WORDS OCCUPIED BY PACKED ARRAY. A WORD OF ALL 1'S MUST TERMINATE THE INPUT (UNPACKED) ARRAY.

| | | | | | | | |
|----------|---|--------|---|---------------|-------------------|--|--------------------------------|
| 00000 | R | 000000 | A | PACK | 0 | | |
| 00001 | R | 200000 | R | LAC* | LAC PACK | | /GET FROM ADDRESS. |
| U 00002 | R | 040044 | R | ISZ PACK | DAC PFROM | | /GIVE TO FROM POINTER. |
| 00003 | P | 220000 | R | | LAC* PACK | | /GET ADDRESS OF TO ARRAY. |
| U 00004 | R | 040052 | R | | DAC PTO | | /GIVE TO OUTPUT POINTER. |
| U 00005 | R | 040047 | R | | DAC PLBH | | /SAVE AS START ADDRESS. |
| 00006 | P | 440000 | R | | ISZ PACK | | /BUMP TO RETURN. |
| 00007 | R | 777773 | A | PLOOP1 | LAW 17773 | | /SET UP |
| 00010 | R | 040045 | R | | DAC PK5CHR | | 15-CHARACTER COUNTER. |
| U 00011 | R | 220044 | R | PLOOP2 | LAC* PFROM | | /GET NEXT WORD IN INPUT ARRAY. |
| LU 00012 | R | 000000 | R | SAD | SAC LIT (ENDCHR | | /TERMINATOR? |
| 00013 | R | 741000 | A | | SKP | | /NO, SKIP. |
| 00014 | R | 600033 | R | | JMP PCLOS | | /YES, GO CLOSE OUTPUT ARRAY. |
| U 00015 | R | 440044 | R | | ISZ PFROM | | /POINT TO NEXT WORD. |
| U 00016 | R | 440044 | R | | ISZ PFROM | | /POINT TO NEXT WORD. |
| U 00017 | P | 040055 | R | | DAC PWRD3 | | /SET UP TO ROTATE. |
| U 00020 | R | 100051 | R | PLOOP7 | JMS PRAL7 | | |
| U 00021 | R | 440046 | R | | ISZ PK5CHR | | 15 CHARS IN (0)? |
| U 00022 | R | 600043 | R | | JMP PLOOP2 PLOOP2 | | /NO, GET ANOTHER. |
| U 00023 | R | 200054 | R | | LAC PWRD2 | | /WORD PAIR COMPLETE. |
| 00024 | R | 740010 | A | RAL:CLL | RAL | | /CLEAR PAIR BIT 35. |
| U 00025 | R | 040054 | R | | DAC PWRD2 | | |
| U 00026 | R | 200053 | R | | LAC PWRD1 | | /GET FIRST WD OF PAIR. (IR) |
| 00027 | R | 744010 | A | RAL | RAL:CLL | | /BIT 0 OF WD 2. |
| U 00030 | R | 060052 | R | | DAC* PTO | | /INSERT FIRST WD IN OUT ARRAY. |
| U 00031 | R | 440052 | R | | ISZ PTO | | /BUMP OUT ADDRESS. |
| U 00032 | R | 600050 | R | | JMP PLOOP1 PLOOP1 | | /GO SET UP NEXT PAIR. |
| 00033 | R | 777773 | A | PCLOS | LAW 17773 | | /MAKE SURE PAIR IS COMPLETE. |
| U 00034 | R | 540045 | R | DEM PWRD3 | SAD PK5CHR | | |
| 00035 | R | 600020 | R | | JMP PLOOP7 | | /INCOMPLETE PAIR. |
| 00036 | R | 750001 | A | ENDCHR LAW -1 | CLA:CMR | | /FORM WORD PAIR COUNT |
| U 00037 | R | 340047 | R | TAD*PACK | TAD PLBH | | /START ADDRESS. |
| 00040 | R | 740001 | A | | CMA | | |
| U 00041 | R | 340052 | R | ISZ PACK | TAD PTO | | /LESS END ADDRESS. |
| 00042 | R | 620000 | R | | JMP* PACK | | /RETURN TO CALLER. |
| | | 000000 | A | | .END | | |
| 00057 | R | 777777 | A | *LIT | | | |

PFROM \emptyset
 PTO \emptyset
 PK5CHR \emptyset

Figure 6-2 Input File Listing Marked for Correction

| | | |
|---|-------------------------|----|
| EDITOR | | 1 |
| >OPEN PACK SAC | | 2 |
| >FIND /SUBROUT | | 3 |
| /SUBROUTINE PACK, 7-BIT CHARS TO IOPS ASCII. | | 4 |
| >OVERLAY 1 | | 5 |
| INPUT | | 6 |
| /SUBROUTINE PACK, 7-BIT LEFT-ADJUSTED CHARS TO NON-HEADERED IOPS | | 7 |
| /ASCII. ON RETURN, AC HOLDS TOTAL WORDS OCCUPIED BY PACKED ARRAY. | | 8 |
| /A WORD OF ALL 1'S MUST TERMINATE THE INPUT (UNPACKED) ARRAY. | | 9 |
| EDIT | | 10 |
| >LOCATE FROM | | 11 |
| / FROM | | 12 |
| >APPEND | /START OF INPUT ARRAY. | 13 |
| >NEXT | | 14 |
| >APPEND | /START OF OUTPUT ARRAY. | 15 |
| NOT A REQUEST: | | 16 |
| APPEND | /START OF OUTPUT ARRAY. | 17 |
| >APPEND | /START OF OUTPUT ARRAY. | 18 |
| >PRINT 1 | | 19 |
| / TO | /START OF OUTPUT ARRAY. | 20 |
| >INSERT .GLOBL PACK, PRAL7, PWRD1, PWRD2, PWRD3 | | 21 |
| >L LAC | | 22 |
| LAC PACK | /GET FROM ADDRESS. | 23 |
| >CHANGE LAC/LAC*/ | | 24 |
| LAC PACK | /GET FROM ADDRESS. | 25 |
| >CHANGE /LAC/LAC*/ | | 26 |
| LAC* PACK | /GET FROM ADDRESS. | 27 |
| >NEXT 1 | | 28 |
| >INSERT | | 29 |
| INPUT | | 30 |
| ISZ PACK | / BUMP TO "TO" ADDRESS. | 31 |
| EDIT | | 32 |
| >PRINT | | 33 |
| ISZ PACK | / BUMP TO "TO" ADDRESS. | 34 |
| >BRIEF ON | | 35 |
| >C ./ ./. | | 36 |
| ISZ PACK | / | 37 |
| >PRINT | | 38 |
| ISZ PACK | /BUMP TO "TO" ADDRESS. | 39 |
| >L PLBH | | 40 |
| DAC PLBH | | 41 |
| >BRIEF OFF | | 42 |
| >PRINT | | 43 |
| DAC PLBH | /SAVE AS START ADDRESS. | 44 |
| >DELETE 2 | | 45 |
| >PRINT | | 46 |
| PLOOP1 LAW 17773 | /SET UP | 47 |
| >N 1 | | 48 |
| >A | /5-CHARACTER COUNTER. | 49 |
| >L (| | 50 |
| SAC (-1 | /TERMINATOR? | 51 |
| >VERIFY OFF | | 52 |
| >C /SAC/SAD/ | | 53 |
| >C /(-1/ENDCHR/ | | 54 |
| >V ON | | 55 |
| >P | | 56 |
| SAD ENDCHR | /TERMINATOR? | 57 |
| >N | | 58 |
| >D | | 59 |
| >N | | 60 |
| >P | | 61 |
| ISZ PFROM | /POINT TO NEXT WORD. | 62 |
| | | 63 |
| | | 64 |

Figure 6-3 (Sheet 1) Hard-Copy Output of Editing Session

| | | | |
|------------------|------------|-------------------------|-----|
| >D | | | 65 |
| >P | | | 66 |
| | ISZ PFROM | /POINT TO NEXT WORD. | 67 |
| >F PL | | | 68 |
| PLOOP7 | JMS PRAL7 | | 69 |
| >N | | | 70 |
| >RETYPE | ISZ PK5CHR | /5 CHARS IN? | 71 |
| >N | | | 72 |
| >C ,LP,PL, | | | 73 |
| | JMP PLOOP2 | /NO, GET ANOTHER. | 74 |
| >N 2 | | | 75 |
| >CHANGE /L/LICLL | | | 76 |
| | RALICLL | /CLEAR PAIR BIT 35. | 77 |
| >L RAL | | | 78 |
| | RALICLL | /BIT 0 OF WD 2. | 79 |
| >C /ICLL// | | | 80 |
| | RAL | /BIT 0 OF WD 2. | 81 |
| >L JMP | | | 82 |
| | JMP PL00P1 | /GO SET UP NEXT PAIR. | 83 |
| >CHANGE /00/00/ | | | 84 |
| | JMP PLOOP1 | /GO SET UP NEXT PAIR. | 85 |
| >N | | | 86 |
| > | | | 87 |
| INPUT | | | 88 |
| | DZM PWRD3 | /FILL PAIR WITH ZEROES. | 89 |
| EDIT | | | 90 |
| >L ! | | | 91 |
| | CLA!CMA | /FORM WORD PAIR COUNT | 92 |
| >R ENDCHR | LAW -1 | /FORM WORD PAIR COUNT. | 93 |
| >N | | | 94 |
| >R | TAD* PACK | /START ADDRESS. | 95 |
| >L PTO | | | 96 |
| | TAD PTO | /LESS END ADDRESS. | 97 |
| >INSERT | ISZ PACK | | 98 |
| >BOTTOM | | | 99 |
| | .END | | 100 |
| >OVERLAY | | | 101 |
| INPUT | | | 102 |
| PFROM | 0 | | 103 |
| PRO0PTO | 0 | | 104 |
| PK5CHR | 0 | | 105 |
| | .END | | 106 |
| EDIT | | | 107 |
| >TOP | | | 108 |
| >L ADDRESS | | | 109 |
| | LAC* PACK | /GET FROM ADDRESS. | 110 |
| >C /ADR/ADDR/ | | | 111 |
| | LAC* PACK | /GET FROM ADDRESS. | 112 |
| >LOCATE .. | | | 113 |
| | LAC PWRD1 | /GET FIRST WD OF PAR.. | 114 |
| >V OFF | | | 115 |
| >C /R./IR/ | | | 116 |
| >PRINT | | | 117 |
| | LAC PWRD1 | /GET FIRST WD OF PAIR. | 118 |
| >CLOSE | | | 119 |
| | | | 120 |
| EDITOR | | | 121 |
| >EXIT | | | 122 |
| | | | 123 |
| MONITOR | | | 124 |
| | | | 125 |
| | | | 126 |
| | | | 127 |
| \$ | | | 128 |

Figure 6-3 (Sheet 2) Hard-Copy Output of Editing Session

```

00000 R 000000 A
00001 R 220000 R
00002 R 040042 R
00003 R 440000 R
00004 R 220000 R
00005 R 040043 R
00006 R 777773 A
00007 R 040044 R
00010 R 220042 R
00011 R 540034 R
00012 R 600030 R
00013 R 440042 R
00014 R 340050 V
00015 R 100045 V
00016 R 440044 R
00017 R 600010 R
00020 R 200047 V
00021 R 744010 A
00022 R 040047 V
00023 R 200046 V
00024 R 740010 A
00025 R 060043 R
00026 R 440043 R
00027 R 600006 R
00030 R 777773 A
00031 R 140050 V
00032 R 540044 R
00033 R 600015 R
00034 R 777777 A
00035 R 360000 R
00036 R 740001 A
00037 R 340043 R
00040 R 440000 R
00041 R 620000 R
00042 R 000000 A
00043 R 000000 A
00044 R 000000 A
000000 A

/SUBROUTINE PACK, 7-BIT LEFT-ADJUSTED CHARS TO NON-HEADERED IOPS
/ASCII, ON RETURN, AC HOLDS TOTAL WORDS OCCUPIED BY PACKED ARRAY.
/A WORD OF ALL 1'S MUST TERMINATE THE INPUT (UNPACKED) ARRAY.
/CALL: JMS PACK
/ FROM /START OF INPUT ARRAY.
/ TO /START OF OUTPUT ARRAY.
/ .GLOBL PACK, PRAL7, PWRD1, PWRD2, PWRD3
PACK 0
LAC* PACK /GET FROM ADDRESS.
DAC PFROM /GIVE TO FROM POINTER.
ISZ PACK /BUMP TO "TO" ADDRESS.
LAC* PACK /GET ADDRESS OF TO ARRAY.
DAC PTO /GIVE TO OUTPUT POINTER.
PLOOP1 LAW 17773 /SET UP
DAC PK5CHR /5-CHARACTER COUNTER.
PLOOP2 LAC* PFROM /GET NEXT WORD IN INPUT ARRAY.
SAD FNDCHR /TERMINATOR?
JMP PCLOS /YES, GO CLOSE OUTPUT ARRAY.
ISZ PFROM /POINT TO NEXT WORD.
DAC PWRD3 /SET UP TO ROTATE.
PLOOP7 JMS PRAL7
ISZ PK5CHR /5 CHARS IN?
JMP PLOOP2 /NO, GET ANOTHER.
LAC PWRD2 /WORD PAIR COMPLETE.
RAL:CLL /CLEAR PAIR BIT 35.
DAC PWRD2
LAC PWRD1 /GET FIRST WD OF PAIR.
RAL /BIT 0 OF WD 2.
DAC* PTO /INSERT FIRST WD IN OUT ARRAY.
ISZ PTO /BUMP OUT ADDRESS.
JMP PLOOP1 /GO SET UP NEXT PAIR.
PCLOS LAW 17773 /MAKE SURE PAIR IS COMPLETE.
NZM PWRD3 /FILL PAIR WITH ZEROES.
SAD PK5CHR
JMP PLOOP7 /INCOMPLETE PAIR.
LAW -1 /FORM WORD PAIR COUNT.
FNDCHR TAD* PACK /START ADDRESS.
CMA
TAD PTO /LESS END ADDRESS.
ISZ PACK
JMP* PACK /RETURN TO CALLER.
PFROM 0
PTO 0
PK5CHR 0
.END

```

Figure 6-4 File Resulting From Editing Session

APPENDIX A
SUMMARY OF EDITING COMMANDS

| Command | Abbreviation | Activity | Line Number [†] | Section |
|--|--------------|---|--------------------------|---------|
| Editor-Monitor Communication | | | | |
| EXIT | n/a | Transfer control to Monitor | 124 | 4.23 |
| File Housekeeping | | | | |
| OPEN nm ext | n/a | Prepare input file (named "nm ext") for editing. | 2 | 4.1 |
| CLOSE | n/a | Terminate editing on input file. | 121 | 4.4 |
| ICLOSE | n/a | Close input file. | | |
| SCLOSE | n/a | Close file and leave on output device. | | |
| Locative Requests | | | | |
| FIND string | F | Bring first line beginning with "string" to work area. | 3,68 | 4.7 |
| LOCATE string | L | Bring first line containing "string" to work area. | 12,52 | 4.8 |
| NEXT | N | Bring next consecutive line to work area. | 15,70 | 4.5 |
| BOTTOM | B | Bring last line of file to work area. | 100 | 4.10 |
| TOP | T | Reset pointer to beginning of file. | 110 | 4.6 |
| PRINT (x) | P | Print the current line on the Teletype. 2-20 OF LINES | 20,58 | 4.6 |
| Manipulative Requests | | | | |
| DELETE | D | Discard the current line. | 47,61 | 4.9 |
| RETYPE string | R | Replace current line with "string". | 71,94 | 4.11 |
| INSERT string | I | Add "string", as a complete line, to the file after (below) the current line. | 99 | 4.12 |
| CHANGE /string1/ string2/ | C | Replace, in the current line, the first occurrence of "string1" with "string2". | 25,27,38 | 4.15 |
| OVERLAY | O | Replace multiple lines. | 5,102 | 4.18 |
| APPEND string | A | Add "string" at the rightmost end of the current line. | 14,16,19 | 4.19 |
| [†] Entries under "Line Number" refer to line sequence numbers (in Figure 6-3) where examples of command usage are to be found. | | | | |

| Command | Abbreviation | Activity | Line Number† | Section |
|--|--------------|--|--------------|---------|
| Mode Control | | | | |
| VERIFY { ON OFF | V | Set verify mode to print (ON) or ignore printing (OFF) lines after processing CHANGE, LOCATE, and FIND requests. | 54,57 | 4.17 |
| BLOCK { ON OFF | n/a | Set program to operate in block mode (ON) or in line-by-line mode (OFF). | | 4.21 |
| BRIEF { ON OFF | n/a | Set brief mode to print truncated (ON) or full (OFF) lines. | 37,44 | 4.20 |
| Input/Output Requests | | | | |
| READ | n/a | Fill block buffer from input file. | | 4.2 |
| WRITE | n/a | Add block buffer to output file. | | 4.3 |
| GET | G | Add lines from subsidiary input device after (below) current line. | | 4.14 |
| Miscellaneous Requests | | | | |
| SIZE | S | Set total lines to occupy block buffer. | | 4.22 |
| INSERT | I | Change mode to input. | 30 | 4.13 |
| †Entries under "Line Number" refer to line sequence numbers (in Figure 6-3) where examples of command usage are to be found. | | | | |

APPENDIX B

339 DISPLAY EDITOR

This Appendix describes a version of the Text Editor which uses the 339 Display to show the text being edited. The Display Editor is a relocatable user program (as opposed to a system program) which is loaded and used according to the directions contained. It is distributed by DEC's Program Library as a binary paper tape and should be PIPed to the system DECtape before use.

SYSTEM REQUIREMENTS

This program requires a 339 Display equipped with either the VA39 or the VC38 optional character generator. If the VC38 (software) is used, the Monitor switch VC38 must be ON. In any case, the Monitor switch 339 must be ON. Both switches can be set normally ON at System Generation. The 339 device handler DYA. must be assigned to .DAT slot + 10. All other device assignments are the same as for the System Editor.

Loading Procedure

```
$ GLOAD
LOADER Vxx
> EDITDY (User types underscored text.)
  EDITDY  xxxxx
  DYA.    xxxxx
  other device handlers
> EDITDY V6A
```

PROGRAM OPERATION

The Display Editor consists of the System Editor (EDIT6A) with the additional ability of displaying the text being edited on the 339 Display assigned to .DAT slot + 10. Display is controlled through use of the TV switch. The TV switch is initially off, and the Display Editor operates exactly like the System Editor. If >TV ON is typed, the text file is displayed as follows, from top to bottom:

- a. a group of the lines most recently added to the output file or block;
- b. spaces;
- c. the current line, to which modifications apply;
- d. spaces;
- e. a group of the lines about to be brought in from the input file or block.

TV can be turned on or off at any time to start or stop the display. Turning TV on implies VERIFY OFF, but turning it off does not turn VERIFY ON automatically. If, at the top of a file or block (i.e., OPEN filename or TOP was just typed), TV ON is typed, a file movement command (e.g., NEXT, FIND, LOCATE, etc.) must be given before any lines are displayed.

In Block Mode, the block buffer must be at least as large as the read-ahead buffer used for displaying lines below the current line. Thus, if a SIZE command is given with too small a number of lines, the error message

BUFFER SIZE TOO SMALL.

is typed, and the SIZE command is disregarded.

In all other respects, operation of the Display Editor is the same as the System Editor.

SECTION 1
INTRODUCTION

1.1 GENERAL INFORMATION

The PATCH utility program[†] provides the user with a convenient means of examining and modifying system programs which are stored in binary form on a bulk storage device (DECtape or disk).

Relocatable programs (link-loadable programs), XCT programs (executable files built by the system program CHAIN), and any other binary program which is not in system program format cannot readily be corrected by using PATCH. System program format means that the binary is a straight core dump onto contiguous blocks of 400 (octal) words each on the bulk storage I/O device.

Normally, commands to PATCH are issued at the teletype keyboard. In BATCH processing, commands are taken from the batch input device under supervision of the BATCH processor. Binary corrections can be specified by the input commands or read in from an auxiliary input device. The latter facility provides a way to transfer new binaries of system programs to a system tape.

[†]The PATCH program in PDP-9 Systems replaces the stand-alone programs SYSTEM and MONITOR.

SECTION 2 CALLING PATCH

2.1 .DAT SLOT ASSIGNMENTS

Before requesting the Monitor to load PATCH into core, check that the proper device assignments have been made to the .DAT slots which PATCH uses.

| <u>.DAT SLOT</u> | <u>Used to</u> |
|------------------|--|
| -14 | Input from and output to the bulk storage device on which patches are to be made. The device handler is required only to perform .TRAN. |
| -10 | Input from the auxiliary device, which may be a bulk storage device. The device handler must handle Dump Mode input and, if it is for a nonfile-oriented device, must handle image alpha mode. |
| -3 | Output to the teletype. |
| -2 | Input from the teletype or batch processing device. |

.DAT slot-10 can be assigned no device handler (NONE) if auxiliary input is not required. .DAT slots -3 and -2 cannot be changed.

2.2 CALLING PATCH

PATCH is loaded and started by the System Loader after the user issues the following command (underscored) to the Monitor:

```
MONITOR  
$PATCH >
```

When PATCH is running it prints:

```
PATCH xxx  
>
```

where xxx is a three-character program version designation, and the right angle bracket (>) indicates that the program is ready to accept a command input.

SECTION 3 COMMANDS

3.1 COMMAND INPUT FORMAT

All teletype and batch processing input commands are standard IOPS ASCII lines, that is, lines of text terminated by either a carriage return (↵) or an ALTMODE character. While typing in a command, but prior to terminating with a carriage return or ALTMODE, the user can modify his input by typing control U (echoed as @) to cancel the entire line or typing N rubout characters (each echoed as \) to delete the last N characters in the line. In all cases, PATCH is idle after it has printed a right angle bracket (>) and is waiting for typed input.

3.2 COMMANDS

PATCH commands are divided into four categories:

- a. A select command to indicate which program or section of the tape is to be patched
- b. A list command that allows examination and modification of specified program locations
- c. A read command to input patches from the auxiliary input device
- d. An exit command which returns control to the Monitor.

3.2.1 Select Commands

Before the user can issue a list or read command, he must indicate the program or block he wishes to patch. This is accomplished by giving either the name of the program or the number of a logical block on the I/O device to be made "current."

3.2.1.1 Select Command Format Examples - Three basic select command formats are shown below.

| | |
|-----------|---|
| Format 1: | > NAME ↵ |
| Format 2: | > B <u> </u> n ↵ or > B+ <u> </u> n ↵ |
| Format 3: | > B- <u> </u> n ↵ |

Format 1 - In this format, NAME stands for the name of the system program to be selected. PATCH contains a table of system program names and, for each, the load address in core, the program's size, and the logical

block number where the program begins on the system tape. The following program names are recognized by PATCH (DDT, CHAIN and the Linking Loader are excluded because they are relocatable binary files):

| | |
|---------|--------|
| CONV | MACROA |
| DUMP | PATCH |
| EDIT | PIP |
| EXECUTE | .SGEN1 |
| F4 | .SGEN2 |
| F4A | .SYSLD |
| KM9 | UPDATE |
| MACRO | |

Formats 2 and 3 - In these formats, the character n stands for an octal number (in the range $0 \leq n \leq 1101_8$) which identifies a logical block on the patch I/O device, for example, DECtape block 100. The two forms shown for Format 2 are equivalent and indicate that the block is to be read and written in the forward direction. Format 3, which is used only for DECtape, means that the block is to be read and written in the reverse direction. Selecting a single block is similar (in effect) to selecting a system program. The load address is set equal to 0, the size is set to 400 octal, and the block number is as specified.

If the NAME in Format 1 or the block number n in Formats 2 and 3 is followed by a space, the remainder of the input line is treated as a comment and is ignored.

Until another select command is issued, all patches made with list and read commands are made to the currently selected program or block.

3.2.2 The List Command

The list command allows the user to list, selectively, the locations within a program or block along with their contents and, optionally, to modify their contents.

The command format is:

```
>L _ OCTADR
```

where OCTADR must be an octal number within the core range specified by the load address and size of the current program or block to be patched. For block patching the range must be $0 \leq \text{OCTADR} \leq 377_8$. For system programs, the range is

$$\text{load address} \leq \text{OCTADR} \leq \text{load address} + \text{size} - 1.$$

The octal address may be followed by a space and a comment (which is ignored).

PATCH determines in which block on the tape the given address is located; and, if that block is not currently in core, it reads in the block. PATCH then prints the address and its contents and waits for command input.

Example:

```
> L 132 ↵
132 00132/777435>
```

The user can now modify the contents of this register and then either list another address or terminate the list command sequence.

3.2.2.1 Modification of Open Location Register – To modify the contents of the open register, type in an expression (defined below) and terminate the expression with any of the following characters:

← or / or ↵ or ALTMODE

Use of Terminators – The value of the expression is stored in the opened location, and subsequent action is determined by the above listed terminators. If no expression is typed, the contents of the open register are not changed, but the terminators still take effect. The manner in which each terminator is interpreted is as follows:

| <u>Terminator</u> | <u>Meaning</u> |
|-------------------|---|
| ← | Treat the remainder of the line as a comment. Take the 13-bit address part of the contents of the current register (as possibly modified) and list that address and its contents. |

Example

```
> L 100 ↵
100 00100/600200> ← ↵ /NOTE: LOCATION 100 UNCHANGED
100 00200/213775>
```

| <u>Terminator</u> | <u>Meaning</u> |
|-------------------|---|
| / | Treat the remainder of the line as a comment. Subsequent action depends on whether the line is terminated by carriage return or by ALTMODE. |
| ↵ | Open the next sequential register. |

Example

```
> L 1266 ↵
1266 01266/000000> ↵
1267 01267/111215> ↵
1270 01270/403620>
```

| <u>Terminator</u> | <u>Meaning</u> |
|-------------------|--------------------------------------|
| ALTMODE | Terminate the list command sequence. |

Example

```
>L _ 3
_ 00003/000110>ALTMODE
>
```

NOTE

A list command sequence must be terminated by ALTMODE; otherwise, select, list, read, or exit commands cannot be given.

When PATCH prints an address and its contents, the first character in the line is either a space () or a right angle bracket (>). The > is used whenever a new block is read into core. If the old block was modified, it is written out on the patch output device before the new block is read in. The device must be WRITE ENABLED.

Definition of Expression - An expression consists of octal numbers (one to six digits) and alphanumeric symbols (one to three characters, the first of which cannot be an octal digit) separated by one operator or a string of operators. An expression is terminated by one of the following characters:

+ , > , / , ALTMODE.

Leading and trailing operators are legal, but the latter are ignored. Whenever a string of consecutive operators is encountered, only the last one is saved. PATCH contains a symbol table of all the basic system opcodes plus all the operate group instructions and the octal values of each.

The expression is evaluated from left to right assuming an initial value of zero followed by the operator +, that is, 0 + user's expression.

The following characters are operators which combine the values to their left and right as indicated:

| | |
|---------|---------------------------|
| - | two's complement subtract |
| ! | inclusive or |
| + | } two's complement add |
| _ | |
| → (TAB) | |
| * | |

The character *, in addition to its use as an operator, causes 20000₈ (the indirect bit) to be XORed into the value of the expression whenever * is encountered. Thus, the expression

$$** = 20000 \text{ XOR } 20000 = 0.$$

The value of the expression is null (no modification to the open register) unless a number, a symbol, or the character * is in the expression.

The symbol "LAW" is a special case. It should be used only in one of the following two ways:

$$\begin{array}{ll} \text{LAW } \square n & \text{/Equivalent to } 760000 + n \\ \text{or} & \\ \text{LAW } \square -n & \text{/Equivalent to } -n \end{array}$$

Use of LAW in other than the prescribed manner often yields an erroneous value.

3.2.3 The READ Command

The READ command is used to input a patch file from the auxiliary device and to make those patches to the currently selected program or block. An example of a typical patch file source listing is given in Appendix A.

The formats for the READ command are:

- a. >READ ↵
- b. >READ □ FNAME ↵
- c. >READ □ FNAME □ EXT ↵

A filename must be given for input from a file-oriented device. The extension is optional. If missing, the extension is assumed to be ABS since the MACRO assembler outputs .ABS programs onto DECtape, disk, etc., with an extension of ABS. The filename is not needed for paper-tape input. A space plus a comment can be used only when both the filename and extension have been typed in the READ command. When input is from the paper-tape reader, the user must press the tape feed button to clear the NO-TAPE-IN-READER flag.

.ABS patch program on paper tape can be optionally headed by the ABS binary loader (which is ignored by PATCH). The patch file can be either a completely new version of a system program or it can contain patches to specific locations in the program. Both are handled identically.

PATCH reads one data block at a time from the auxiliary input file. For each data word in the block, PATCH calculates the address within the currently selected program or block. If the address is within the current block

in core, the data word patch is made to the current core block; if not, the current block is written out on the tape, and then the block containing the contents of the specified address is brought into core and is patched accordingly.

3.2.4 The EXIT Command

The EXIT command returns control to the Monitor (CTRL C works equally well). The format is:

>EXIT

A space and a comment can follow the command.

3.3 ERROR RECOVERY

If an .IOPS error occurs, causing control to return to the Monitor, the user types, CTRL P (↑P) to restart PATCH. CTRL P is also useful in terminating a READ command if the paper-tape reader jams.

Errors detected by PATCH cause the current command to be terminated and an appropriate error message to be printed. All errors detected by PATCH cause the block in core to be written out onto the output tape provided that modifications were made to that block.

3.3.1 Error Messages

The following is a list of error messages printed by PATCH:

| <u>Message</u> | <u>Cause of Error</u> |
|----------------------|---|
| ILLEGAL COMMAND | Not a legal command name, first command was a list or a read before a select command was issued. |
| NOT OCTAL DIGIT | PATCH expected an octal number and found a character other than 0-7. |
| TOO MANY DIGITS | The user-typed octal number contained more than 6 digits. |
| ADDRESS OUT OF RANGE | The address to be listed is not within the legal range of the currently selected program or block. |
| CHECKSUM ERROR | Bad data read in from auxiliary device. |
| FILE NOT FOUND | The file, as specified in a READ command, does not exist on the auxiliary input device. |
| ILLEGAL BLOCK # | The block number specified in the block select command was outside the range $0 \leq \text{BLKNUM} \leq 1101_8$. |

APPENDIX A

PATCH PROGRAM FORMAT

The following is an example of the proper format for a PATCH assembly source program to be assembled and output by MACRO 9:

```
.TITLE          anything
.ABS            NLD
.LOC           234
734777
600261
.LOC           350
-3
.LOC           371
040674
:
.END
```

APPENDIX B

EXAMPLE OF OPERATION

The following, excluding the comments on the right, is a sample listing of what would appear on the teletype after a session using PATCH. Characters typed by the user have been underlined for clarity. (\$) stands for ALTMODE.

| | |
|--|---|
| <p>MONITOR</p> <p><u>\$PATCH</u>↵</p> <p>PATCH V4A</p> <p>> <u>MACRO</u>↵</p> <p>> <u>L 100</u>↵</p> <p>ADDRESS OUT OF RANGE</p> <p>> <u>L 14366</u>↵</p> <p>> <u>14366/000000</u>> <u>777776</u>↵</p> <p><u>14367/231604</u> > <u>DAC*</u> <u>11604</u>↵</p> <p><u>14370/777777</u> > <u>SNAICLA</u>↵</p> <p><u>14371/615422</u> > <u>JMP</u> <u>16777</u> ↵↵</p> <p>> <u>16777/703112</u> > ↵</p> <p><u>17000/452735</u> > <u>(S)</u></p> <p>> <u>READ</u>↵</p> <p>CHECKSUM ERROR</p> <p>> <u>READ</u>↵</p> <p>> <u>B- 357</u>↵</p> <p>> <u>L 400</u>↵</p> <p>ADDRESS OUT OF RANGE</p> <p>> <u>L 40.</u>↵</p> <p>NOT OCTAL DIGIT</p> <p>> <u>L 40</u>↵</p> <p>> <u>00040/317754</u> > <u>(S)</u></p> <p>> <u>EXIT</u>↵</p> <p>MONITOR</p> <p>\$</p> | <p>/CALL IN PATCH.</p> <p>/SELECT MACRO.</p> <p>/LIST LOCATION 100.</p> <p>/LIST LOCATION 14366.</p> <p>/MODIFY. LIST 14367.</p> <p>/CHANGE LAC* TO DAC*. LIST 14370.</p> <p>/MODIFY. LIST 14371.</p> <p>/MODIFY. LIST 16777.</p> <p>/NEW BLOCK READ IN. NO CHANGE.</p> <p>/NO CHANGE. TERMINATE SEQUENCE.</p> <p>/READ AUXILIARY INPUT (PAPERTAPE).</p> <p>/TRY AGAIN.</p> <p>/SELECT BLOCK 357 IN REVERSE DIRECTION.</p> <p>/LIST LOCATION 400.</p> <p>/PERIOD, THAT IS.</p> <p>/LIST LOCATION 40.</p> <p>/NO CHANGE. END SEQUENCE.</p> <p>/GO BACK TO MONITOR.</p> |
|--|---|

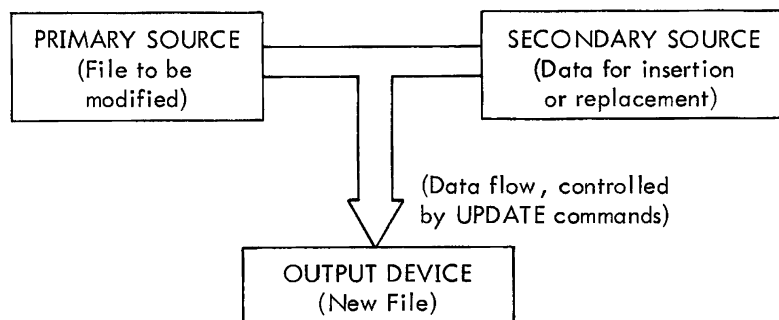
SECTION 1
INTRODUCTION

The ADVANCED Monitor Library Update Utility program (UPDATE) permits the contents of binary library files stored on a file-oriented media to be examined and updated. A binary library file is defined as any file which contains one or more relocatable binary programs. Some examples of binary library files are:

- a. the ADVANCED Monitor's .LIBR BIN file, which contains the standard I/O handlers and FORTRAN IV Object Time System (OTS)
- b. the BACKGROUND/FOREGROUND Monitor system's .IOLIB BIN (I/O Library) and the .F4LIB BIN (OTS routines) files.

UPDATE alters files by replacement, deletion or insertion operations; it can also be used in the creation/modification of user library files (refer to Linking Loader section of this manual).

The UPDATE functions of this utility program are carried out by the generation of a new file from information copied, on command, from either a primary (original file) or secondary (new information) data sources. As shown in the following diagram, the file to be modified is considered the primary data source; the information to be added or inserted into the file is contained by a secondary source. The final modified file is produced on an assigned output device.



SECTION 2

UPDATE COMMANDS

The UPDATE program is called into memory by the following command to the Monitor:

\$ UPDATE ↵ (command can also be terminated with ALTMODE)

When loaded, UPDATE outputs the following response on the teleprinter:

UPDATE Vnn (Vnn indicates version number)
>

The user keyboard commands to UPDATE are divided into three categories, arranged in the following operational sequence:

- a. File specification commands
- b. UPDATE action commands
- c. Termination commands.

2.1 FILE SPECIFICATION COMMANDS

The first command issued to UPDATE must be a file specification command to indicate the UPDATE operation desired and the name of the file to be modified or created. The general form of the file specification command is:



2.1.1 Options

The following three UPDATE options are available:

| <u>Designator</u> | <u>Operation</u> |
|-------------------|---|
| L | List the named library file by element and element size (octal). |
| U | Update the named library file by the insertion, deletion/replacement of elements. |
| N | New library file (named) is to be created. |

The options U and N are mutually exclusive. Option L can be used alone to obtain a clean listing or in combination with either U or N to observe updating results on the file.

The use of the L option provides a library file listing of the following form:

| LIBRARY FILE LISTING FOR FILENM PAGE 1 | | |
|--|------------------|------------------------|
| PROGRAM NAME | PROGRAM SIZE | ACTION |
| | 0 | DELETE NAME |
| NAME 2 | 477 ₈ | REPLACE NAME 1, NAME 2 |
| NAME 4 | 352 ₈ | |
| NAME 3 | 517 | INSERT NAME 3, NAME 4 |
| . | . | . |
| . | . | . |
| . | . | . |

2.1.2 File Name

The command file name consists of any combination of from 1 to 6 alphanumeric characters. If no file name is specified by the user, the name .LIBR BIN is assumed by the program. In all cases, the name BIN is assumed to be the command file name extension.

2.1.3 Command Terminator

Either a carriage return (↵) or an ALTMODE keyboard entry is used to terminate the file specification command.

If a ↵ terminator is used, control is returned to the UPDATE program when work on the named file is complete.

If an ALTMODE terminator is used, control is returned to the Monitor when updating of the named file is complete.

2.2 UPDATE ACTION COMMANDS

After acceptance of the file specifications command, UPDATE is ready to accept commands for the manipulation of the contents of the named file, indicated by the printout:

>

When the symbol > is printed, the named file is positioned immediately before the first element of the file, thus permitting the insertion of new elements at the very beginning of the file. (Insertion is described under the Insert (I) command, 2.2.3.)

There are four action commands (i.e., DELETE, REPLACE, INSERT and END) each of which is described in detail in the following paragraphs.

All action commands are terminated by a `␣`. It must be noted that the UPDATE program operates on a file in a sequential routine-by-routine manner. When a routine is passed, any action command made to that routine results in an error message. Because the preceding is a common operator error, it is recommended that the user have a listing of the library file being worked upon.

2.2.1 DELETE (D) Command

The D command causes the deletion from the specified file of a named routine or a range of routines; this is specified by the names of the first and last routines of the range. The deletion is carried out by copying all the routine or file elements in the file up to the deleted routines onto an output file media.

The delete command can take either of the following two forms; as shown, the term DELETE or the letter D is used.

| <u>Command String</u> | <u>Operation</u> |
|----------------------------------|--|
| >DELETE NAME ␣ (or D) | Delete the named routine. |
| >DELETE NAME1, NAME2 ␣ (or D) | Delete routines NAME1 through NAME2, inclusive |

The DELETE command may be used only when an update (U) option has been specified in the file specification command.

After the delete operation, the file being worked on is positioned at a point immediately after the last routine deleted.

2.2.2 REPLACE (R) Command

The R command causes information in the original file to be replaced by new information by copying all of the elements of the original file onto the output file device or media up to the element or routine to be replaced. The new information (replacement) is then copied onto the output file from a secondary input service. After the replace operation, the output file is positioned at a point immediately after the replaced routine.

The replace command can take either of the following two forms; as shown, the letter R or the word REPLACE is used.

| <u>Command String</u> | <u>Operation</u> |
|-------------------------------------|--|
| > REPLACE □ NAME ↵ (or R) | NAME is replaced by a new routine of the same name. |
| > REPLACE □ NAME1,NAME2 ↵ (or R) | NAME1 is replaced by a new routine having the name NAME2 |

The REPLACE (R) command may be used only when an update (U) option has been specified in the file specification command.

2.2.3 INSERT (I) Command

The I command causes information contained on the secondary input source to be inserted, at any point, into the original file as it is copied onto the output file device media. On completion of the insert operation, the output file is positioned at a point immediately after the inserted material.

The I command can take either of the following two forms; as in the previous commands, the letter I or the term INSERT may be used.

| <u>Command String</u> | <u>Operation</u> |
|------------------------------------|---|
| > INSERT □ NAME ↵ (or I) | The routine NAME is inserted into the output file starting at the current position of the file. |
| > INSERT □ NAME1,NAME2 ↵ (or I) | The routine NAME1 is inserted into the specified file immediately after routine NAME2. |

The I command is used only when an update (U) option has been previously specified in the file specification command.

2.2.4 END (E) Command

The use of E command causes the output file to be positioned at the end of the final routine of the original file. This is accomplished by copying all the information from the original file, from its current position, onto the output file. The END command is a convenient method of positioning the file to be modified in order that new routines from the secondary input may be appended to the file via the INSERT (I) command.

The form of the END command is as follows; as shown, the letter E or the term END may be used.

| <u>Command String</u> | <u>Operation</u> |
|-----------------------|---|
| > END ↵ (or E) | All of the information on the original file from its current position is copied onto the output file. |

The End (E) command is used only when an update (U) option has been previously specified in the file specification command.

2.3 UPDATE TERMINATION COMMANDS

The UPDATE operation (U option) is terminated by either the CLOSE (C) or the KILL (K) commands.

2.3.1 CLOSE (C) Command

The CLOSE (C) command can take either of the following two forms;

| <u>Command String</u> | <u>Operation</u> |
|--|--|
| >CLOSE ↵ (or C) (or ALTMODE) | The file is closed and the name given in the file specification command is assigned to the file. |
| >CLOSE FILENM ↵ (or C) (or ALTMODE) | The file is closed and the name FILENM BIN is assigned to the file. |

NOTE

Terminating the C command with ↵ restarts the UPDATE program; terminating it with ALTMODE returns control to the Monitor.

The C command is normally used to terminate UPDATE operations. The use of this command causes all of the remaining elements on the original (primary input) file to be copied from its current position onto the output file. All optional operations (e.g., listing (L)) are also completed on the initiation of this command.

2.3.2 KILL (K) Command

The use of the K command aborts the current UPDATE operations and returns control to UPDATE. The form of this command is:

>KILL ↵
(or K)

SECTION 3
DEVICE (.DAT) SLOT ASSIGNMENTS

The UPDATE utility program uses six negative .DAT slots, which are assigned (according to use and option) in the following manner:

| <u>.DAT Slot</u> | <u>File</u> | <u>Options Specified</u> |
|------------------|-------------------------|--------------------------|
| -15 | Library File Output | U,N |
| -14 | Library File Input | U,L |
| -12 | Library Listing | L |
| -10 | Secondary (new) Input | U,N |
| -3 | Teletype Command Output | U,L,N |
| -2 | Teletype Command Inputs | U,L,N |

SECTION 4

ERROR CONDITIONS AND RECOVERY PROCEDURES

The UPDATE program outputs either recoverable or unrecoverable error messages. In either case, the user always has the option of restarting UPDATE by typing CTRL P (↑P) or returning control to the Monitor by typing CTRL C (↑C).

4.1 RECOVERABLE ERRORS

When a recoverable error message is printed, the user must retype his command for recovery. This procedure works unless the specified program name cannot be found (see Improper Name in Action Command).

4.1.1 Unintelligible Commands

If UPDATE cannot understand a command, it prints out the following:

```
> ?  
>
```

The user should then check his previously entered command to ensure that it complies in sequence and form with the requirements of UPDATE.

4.1.2 Command Function/Option Errors

Error messages printed during recoverable error conditions caused by improper function commands are listed below:

- a. If a two-argument D, R, or I action command is used with any option other than U, the UPDATE program outputs:
VALID ONLY IN U MODE - COMMAND IGNORED
>
- b. If a single-argument D, R, or I action command is used in any but the U and N option modes, the program outputs:
VALID ONLY IN U OR N MODE - COMMAND IGNORED
>
- c. If no name is given in the I, D, or R command string, the program outputs:
ILLEGAL COMMAND STRUCTURE - COMMAND IGNORED
>

4.1.3 Improper Name in Action Command

If the name given in any UPDATE action command cannot be found, UPDATE outputs:

```
EOF REACHED BY SEARCH - COMMAND IGNORED  
>
```

If this error condition is indicated, the library file is positioned at a point immediately after its last routine and the file can be accessed only by the INSERT (I), CLOSE (C) or KILL (K) commands.

4.1.4 Incorrect Input Source

Named data for insertion/replacement functions (i.e., inputs from devices assigned to .DAT slot-10) which cannot be found on the input device causes the following error printout:

```
WRONG PROGRAM AS INPUT - CORRECT INPUT AND tP
```

In most cases, this error printout indicates that the wrong program (e.g., paper tape) was placed in the reader. If this is the case, the user need only place the proper tape in the reader, clear the reader, and type tP to continue. For file oriented devices, the file name and the program name (the name given at assembly or compile time) must both agree with the name in the UPDATE command.

4.2 UNRECOVERABLE ERRORS

The following errors are terminal; they require the user to restart with a file specification command.

- a. If an end code is detected before the program name on binary input, the following is printed:

```
PROGRAM NAME MISSING - DYNAMIC KILL UPDATE  
>
```

- b. If there is not enough room in core for the program, the following is printed:

```
BUFFER OVERFLOW - DYNAMIC KILL  
UPDATE  
>
```

- c. If a read error occurs in the input buffer, the following is printed:

```
UNRECOVERABLE READ ERROR ON .DAT N - DYNAMIC KILL  
UPDATE  
>
```

SECTION 5

UPDATE EXAMPLE

5.1 UPDATE FILEA

To update FILEA: (user responses are underscored).

| | |
|---------------------------|--|
| UPDATE | |
| > <u>U ←FILEA ↵</u> | /File specifying command must be first |
| > <u>I NAME2, NAME3 ↵</u> | /Insert routine NAME2 after NAME3 |
| > <u>R NAME4, NAME5 ↵</u> | /Replace routine NAME4 with NAME5 |
| > <u>D NAME1 ↵</u> | /Delete routine NAME1 from file |
| > <u>C ↵</u> | /Close FILEA |
| > UPDATE | /Returns to UPDATE since |
| > | /file specifying command above |
| > | /was terminated with a ↵ |

5.2 UPDATE BCDIO

To update BCDIO on the Monitor Systems .LIBR file (user responses are underscored)

| | |
|--------------------------------------|---|
| \$ <u>A DTAO-14/DTA1-15/PRA-10 ↵</u> | /Scratch tape on .DAT slot-15 |
| \$ <u>UPDATE ↵</u> | /Call Update |
| UPDATE | |
| > <u>U ← ↵</u> | /Specify Update function (note: .LIBR BIN assumed) |
| > <u>R BCDIO ↵</u> | /Replace BCDIO with new version from paper tape reader |
| > <u>C ↵</u> | /Close the file |
| UPDATE | /New .LIBR on .DAT slot-15 |

To complete the update of the MONITOR System's .LIBR BIN file, it is necessary to call in PIP to delete the old .LIBR from unit 0 and to transfer the new .LIBR from unit 1 to unit 0.

SECTION 1
INTRODUCTION

This chapter describes the operation of the Linking Loader and the composition of the binary information which comprises a loadable program unit. Operating procedures for the BASIC I/O and ADVANCED Monitor environments are included along with memory maps of the various phases of loading by the Linking Loader.

SECTION 2

DESCRIPTION

2.1 OPERATION

The Linking Loader loads and links relocatable or absolute[†] binary program units as produced by the FORTRAN IV Compiler and the MACRO Assembler. Absolute and relocatable coding should not be intermixed in one unit, and care should be taken in linking relocatable and absolute units. For FORTRAN generated program units, the Loader also assigns the common data storage area. Any input device that will input binary code can be used.

Initially, the loader loads all the program units the names of which appear in the command string (see Operating Procedures, Section 7). Then, the Loader automatically loads and links all requested and unresolved library subprograms. The requested library subprograms are loaded from the external (user) library (if one exists) and the system library (in that order). After both libraries have been examined for requested subprograms, the Loader prints the names of all subprograms which have not been found. If the user requires I/O handlers that are already in core for Linking Loader purposes, the resident handlers are used.

Individual program units cannot be executed if the program flows across a 4K (8K for PDP-9) memory bank; the Loader prevents this type of loading. The Loader will, however, load (and link) the program in the next memory bank. No overlap checking of any kind is made with absolute binary program units.

Optionally, symbols and their absolute definitions are loaded into a program dictionary (symbol table) for use by the dynamic debugging technique (DDT). The loader also sets up for DDT the start execution address of the main program (in the system communication table) and the initial relocation value of all the program units (in the symbol table).

2.2 FORTRAN COMMON STATEMENTS

The Linking Loader permits FORTRAN COMMON blocks and block-data subprograms to overlap memory banks.

For PDP-9 Systems, FORTRAN allows COMMON block sizes greater than 8192_{10} , provided that each element in COMMON does not exceed 8192_{10} . For example,

[†]A MACRO assembled program headed by an absolute .LOC statement, e.g., .LOC 100, is an absolute binary program and the binary is output in link loadable format. A program headed by a .ABS statement is output as absolute block binary and cannot be loaded by the Linking Loader.

```
COMMON /I/L(100,100)
```

is illegal because the size of array L is 10000_{10} .

```
COMMON /I/L1(100,50),L2(100,50)
```

is acceptable. Each array size is 5000_{10} and the size of the common block is 10000_{10} .

For a PDP-15, FORTRAN allows COMMON block sizes greater than 4096_{10} , provided that each element in COMMON does not exceed 4096_{10} . For example,

```
COMMON /J/K(80,80)
```

is illegal because the size of array K is 6400_{10} .

```
COMMON /J/K1(80,40),K2(80,40)
```

is legal. Each array size is 3200_{10} , and the size of the common block is 6400_{10} .

Non-COMMON arrays and variables are initialized to zero (0) by the Loader.

MACRO programs can be linked to COMMON areas defined by FORTRAN IV. If any virtual globals remain after the Loader has searched the user and system libraries and has defined COMMON blocks, an attempt is made to match those global names to COMMON block names. If a match is made, the global becomes defined as the COMMON block. An example follows:

```
FORTAN IV PROGRAM
  INTEGER A,B,C
  COMMON /NAME/C
  COMMON A,B
  :
MACRO PROGRAM
  .GLOBL NAME, .XX          /.XX is name given to blank COMMON
                             /by the F4 Compiler
  DZM* .XX                 /CLEAR A - NOTE INDIRECT REFERENCE
  ISZ .XX                  /BUMP COUNTER
  DZM* .XX                 /CLEAR B
  DZM* NAME                /CLEAR C
```

Note that if the values are REAL (2 words) or DOUBLE PRECISION (3 words), the MACRO program must account for the number of words when accessing specific variables.

SECTION 3

INFORMATION UNITS

The binary output from the FORTRAN compiler and the MACRO Assembler consists of blocks of information units. Each information unit consists of an identification code (6 bits) and a data word (18 bits). The form of the object program at run time is determined by the content and the ordering of the information units. Several information units may be grouped to convey a single run-time instruction to the Loader.

A block of information units consists of four 18-bit machine words arranged in the following manner:

| | | | | | | |
|--------|-------------|---|--------|----|--------|----|
| | 0 | 5 | 6 | 11 | 12 | 17 |
| Word 1 | Code 1 | | Code 2 | | Code 3 | |
| Word 2 | Data Word 1 | | | | | |
| Word 3 | Data Word 2 | | | | | |
| Word 4 | Data Word 3 | | | | | |

Standard IOPS binary line sizes (48 information words and a 2-word header) are input by the Loader.

SECTION 4

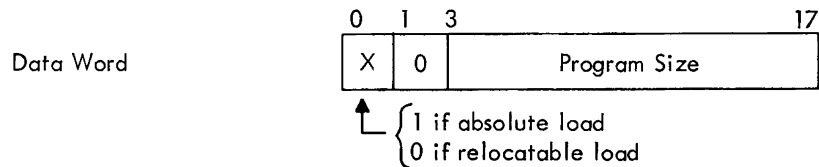
IDENTIFICATION CODES

The identification code instructs the Loader on how to handle the associated data word. There is an implied order in which codes appear within a binary file (described in general under Program Unit Organization, Section 5).

| <u>Code</u> | <u>Loader Action</u> |
|-------------|----------------------|
|-------------|----------------------|

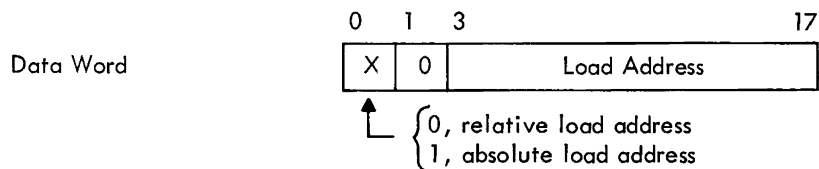
| | |
|----|-------------------|
| 01 | Program Unit Size |
|----|-------------------|

The data word specifies the number of machine words required by this program unit. This number does not include the required number of machine words for common storage. The program size is used by the Loader to determine whether or not the program will fit within the unused locations of any available 4K (8K for PDP-9) memory bank. Loading terminates with an appropriate error message if the program cannot be loaded. This information unit appears only once per program unit and is the first information unit of the binary output. In absolute loads, no checking is made for overlays; this is left to the discretion of the user. The program size is also used to determine where to begin loading as loading proceeds from the top of core down (see Memory Maps).



| | |
|----|----------------------|
| 02 | Program Load Address |
|----|----------------------|

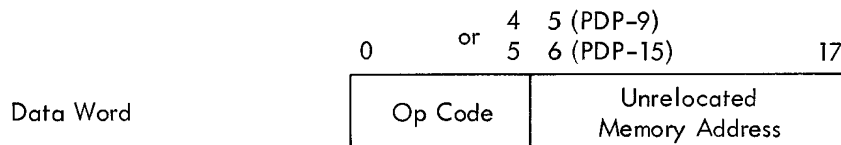
The data word is an unrelocated memory address. This address specifies either an absolute or a relative storage address for program data words and is incremented by one for each data word stored (codes 03, 04, and 05). If the address is relative, it is initially incremented by the current relocation factor (modulo 15 bits). Bit 0 of the data word is used to indicate an absolute address (bit 0 = 1) or a relative address (bit 0 = 0).



CodeLoader Action

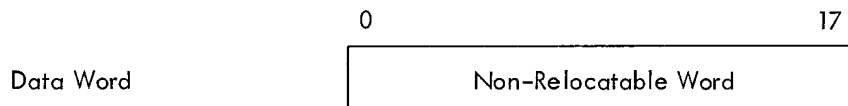
03 Relocatable Instruction

The data word is a memory referencing instruction. The address portion of the instruction is incremented by the current relocation factor (modulo 12 bits for the PDP-15 and 13 bits for the PDP-9). The instruction is stored in the location specified by the load address which is incremented by one after the word is stored.



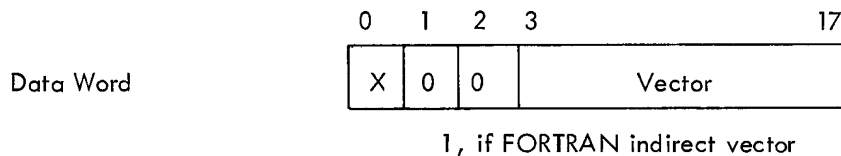
04 Absolute Instruction/Constant/Address

The data word is either a non-memory referencing instruction, a non-relocatable memory referencing instruction, an absolute address, or a constant. The word is stored in the location specified by the load address which is incremented by one after the word is stored.



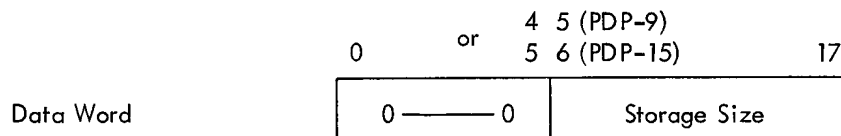
05 Relocatable Vector

The data word contains a relocatable program address (vector). The word is incremented by the current relocation factor (modulo 15 bits). The data word is stored in the location specified by the load address which is incremented by one after the word is stored.



06 Non-Common Storage Allocation

The data word specifies the number of machine words required for non-common variable and array storage. Storage allocation begins at the address specified by the load address. The load address is incremented by this number. This block of memory is cleared.



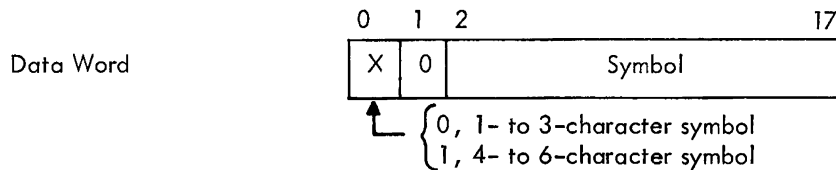
CodeLoader Action

07

Symbo! - First Three Characters

The data word contains the first 3-characters of a symbol in radix 50_8 format (see Appendix A).

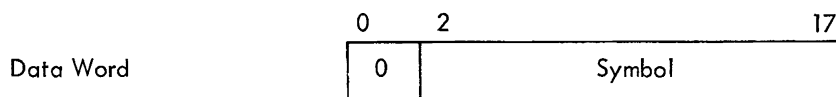
The data word is saved by the loader for future reference.



08

Symbol - Last Three Characters

The data word contains the last 3 characters of a symbol in radix 50_8 format. The data word is saved by the loader for future reference. This word is used only if in the code 07 data word bit 0 = 1.



09

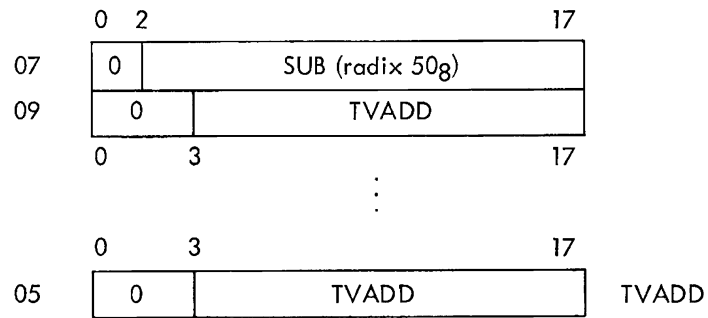
External Symbol Definition

The data word contains the unrelocated address of the transfer vector for the subprogram named by the last symbol loaded (codes 07 and 08). If the external subprogram has already been loaded, the address (definition) of the symbol is stored into the specified vector address (relocated modulo 15 bits). If the subprogram has not been loaded and this is the initial request, the symbol and the relocated (modulo 15 bits) transfer vector address are entered into the Loader symbol dictionary as a request for subprogram loading. This action automatically forces the Loader into a Library Search Mode when the end of the command string is encountered. If the Loader is already in the Library Search Mode, it remains there until all virtual globals have been resolved. If the subprogram has been previously requested (symbol in dictionary) but not loaded, the Loader chains the reference locations. This chain, generated exclusively by the Loader, is followed when the external definition is encountered. (Unchained transfer vector locations must initially contain a reference address (code 04 or 05) to themselves.) For example, .GLOBL SUB where SUB is virtual should cause the output of the following:

Code

Loader Action

09 (Cont)



and SUB defined internally as TVADD. Subroutine calls are made via JMS* SUB

Data Word

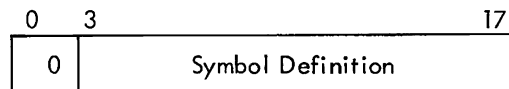


10

Internal Global Symbol Definition

The data word contains the unrelocated or absolute address (definition) of the last symbol loaded (codes 07 and 08). The last symbol loaded is a global symbol internal to the program unit which follows. In the Library Search Mode, if a request for subprogram loading exists (code 09) in the Loader dictionary, the relocated (modulo 15 bits) or absolute definition is stored in the specified transfer vectors and the program unit is loaded. The definition also replaces the transfer vector address in the Loader dictionary. If no request for loading exists, the program unit is not loaded and the Loader continues to examine information units until the next internal global symbol definition is found (Library Search Mode). If the program unit is to be loaded, all internal global symbols following the one causing loading are automatically entered into the Loader dictionary as defined global symbols. If the symbol already exists in the dictionary and is defined (indicating that a program unit with the same name is already loaded) the current program unit is ignored.

Data Word

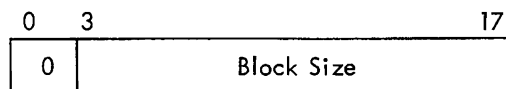


11

Block Data Declaration

This information unit instructs the Loader that the common blocks and data constants following are part of a block data subprogram.

Data Word



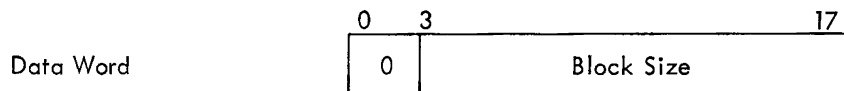
CodeLoader Action

12

Common Block Definition

The data word specifies the number of machine words required for the common block named by the last symbol loaded (codes 07 and 08). In general, the assignment of memory space for the common block is deferred until all requested library subprograms have been loaded. The exception to this rule occurs when the block data declaration (code 11) has been encountered. In this case, the common block name is treated as an internal global symbol, and the block is assigned to memory. After the block is assigned to memory, the starting address is entered into the Loader dictionary, and the starting address is saved by the Loader for future use (code 13). All symbols in the dictionary associated with the block are assigned addresses with respect to this starting address. All symbols which are yet to be loaded (via code 13 and 14) will also be assigned as they are encountered. When the block data flag is not set, the Loader enters the name and the size into the dictionary (if it is not already there) and also enters the word containing the next available dictionary entry address. This entry will contain the first symbol in this common block and will be used as the head of the chain of all symbols in this common block. The address of the head of chain is saved by the Loader so that the new set of symbols in the common block may be added to the chain. The larger of the two block sizes is retained as the block size.

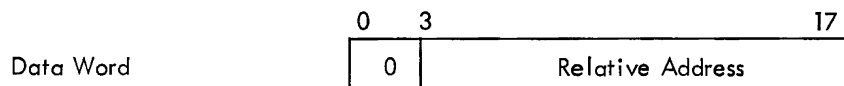
When the common block has already been assigned memory locations, the respective lengths are compared. Loading terminates, with an appropriate error message, if the assigned block is smaller. When the assigned block is larger or both are equal, loading continues.



13

Common Symbol Definition

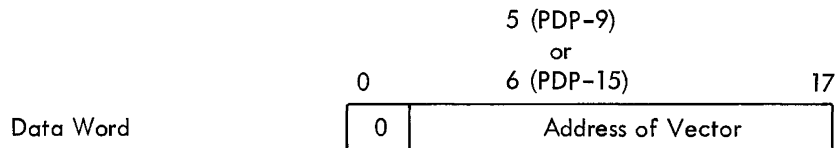
The data word specifies the relative location of the last symbol loaded (codes 07 and 08) in the last common block (code 12). If the associated common block has been defined (block data), the absolute address of the symbol is calculated (block address plus relative position) and placed in TV location (code 14). When the common block has not been assigned, the relative address is entered into the Loader dictionary and chained to the symbols associated with the common block.



CodeLoader Action

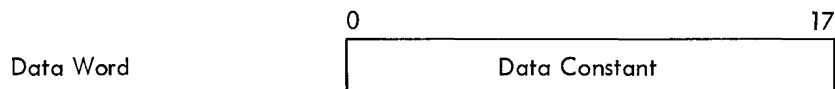
14 Common Symbol Reference Definition

The data word contains the unrelocated address of the transfer vector for references to the common symbol named by the last symbol loaded (codes 07 and 08). The symbol definition (code 13) is stored in the relocated (modulo 15 bits) address specified when the associated common block has been assigned (code 12). When the block has not been assigned, the relocated (modulo 15 bits) address is entered into the Loader dictionary along with the relative address (code 13) of the symbol.



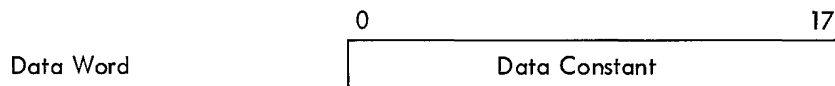
15 Data Initialization Constant - First Word

The data word contains the first machine word of a data initialization constant. It is saved by the Loader for future use (code 18).



16 Data Initialization Constant - Second Word

The data word contains the second machine word of a data initialization constant. It is saved by the Loader for future use (code 18).



17 Data Initialization Constant - Third Word

The data word contains the third machine word of a data initialization constant. It is saved by the Loader for future use (code 18).

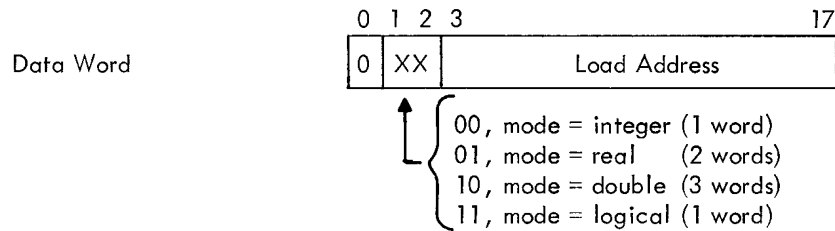


18 Data Initialization Constant Definition

The data word contains the relative load address of the last data initialization constant loaded (codes 15, 16, and 17) and a mode code identifying the constant (real, integer, double, logical). The load address is incremented by the current relocation factor (modulo 15 bits) if the constant initializes a non-common storage element. When the constant

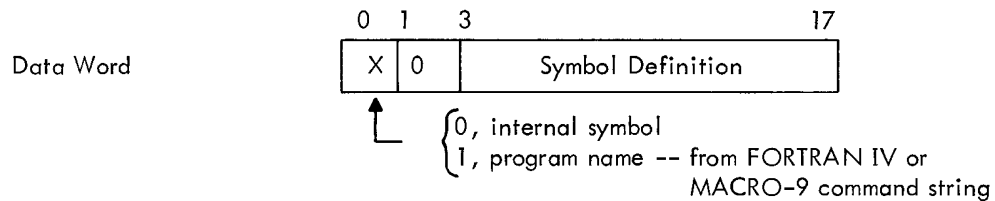
CodeLoader Action

- 18 (Cont) initializes a common storage element (indicated by the presence of the block data flag, code 11), the load address is incremented by the address of the last common block loaded (code 12). The constant is stored according to mode and the relocated load address.



- 19 Internal Symbol Definition

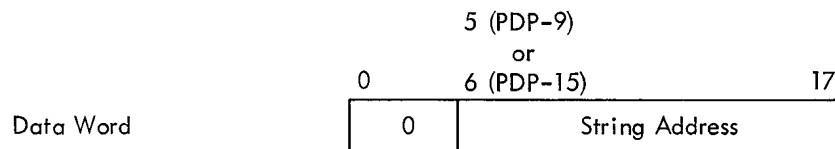
The data word contains the unrelocated or absolute address (definition) of the last symbol loaded (codes 07 and 08). The symbol is strictly internal to the program being loaded and is entered conditionally (if a DDT Load) along with its relocated (modulo 15 bits) or absolute address into the DDT symbol dictionary. The program unit name is indicated by bit 0=1 of the data word.



All symbols fall into this category.

- 20 String Code - First Half

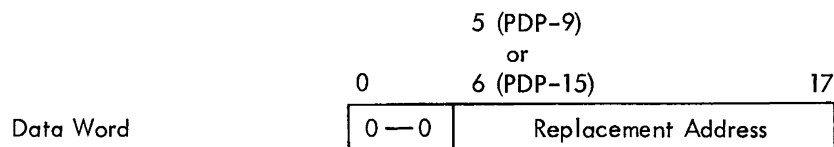
The data word contains the unrelocated address of a data word whose address portion is to be replaced by another value. The relocated (modulo 15 bits) address is saved by the Loader for future use (code 21).



CodeLoader Action

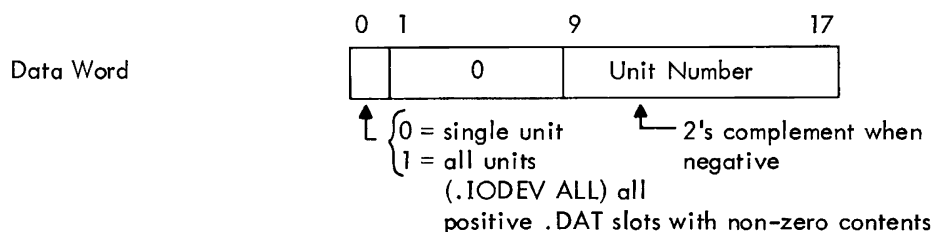
21 String Code - Second Half

The data word contains an unrelocated address. The address portion of the data word specified by the first half-string code (code 20) is replaced with this address (relocated modulo 12 bits (13 bits for a PDP-9)).



22 Input/Output Device Routine Request

The data word specifies the unit number (.DAT slot number) associated with a device level I/O routine. The Loader defers loading of any I/O routines until all other subprogram loading has been completed; when subprogram loading is complete, the system library is searched for all requested I/O device routines not already residing in memory (see Operating Procedures). The I/O routines are then loaded.



23 End of Program Unit

This information unit is the last unit of a program unit. The data word contains the unrelocated or absolute start execution address of the program. The relocated (modulo 15 bits) or absolute start address is entered into the system communication tables to be used when control is given to the user. Only the first start address encountered is entered into the communication tables. (It is assumed that the first program unit specified in the command string is the main program.) The first address of the main program is used if the .END pseudo-op did not have a start address. When loading from either the system or external libraries, the end unit causes the Loader to examine the next line buffer for the end-of-file (EOF) condition. When the EOF for the external library is obtained, the Loader automatically begins searching the system library to resolve any remaining globals. Upon encountering the EOF of the system library, the Loader announces any unresolved global names. When loading is complete, control goes to the user program, DDT, or to the keyboard listener in the Monitor as a function of the load command (GLOAD, LOAD, DDT, or DDTNS) (see Operating Procedures).

CodeLoader Action

| | | | |
|-----------|---|---------------|----|
| | 0 | 3 | 17 |
| Data Word | 0 | Start Address | |

24 18-Bit Parameter Assignment

The data word is ignored by the Linking Loader. It was to have been passed on to DDT in the symbol table, however DDT was not implemented to recognize it.

SECTION 5
PROGRAM UNIT ORGANIZATION

5.1 MAIN PROGRAM AND SUBPROGRAM ORGANIZATION

PROGRAM SIZE (code 01) for absolute or relocatable program, does not include COMMON size
INTERNAL GLOBAL DEFINITIONS (code 10)

PROGRAM NAME (code 19)

PROGRAM LOAD ADDRESS (code 02) absolute or relative

COMMON STORAGE (codes 12, 13 and 14)

NON-COMMON STORAGE (code 06)

 Array Declaration Information

 Equivalenced Arrays and Variables

 Non-Equivalenced Arrays

PROGRAM BODY

| <u>Codes</u> | | <u>Codes</u> | |
|--------------|--------------|--------------|--------|
| 03 | } | 07 | } |
| 04 | | 08 | |
| 05 | | | |
| | Instructions | | Symbol |
| | & | | |
| | Literals | | |

 Non-COMMON Variables and Arrays (06)

 Transfer Vectors (code 05)

EXTERNAL GLOBAL SYMBOL DEFINITIONS (code 09)

END (code 23)

5.2 BLOCK DATA SUBPROGRAM ORGANIZATION

BLOCK DATA INDICATOR (code 11)

PROGRAM NAME (code 19)

COMMON STORAGE (codes 12, 13, and 14)

DATA INITIALIZATION CONSTANTS (codes 15, 16, 17, and 18)

END (code 23)

SECTION 6

DEFINITIONS

| <u>Term</u> | <u>Definition</u> |
|------------------------------|---|
| Loadable Program Unit | A main program, subprogram, or a block data subprogram. |
| Transfer Vector | A core location containing the address of a subprogram or an entity in COMMON. All references to subprograms and entities in COMMON are indirect. |
| Internal Global Symbol | A symbol defined in the current program unit and accessible to all programs. |
| External Global Symbol | A symbol which is referenced in the current program unit and defined in another. |
| Virtual Global Symbol | An external global symbol reference which has not yet been resolved by replacement with the internal global symbol definition. |
| Relocation Factor | The amount added to relative addresses to form absolute addresses; initially, the first loadable core location. The relocation factor for programs following the first program unit is the next available load address. |
| Radix 50 ₈ Format | A method of symbol concatenation [†] utilizing 50 ₈ characters as a "number set", each with a unique value between and including 0 to 47 ₈ . The symbol (number) is converted using standard base conversion methods (see Appendix A). |

[†]i.e., linking together

SECTION 7

LINKING LOADER OPERATING PROCEDURES

7.1 BASIC I/O MONITOR ENVIRONMENT

When the Linking Loader is ready to accept the load command string from the keyboard, it outputs to the teleprinter:

LOADER

> Set up the input device and, if it is the paper tape reader, momentarily depress the tape feed control to clear the reader out-of-tape flag.

The file names of all the programs that are to be unconditionally loaded from the input device (.DAT Slot -4) must be input from the teletype keyboard in the following form:

> NAME1, NAME2, NAME3
> NAME4, NAME5 (ALT MODE)

Comma and carriage return are used as file name separators. ALTMODE terminates the command string. N RUBOUTS are typed to erase the N previous characters in a line. CTRL U is typed to erase everything typed on the current line; however, if a typing error is discovered in a previous line, the user must type control P to restart the loader.

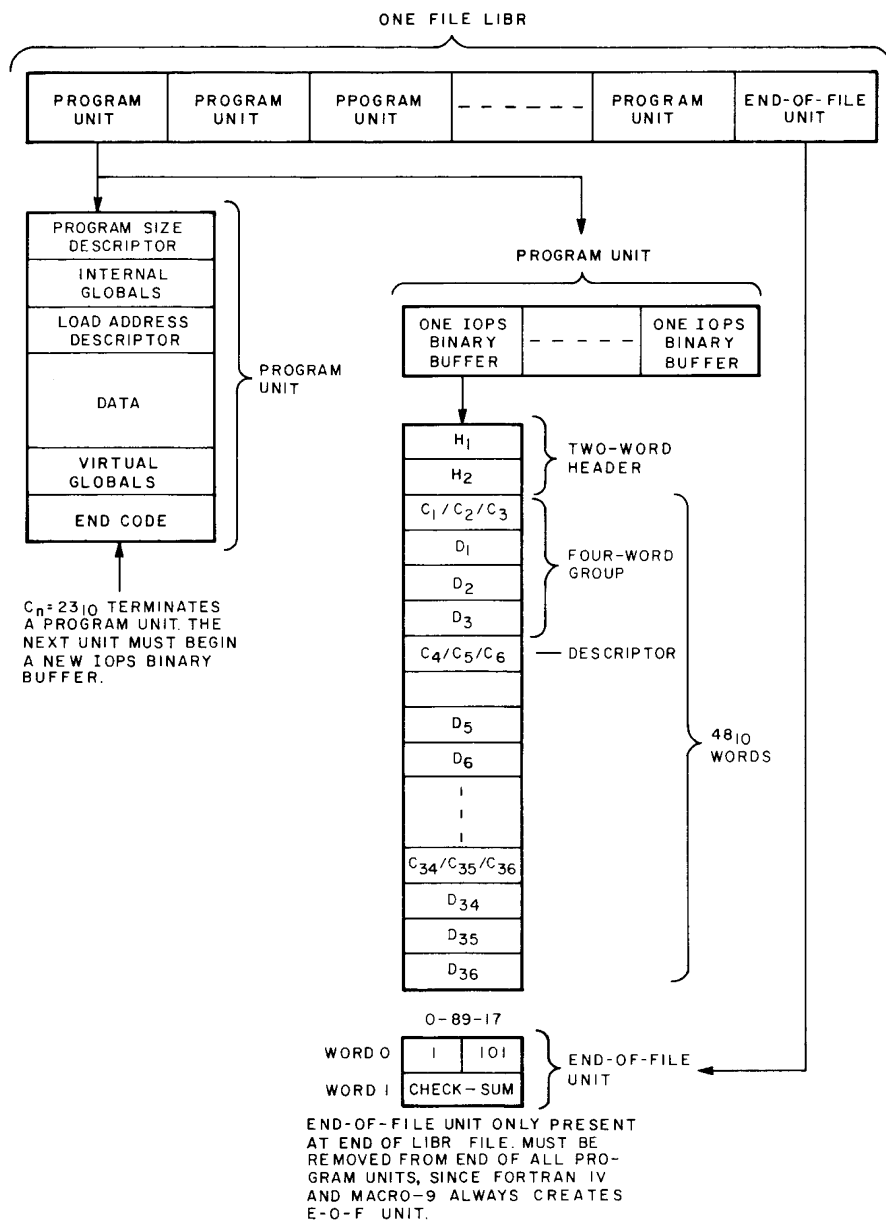
The main program must be requested first. The file names consist of one to six characters (any characters over 6 are ignored). File names are exactly those used in command strings for assembly or compilation. The Linking Loader appends the extension BIN.

When the input device is not file-oriented, N commas followed by ALTMODE primes the Loader to load N + 1 programs from the device.

After loading the programs requested in the keyboard command string, the Loader attempts to resolve all unsatisfied subroutine requests by scanning the system library (.DAT Slot-1). If, after scanning the library, there still remain some unresolved global symbol references, the Loader first defines all existing COMMON blocks and then defines these globals as the base address of COMMON blocks with the same name.

The library must be in the format shown in the following illustration.

7.1.1 Structure of System Library



15-0066

7.1.2 Loader Memory Map

The loader outputs to the teleprinter the names and relocation factors (starting load addresses) of all the programs requested in the command string, followed by the required library routines in the following format:

| | |
|-------|-------|
| NAME1 | 16572 |
| NAME2 | 14301 |
| NAME3 | 10765 |
| NAME4 | 06427 |
| NAME5 | 06313 |
| LIBR1 | 05304 |
| LIBR2 | 04112 |

NOTE

The relocation factor for absolute programs is zero. Whenever the Loader detects end-of-medium in the system library input device, $\uparrow P$ is typed on the teleprinter. To continue, load more input (if the device is a paper-tape reader, momentarily depress the tape-feed control), and type CTRLP on the keyboard.

7.1.3 Error Messages

The Loader outputs to the teleprinter .LOAD followed by the pertinent error code and then halts (in paper-tape system) or exits to the Monitor (in Monitor System).

| <u>Error Code</u> | <u>Meaning</u> |
|-------------------|---|
| 1 | Memory overflow - the Loader's symbol table and the user's program have overlapped. The loader memory map will contain printouts of all programs successfully loaded prior to the one which caused the memory overflow. Use of COMMON storage may enable the program to be loaded as it can overlay the Loader and its symbol table because it is not loaded into until run time. |
| 2 | Input Data Error - parity error, checksum error, illegal data code or buffer overflow (input line bigger than Loader's buffer). |
| 3 | Unresolved globals - if an explicitly or implicitly requested program cannot be found, it will appear in the memory map with an address of 00000. This indicates that loading was unsuccessful; the cause of the trouble should be remedied and loading tried again. |
| 4 | Illegal .DAT slot request - the .DAT slot requested is <ol style="list-style-type: none">out of the range of legal .DAT slots0does not have a device associated with it; that is, it was not set up at SYSTEM generation time, and (in ADVANCED Monitor systems) was not set up by an ASSIGN command. |

When all requested programs have been loaded and all library requests satisfied, the Loader outputs $\uparrow S$ on the teleprinter and sits in a JMP loop. Typing $\uparrow S$ on the keyboard gives control to the starting address of the user's main program.

NOTE

If use is to be made of the paper-tape reader, load the reader and then momentarily depress the tape-feed control.

When the user program has completed its operation and terminated via the .EXIT command, the computer will halt.

If this is a DDT load, on completion of the loading and building of a DDT symbol table (exclusive of the library routine symbols and those of DDT itself) control is automatically given to the starting address of DDT. DDT types `DDT` to inform the user that it is waiting for a DDT command.

The user can force control back to DDT at anytime, by typing `!T` on the teletype keyboard.

7.2 ADVANCED MONITOR ENVIRONMENT

The operating procedures noted below are required in addition to those described under the BASIC I/O Monitor environment.

After loading the programs requested in the keyboard command string, the Loader attempts to resolve all unsatisfied subroutine requests by scanning the external (.DAT Slot -5) and system (.DAT Slot -1) libraries, in that order.

In order to inform the Loader that an external (user) library file exists for this load, it is necessary to `ASSIGN` an I/O device to .DAT Slot -5 prior to the `LOAD`, `DDT`, `DDTNS` or `GLOAD` command, i.e.,

```
$ASSIGN   DTA4   -5
$LOAD
```

The format of the external library file is identical to that of the system library file (see Section 7.1.1).

If this is a DDT load, (DDT), on completion of the loading and the building of a DDT symbol table (exclusive of the library routines' symbols and those of DDT itself), control is automatically given to the starting address of DDT.

A program can be loaded with DDT but without the user symbol table by requesting loading with the `DDTNS` keyboard command. For example,

```
$DDTNS
```

This feature gives the user more operating space but deprives him of symbolic references to user symbols in DDT commands.

If a loading error occurs, an appropriate error message is output to the teleprinter and control is given to the system bootstrap to reinitialize the Keyboard Monitor.

When all the requested programs have been loaded and all library requests satisfied, the Loader will

- a. If `LOAD`, wait on the recognition of `!S` by the keyboard handler and then give control to the starting address of the user's main program.

- b. If GLOAD, give control to the starting address of the user's main program.
- c. If DDT or DDTNS, automatically give control to the starting address of DDT.

When the user program has completed its operation and terminated via the .EXIT command, control is given to the system bootstrap to reinitialize the ADVANCED Monitor and wait for the next keyboard command.

SECTION 8

MEMORY MAPS

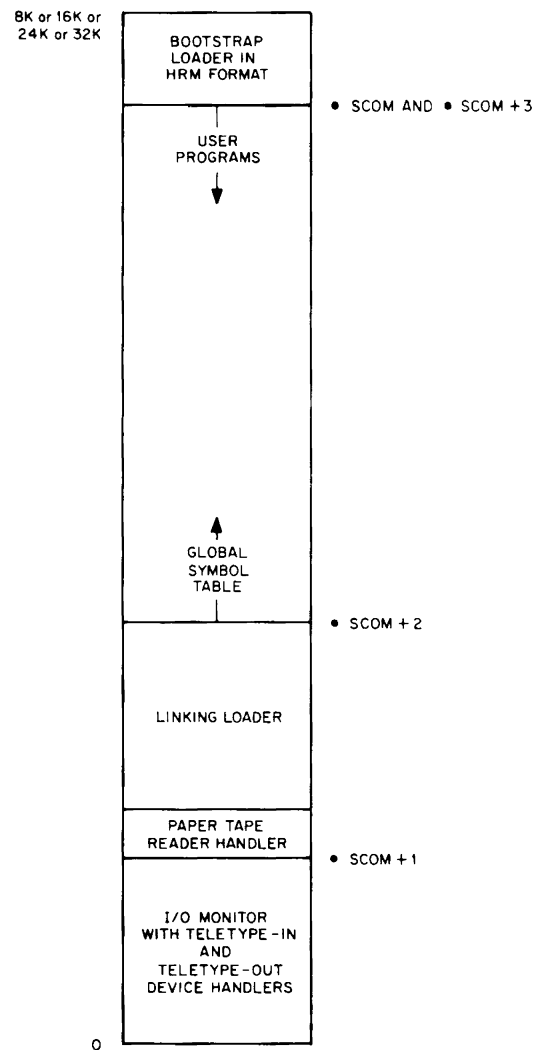
8.1 INTRODUCTION

The following paragraphs present memory maps depicting the core allocation for the various types of programs which may be loaded during Linking Loader operations. These maps, as shown, apply only to systems which have core sizes that are some multiple of 8K (i.e., 8K, 16K, 24K, or 32K). In PDP-15 Systems core is available in 4K modules; thus, it is possible to have systems which, based on 8K multiples, have an odd 4K segment (i.e., 12K, 20K, or 28K). In systems of this type, the odd number of 4K segments is indicated by having bit 0 = 1 in .SCOM+20. To make the memory maps presented in this Section applicable to systems having an odd number of 4K core memory modules (segments), the following additional information must be considered when viewing the maps.

In systems containing an odd number of 4K core segments, the bootstrap loader does not reside at the top of core but, rather, resides in the top of the highest even-numbered 4K segment; the top (odd-numbered) segment is left free. When the Linking Loader is brought into core, registers .SCOM and .SCOM+3 are set to point to the core register located immediately below the bootstrap. During the core loading process, DDT and user programs are loaded starting at the top of core (in top odd-numbered segment); care is taken to avoid overlaying the bootstrap. If the loaded programs fit entirely within the top 4K segment, the pointer which indicates the highest free core location (.SCOM+3) is set to point to the register located immediately below the bootstrap to protect it.

8.2 I/O MONITOR ENVIRONMENT

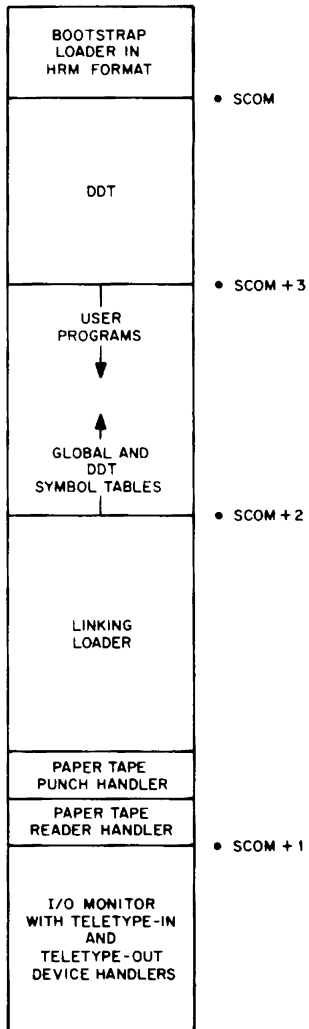
Linking Loader Tape



Refer to memory map 2A of Advanced Monitor Systems for results of Link Loading.

09-0126

8K or 16K or
24K or 32K



09-0125

Refer to memory map 2B of Advanced Monitor Systems for results of Link Loading in DDT mode.

Paper Tape Punch Handler is only present in version of DDT with patch file capabilities.

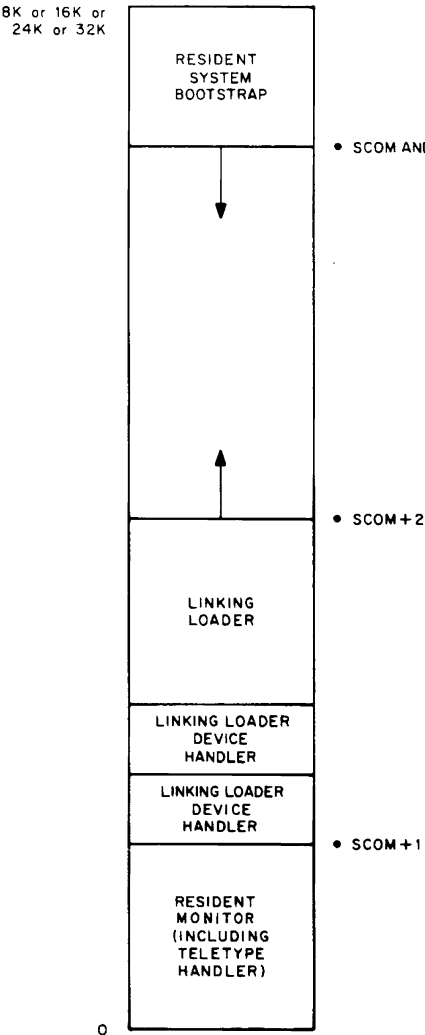
NOTE

Refer to Paragraph 8.1 for information regarding map configurations for 12K, 20K, and 28K systems.

8.3 ADVANCED MONITOR ENVIRONMENT

LOAD
GLOAD
DDT
DDTNS (DDT without symbol table)

Phase 1



The System Loader learns which I/O handlers are required by the Linking Loader, loads them relocatably, and then loads the Linking Loader relocatably.

The Linking Loader, during loading of user programs down from .SCOM+3 builds the loader (GLOBAL) and DDT (if DDT) symbol tables up from .SCOM+2.

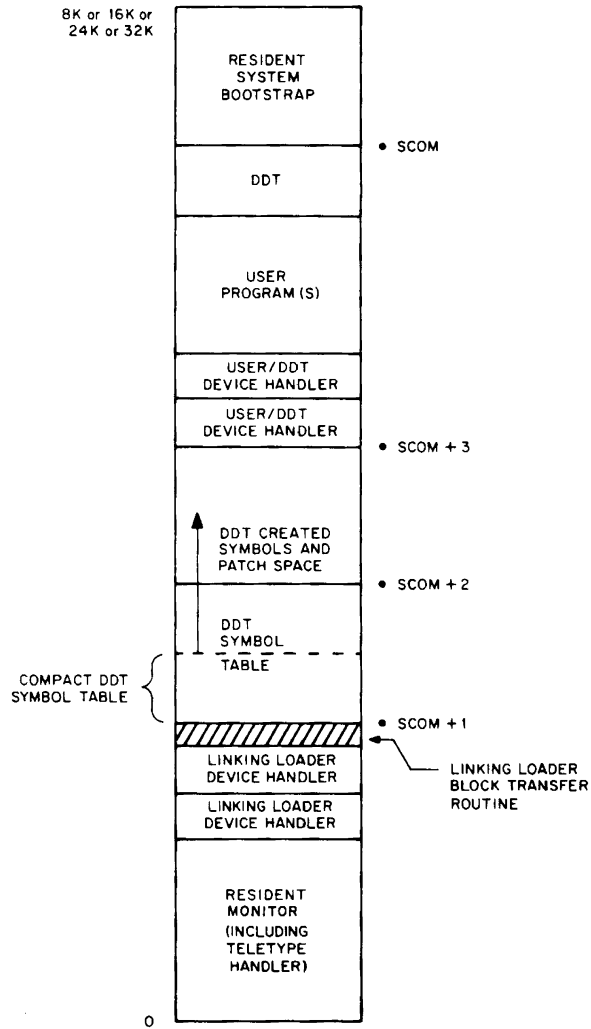
If a DDT load, the Linking Loader just prior to giving control to DDT moves the DDT symbol table down in core so that it overlays all of the Linking Loader except for the small routine that makes the block transfer.

The Linking Loader will not load a device handler that is already in core for its own use.

NOTE

Refer to Paragraph 8.1 for information regarding map configurations for 12K, 20K, and 28K systems.

Phase 2 (DDT or DDTNS)



.EXIT from the user program causes the system bootstrap to re-initialize the Advanced Monitor.

If a DDTNS load, no DDT symbol table is built.

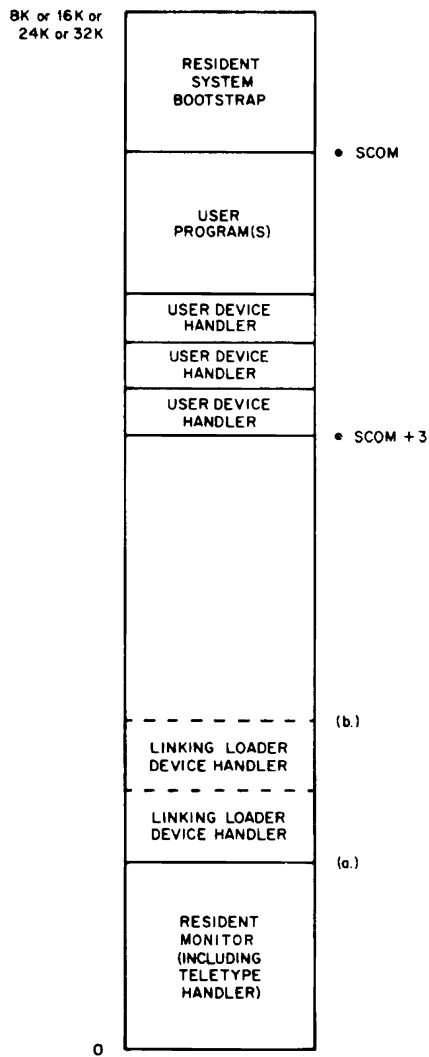
Non BLOCK DATA COMMON (FORTRAN IV or MACRO output) may make use of core as low as the compact DDT symbol table (DDT only retains certain symbol table entries). However, the user must be careful about placing patches.

The Linking Loader device handlers would have been used to satisfy user device requests.

NOTE

Refer to Paragraph 8.1 for information regarding map configurations for 12K, 20K and 28K systems.

Phase 2A (Not DDT or DDTNS)



09-0126

.EXIT from the user program causes the system bootstrap to re-initialize the Advanced Monitor.

.SCOM+1 and .SCOM+2 point to one of two places.

- If the user program did not have any device handlers in common with the Linking Loader.
- If the user program did have at least one device handler in common with the Linking Loader.

Non BLOCK DATA COMMON (FORTRAN IV or MACRO output) may make use of core as low as .SCOM+2.

NOTE

Refer to Paragraph 8.1 for information regarding map configurations for 12K, 20K, and 28K systems.

APPENDIX A SYMBOL CONCATENATION[†] - RADIX 50₈ FORMAT

Radix 50₈ is a technique used by the MACRO Assembler and the FORTRAN IV Compiler to condense the binary representation of symbolic names in symbol tables. Three characters, plus two symbol classification bits, are contained in each 18-bit word. A symbol is defined as a string of one to six characters, i.e.,

$$C_1 C_2 C_3 C_4 C_5 C_6$$

where any of the possible six characters (C_1 through C_6) can be defined as:

| Character | 6-bit octal code |
|-----------|------------------|
| Space | 00 |
| A | 01 |
| ↓ | ↓ |
| Z | 32 |
| % | 33 |
| . | 34 |
| 0 | 35 |
| ↓ | ↓ |
| 9 | 46 |
| # | 47 |

The characters which make up a symbol are linked together in the following manner:

$$\begin{aligned} \text{Word 1} & ((C_1 * 50_8) + C_2) 50_8 + C_3 \\ \text{Word 2} & ((C_4 * 50_8) + C_5) 50_8 + C_6 \end{aligned}$$

For example, the symbol SYMNAM would be entered in the Loader's symbol table as:

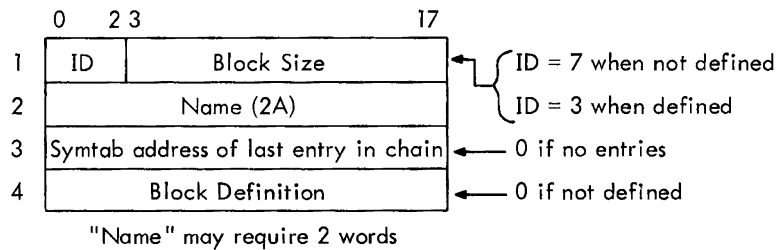
$$\begin{aligned} \text{Word 1} & ((23_8 * 50_8) + 31_8) 50_8 + 15_8 = 475265^{\dagger\dagger} \\ \text{Word 2} & ((16_8 * 50_8) + 1) 50_8 + 15_8 = 053665 \end{aligned}$$

[†]i.e., linking together

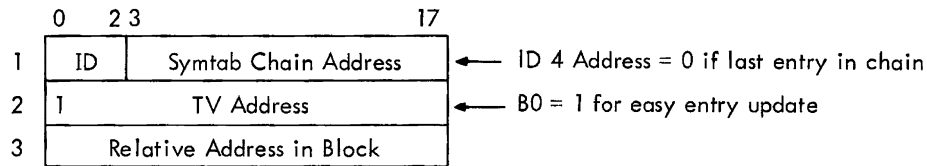
^{††}The sign bit of WORD 1 is set to 1 to indicate that this symbol consists of more than 3 characters and that the WORD 2 is necessary.

APPENDIX B LOADER SYMBOL TABLE

Common Block Name

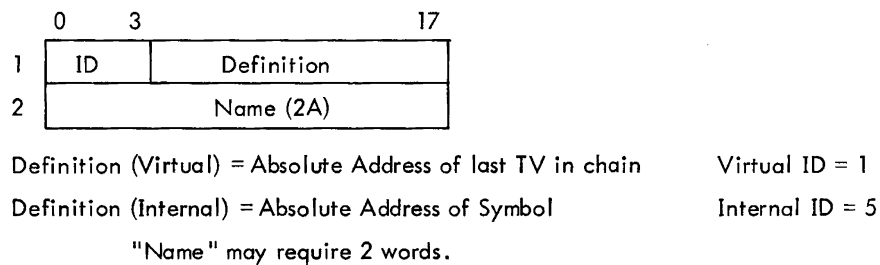


Common Name

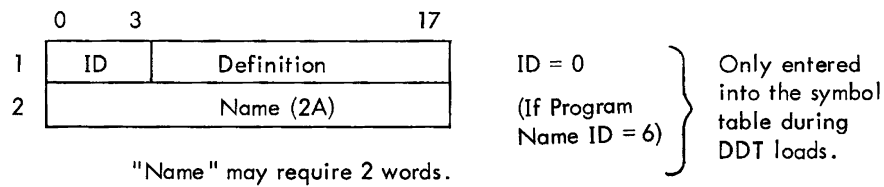


If associated COMMON block was defined when code 14 is encountered, no entry is needed in the symbol table.

Virtual Global (internal)



Internal Names



SECTION I
INTRODUCTION

1.1 PROGRAM DESCRIPTION

Chaining in the Advanced Monitor System is a method of segmentation which permits multiple overlays of executable code, constants, variables, arrays and labeled COMMON blocks. Core which is not overlaid is available for inter-segment communications (i.e., blank COMMON).

- a. CHAIN - a modified version of the Linking Loader which allows the user to build all the various segments (or chains) of his program into an executable (XCT) type file. CHAIN is relocatable.
- b. EXECUTE - a control program which initiates loading of an executable (XCT) file and transfers control from one segment (chain) to another. EXECUTE is absolute.

1.2 CALLING SEQUENCE

FORTTRAN IV

The statement CALL CHAIN (N) must be used to get from one program segment to another (i.e., chain to chain). The argument N is the number of the chain to be called; N can be greater or less than the current chain number but cannot equal it. Variables and arrays which are retained from chain to chain must appear in blank COMMON.

MACRO

The calling sequence for chaining in MACRO is:

```
.GLOBL  CHAIN
JMS*    CHAIN
JMP      .+2
.DSA    N
```

Where N is the address of a register containing the number of the chain to be called.

NOTE

All I/O must be completed (.CLOSEd) before CHAIN is called.

The scratch, or inter-segment communication, area available to MACRO written segments is delimited by system registers .SCOM+2 and .SCOM+3. The lower limit (.SCOM+2) remains constant throughout execution. The upper limit (.SCOM+3) is the first register below the currently resident segment and will vary unless all segments are of the same size. The total size of this free area is greater than the sum of the size of CHAIN and its I/O HANDLER(S) minus the size of EXECUTE'S I/O HANDLER.

SECTION 2

CHAIN SYSTEM PROGRAM

2.1 FUNCTIONAL DESCRIPTION

CHAIN is a relocatable system program which writes an XCT type file. Input consists of the standard relocatable binary (FORTRAN IV or MACRO output) with appropriate calls for segment loading.

The following .DAT slots are used by CHAIN:

| | |
|--------|----------------------------|
| .DAT-6 | Output of XCT file |
| .DAT-5 | External Library |
| .DAT-4 | User Program(s) |
| .DAT-3 | Control and Error Messages |
| .DAT-2 | Command String |
| .DAT-1 | System Library |

NOTE

If any of the bulk storage device handlers (e.g., DT or DK) are used, the same handler must be used for all appropriate .DAT slots. The different versions of handlers are unable to communicate with each other. For example, if input and output are to be to and from DECtape, DTA. (or DTB.) should be assigned to .DAT slots -6, -4, and -1.

The CHAIN program recognizes six commands (optional abbreviations are in parentheses):

- a. BUILD FILENM)
- b. CHAIN (C) N)
- c. FILE1, SUB1, etc.)
- d. END (E))
- e. CLOSE)
- f. EXIT)

BUILD FILENM - This command initiates the building of FILENM XCT onto .DAT-6. If no file name is given, an error message results. This command is legal only immediately after the typeout of CHAIN. If it is used at any other time, an error message results, and the BUILD command is ignored.

CHAIN(C) N - A chain with number N is begun at this point. N may be any positive, non-zero decimal number. It must be greater than any N given in a previous CHAIN command. This command is legal only after a BUILD or an END command. If it is used at other times, it is ignored, and an error message given.

FILE1, SUB1, etc. - All commands immediately following the CHAIN command and before the END command are interpreted as filenames to the Linking Loader portion of the XCT file builder. The following are illegal file names: BUILD, CHAIN, C, END, E, and CLOSE.

If these files are to be loaded from a nonfile-structured device (e.g., paper tape) the file names can be omitted. Only the number of files, N, need be indicated by typing N-1 separators.

NOTE

File names may be separated by comma, space, carriage return, or ALTMODE.

CLOSE - Finish building the file FILENM XCT, and restart CHAIN.

END(E) - Terminates the filenames used for a particular chain. This command must be used after a CHAIN command and with at least one filename between it and the CHAIN command. It can appear on the same line as the filenames, if so desired.

EXIT - Return to Monitor. In the case of the I/O Monitor, CHAIN halts. It can be restarted from this point by doing an I/O Reset and starting at location 363 (with the EXTEND switch up if the machine has more than 8K of core).

2.2 OPERATING PROCEDURES

The following procedures are used in the performance of CHAIN's operations.

a. Calling Procedure

(1) Advanced Monitor System

To load and start the CHAIN system program, type CHAIN after the Monitor's \$ request.

(2) I/O Monitor System

To load and start the paper-tape version of CHAIN, place the tape in the reader; depress the tape-feed switch to clear the end-of-tape flag; set the address switches to 17720 (for 8K or 12K), 37720 (16K or 20K), 57720 (24K or 28K), or 77720 (32K); and press I/O RESET and READIN.

When CHAIN is in core, the message

CHAIN
>

is typed on the teleprinter, and CHAIN waits for a command from the user.

b. General Command Characters

RUBOUT (echoes \)

Delete last character in current line of command string.
May be repeated n times to delete n characters in a line.

CTRL U (echoes @)

Delete entire line.

CTRL P (echoes †P)

Stop current program execution and restart the CHAIN
program at the beginning.

c. Restart Procedures

CLOSE

Finish building FILENM XCT, and restart CHAIN.

EXIT

Return to Advanced Monitor (or halt I/O Monitor System).

CTRL P (echoes †P)

Stop current program execution and restart the CHAIN
program at the beginning.

d. Example

CHAIN

>BUILD TEST ↵

Initiates building of file named TEST XCT

>CHAIN 1 ↵

First chain

>FILE1, SUB1, SUB2 ↵

Programs in this chain

>SUB3, SUB4 ↵

(First one is main program)

>END ↵

End of this chain

.

.

.

Memory allocation typeouts

.

.

.

.

.

.

>CHAIN 4 ↵

Last chain

>FILE4, SUBA ↵

>SUBB, END ↵

End of this chain

.

.

.

Memory allocation typeouts

.

.

.

>CLOSE

Terminate TEST XCT

CHAIN

>EXIT

Return to Advanced Monitor (or halt I/O Monitor System).

2.3 ERROR CONDITION TYPEOUTS

| <u>Message</u> | <u>Meaning</u> |
|--------------------------|--|
| ? | Illegal command |
| ILLEGAL DECIMAL DIGIT | Illegal decimal digit in chain number |
| ILLEGAL COMMAND ORDERING | Command entered is out of order and should not be used at this point. |
| ILLEGAL CHAIN NUMBER | Chain number is less than or equal to the last chain number. |
| ILLEGAL FILE NAME | File name used is same as a reserved command name. |
| .LOAD N Errors | Linking loader errors, which are terminal. CHAIN can be restarted, however, by typing control P. |
| .LOAD 1 | Memory overflow |
| .LOAD 2 | Input data error |
| .LOAD 3 | Unsatisfied Global symbol (missing program) |
| .LOAD 4 | Illegal .DAT slot requested by user program |

2.4 MEMORY ALLOCATION TYPEOUT

a. Load addresses of programs, subprograms, and library routines loaded

| | |
|------|------|
| FILE | XXXX |
| SUB | XXXX |
| LIB1 | XXXX |
| . | |
| . | |
| . | |
| LIBN | XXXX |

NOTE

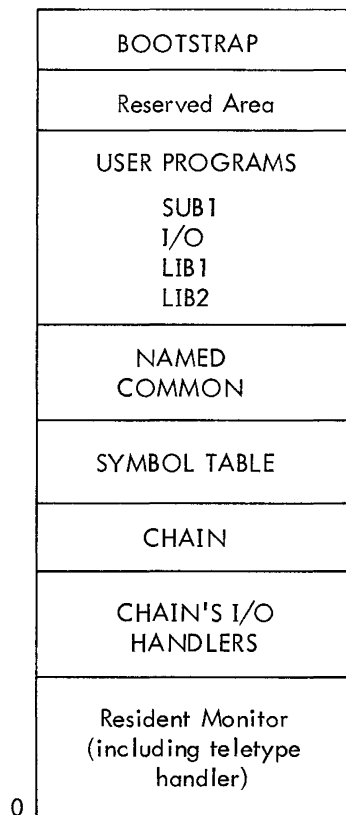
XXXX is the relocation factor in octal.

b. Special typeouts

| | | |
|---------|------|------------------------------|
| CHAIN# | N | (chain number in octal) |
| LOWEST | XXXX | (lowest register used) |
| | | (contents of .SCOM+3 plus 1) |
| | | (C(.SCOM+3)+1) |
| COMSIZE | XXXX | (size of blank common) |

2.5 MEMORY MAP AT CHAIN BUILD TIME (same for each chain)

top of core



{ occupied at run time by EXECUTE starting at 17077, 37077, 57077, or 77077

{ NAMED COMMON can go as low as the top of the resident Monitor and EXECUTE'S I/O Handler if there is no blank COMMON
Blank COMMON is allocated at EXECUTE time.

(PRA. and PPC. for the paper tape system)

NOTE

In systems with 12K, 20K or 28K of core, the bootstrap loader does not reside in the top of core. Instead, it is located in the top of the highest even 4K core segment. When user programs are loaded into core, avoid overlaying the bootstrap.

SECTION 3

EXECUTE SYSTEM PROGRAM

3.1 FUNCTIONAL DESCRIPTION

EXECUTE is an absolute bank relocatable system program which loads the first chain (i.e., segment) of an XCT type file and remains in core to load subsequent chains as they are called.

DAT SLOT USAGE

| | |
|---------|--|
| .DAT -4 | XCT type file input (use smallest IOPS BINARY input only handler (e.g., DTC.) to allow maximum core area for blank common) |
| .DAT-3 | Error messages |

3.2 OPERATING PROCEDURES

Calling Procedure (Keyboard Monitor System)

The command to load EXECUTE consists of either the letter E or the name EXECUTE and the name of the XCT type file which is to be run; it has the following form:

```
MONITOR
$EXECUTE FILEN )
    or
MONITOR
$E FILEN )
```

I/O Monitor System

To load the paper tape version of EXECUTE, place the tape in the reader; depress the tape-feed switch to clear the end-of-tape flag; set the address switches to 17720 (for 8K or 12K), 37720 (16K or 20K), 57720 (24K or 28K) or 77720 (32K); and depress I/O RESET and READIN. When EXECUTE is loaded, it types

RESTART INPUT & 1P

Put the paper-tape XCT file in the reader, depress the tape-feed switch and type CTRL P.

3.2.1 Special Operational Characteristics

a. COMMON SIZE

A warning message is given if the blank common size in a new chain is different from that of the previous chain.

WARNING - COMMON SIZE DIFFERS

b. NON BULK STORAGE INPUT

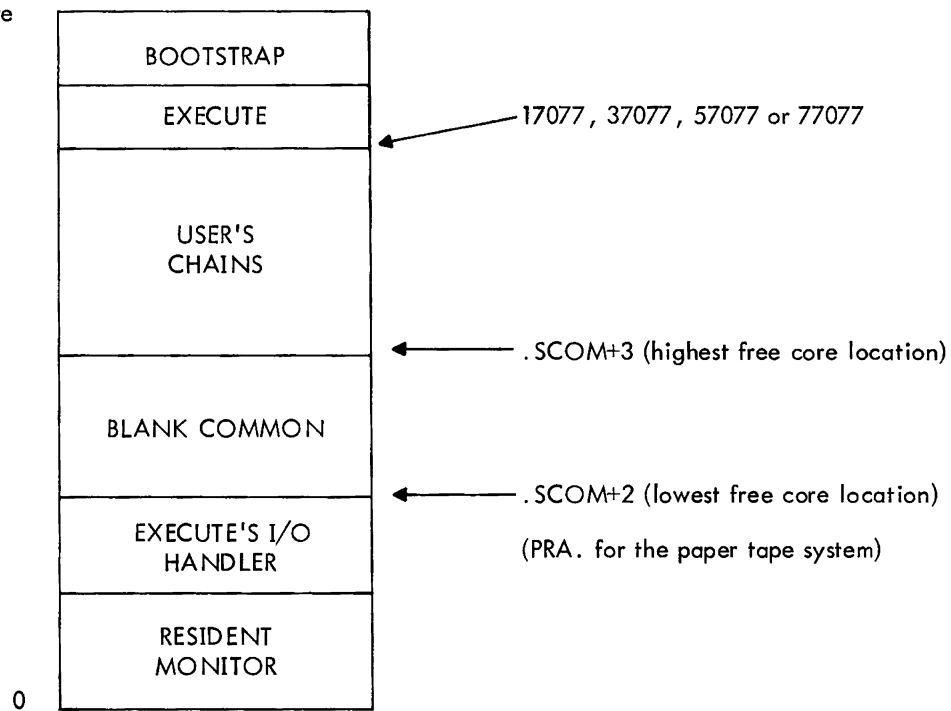
The following message is given if an attempt is made to call a chain number less than the current chain using non-bulk storage input:

RESTART INPUT & 1P

This means that the user should reload the XCT file in tape reader, depress the tape-feed switch, and type CTRL P.

3.3 MEMORY MAP AT EXECUTE TIME

top of core



NOTE

In systems with 12K, 20K or 28K of core, the bootstrap loader resides in the top of the highest even 4K core segment, which leaves 4K above the bootstrap. Execute is loaded into the top of core, above the bootstrap. User chain files are built to avoid overlaying the bootstrap.

SECTION 1 INTRODUCTION

The Peripheral Interchange Program (PIP) is a utility program in the ADVANCED Monitor System used to transfer data files from one standard peripheral device to another. PIP operates under Monitor control, using the Monitor I/O device handlers.

Files can be verified, renamed, deleted, combined or split into segments. Entire DECtapes, or individual DECtape blocks, can be copied and verified. File directories can be listed or initialized. Many of these functions and other subsidiary functions can be combined by inserting optional switches when the user types a command string to PIP.

The following peripheral devices can be used as either input (source) or output (destination):

| <u>Peripheral Device</u> | <u>Mnemonic</u> |
|---|---------------------------|
| DECtape (TC02 Control Unit with TU55 Transports) | DTn |
| Paper Tape Reader/Punch (PC15) | PR (Reader) PP (Punch) |
| Line Printer (Type 647) (output only) | LP |
| Teletype (KSR33 or KSR35) | TT |
| Card Reader (CR03B) | CR |
| Magnetic tape (TC59 Control Unit with TU20, TU20A, TU30, or TU30A Transports) | MTn |
| Disk System (Type RB09) | DKn |
| Disk System (Type RF15, RS15) | RFn |
| Disk System (Type RP15, RP02) | RPn |
| Display (VP15, 611 Storage Scope) | VP |

SECTION 2

DEVICE ASSIGNMENTS

Before using PIP, the user must be sure that the peripheral devices he plans to use are assigned to positive slots in the Monitor's Device Assignment Table (.DAT), because PIP scans this table for legal assignments other than teletype. Because .DAT slots -2 and -3 are assigned to teletype, PIP uses these for all teletype I/O. When typing a command string, the user specifies devices simply by writing mnemonic codes, such as DT2, PR, or TT.

2.1 BASIC I/O MONITOR SYSTEM

In paper-tape I/O Monitor systems, where the Device Assignment Table is fixed, the user need not be concerned with .DAT slot assignments. Table 2-1 contains the standard I/O Monitor assignments for PIP.

Table 2-1
I/O Monitor .DAT Slot Assignments

| .DAT Slot No. | Assignments |
|---------------|-------------|
| 1 | TTA |
| 2 | TTA |
| 3 | PRA |
| 4 | LPA |
| 5 | PPA |
| 6 | CDB |
| 7 | PPA |
| 10 | PRA |

2.2 ADVANCED MONITOR

In ADVANCED Monitor systems, the user must ensure that the devices he plans to use are assigned .DAT slots. The Monitor REQUEST PIP command is used to get a typeout of all current .DAT slot assignments. If a device he plans to use is not listed, an ASSIGN command is used to assign that device to any positive .DAT slot. The most complete handler (e.g., DTA, PPA, etc.) should normally be assigned. If the same device is to be used as both the source and destination device, it must be assigned to two positive .DAT slots.

Because these .DAT slot assignments are for use by PIP, the user need not remember them. Systems distributed by DEC initially have the assignments shown in Table 2-2.

Table 2-2
Initial .DAT Slot Assignments

| .DAT Slot No. | Assignment |
|---------------|------------|
| -3 | TTA |
| -2 | TTA |
| 1 | DTA0 |
| 2 | DTA1 |
| 3 | DTA2 |
| 4 | TTA |
| 5 | PRA |
| 6 | PPB |
| 7 | DTA1 |
| 10 | DTA2 |

SECTION 3

PIP COMMAND STRING: GENERAL

When PIP is in core memory (in an Basic Monitor or ADVANCED Monitor environment), it informs the user of its readiness to accept keyboard commands by outputting the following on the teleprinter:

```
PIP Vnn
>
```

The user can then type a command string to PIP on the same line as the right angle bracket (>). Successful completion and readiness for the next command is normally acknowledged by CR, LF, > unless there has been intermediate output to the teleprinter by PIP. In the latter case, the initial response (PIP, CR, LF, >) is output once again for ease of later printout examination. PIP command strings are of the following general form:

$$a \text{ dd } \left[\begin{array}{c} : \\ \text{SPACE} \end{array} \right] \text{ fname } \left[\begin{array}{c} ; \\ \text{SPACE} \end{array} \right] \text{ ext (x) } \leftarrow \text{ sd } \left[\begin{array}{c} : \\ \text{SPACE} \end{array} \right] \text{ fname } \left[\begin{array}{c} ; \\ \text{SPACE} \end{array} \right] \text{ ext (x) } \left[\begin{array}{c} \text{ } \\ \text{ALT MODE} \end{array} \right]$$

where:

- a = A single letter, specifying a PIP operation.
- dd = the destination device
- fname = file name
- ext = file name extension
- (x) = letter(s) specifying a PIP switch option(s)
- sd = source device

The left arrow (←) terminates information concerning the destination device. Data for the source device follows the ←. CR or ALTMODE must terminate a command string. ALTMODE forces PIP to return control to the Monitor upon successful completion of the command. CR causes PIP to wait for another command upon completion of the current one.

Multiple spaces are ignored by the command string processor. In fact, delimiters are only required following the operation character, device names, and file names.

Example:

```
T DT1  NEWNAM BIN  (B) ← DT2  OLDNAM BIN  ␣
or
T DT1:  NEWNAM;BIN  (B) ← DT2:  OLDNAM;BIN  ␣
```

The elements in the preceding example are:

| | |
|----------------|---|
| T | PIP Transfer File operation |
| DT1, DT2 | DECTape 1 is the destination device, DECTape 2 is the source device. |
| NEWNAM, OLDNAM | File names |
| BIN | File name extension |
| ← | Transfer direction indicator (right to left, i.e., DT2 to DT1) |
| B | Switch option |

3.1 OPERATION CHARACTER

The first character in a PIP command string must be an operation character defining the main function to be performed. It must be followed by a space. Legal operational characters are listed in Table 3-1.

Table 3-1
PIP Operation Characters

| | |
|--------------------|-------------------|
| (T) Transfer File | (V) Verify File |
| (L) List Directory | (S) Segment File |
| (D) Delete File | (B) Block Copy |
| (C) Copy | (N) New Directory |
| (R) Rename File | |

3.2 DEVICE NAME

Because the ADVANCED Software System provides more than one device handler for some peripherals, a two-letter mnemonic (normally corresponding to the first two letters of the handler name) is used for device name specification in PIP. Table 3-2 lists legal device names. For multi-unit peripherals (e.g., DECTape) the unit number, 0-7, appears after the device mnemonic (e.g., DT7). The device name delimiter must be a colon (:) or a space. The device name comprises three alphabetic characters, if desired (e.g., PRA, DTA7). This is permitted in PIP for command consistency within the system; i.e., the user, accustomed to typing \$ASSIGN DTAn to the Monitor, is relieved of learning a different convention for PIP.

Table 3-2
PIP Device Names

| | |
|------------------------|--------------------|
| (PR) Paper Tape Reader | (DT) DECTape |
| (PP) Paper Tape Punch | (MT) Magnetic Tape |
| (TT) Teletype | (CD) Card Reader |
| (LP) Line Printer | (RF) Disk |

3.3 FILENAME AND EXTENSION

Filename and extension, if used, constitute one element of the command string, where the filename delimiter is a semicolon (;) or space. If the extension is omitted, the default assumption is three full characters (refer to NOTE of paragraph 5.2). If more than one filename is specified, the second, third, etc., are separated from earlier names by commas (,). If the device is not a file-oriented device, file names can be omitted. Commas, however, must still be used for file count purposes. Some examples of device, filename, and filename extensions follow:

| | | |
|---------------------|-----------|------------------------|
| DT5:FILEA,FILEB;SRC | (2 files) | or DT5 FILEA,FILEB SRC |
| PR:,, | (3 files) | or PR <u> </u> , , |
| PP: | (1 file) | or PP |

A filename is a string of up to six alphanumeric characters. Any printing character in the ASCII set can be used with the exception of a space, (:), (;), (,), (() and ()), which have specific delimiter meanings to PIP. The filename extension can be up to three characters long.

3.4 SWITCH OPTIONS

Switch options are enclosed in parentheses and require no delimiters to separate them from each other. They may appear either with the destination device information or with the source device information. PIP switch options are divided into two classes: data modes and subsidiary operations.

3.4.1 Data Modes

| | |
|-----|--------------------|
| (A) | IOPS ASCII |
| (B) | IOPS Binary |
| (I) | Image Alphanumeric |
| (H) | Image Binary |
| (D) | Dump |

3.4.2 Subsidiary Operations

| | |
|-----|---|
| (G) | Display on teletype and allow correction of bad parity checksum lines |
| (E) | Convert tabs to spaces |
| (C) | Convert multiple spaces to tabs |
| (N) | New directory |
| (S) | New directory plus reservation of † QAREA |
| (Y) | Segment file |

- (W)[†] Combine several source files, or tapes, stripping .EOT's and .END's from intermediate tapes.
- or
- (W)[†] Combine several binary files, stripping EOF's from intermediate files.
- (F) Insert a form-feed (FF) and a Carriage-Return (CR) after every .EJECT statement or after every 57_{10} lines.
- (T) Delete trailing spaces from ASCII file.
- (Q) Delete sequence numbers from ASCII input lines during transfer (T) operations.

[†] .END and .EOT on the final ASCII tape and EOF of the final binary tape are retained.

SECTION 4

PIP FUNCTIONAL DESCRIPTION

Functionally, PIP can be described in terms of operations which are specified and subsidiary switch functions which are requested as parts of a given operation. All PIP operations and switches which are valid in the BASIC I/O Monitor paper-tape system are also valid in the ADVANCED and Background/Foreground Monitor systems. The converse is not true, however. Operations outlined below are described in detail and with examples, in Section 5.

4.1 OPERATIONS UNDER THE BASIC I/O MONITOR

Three PIP operations are provided in the BASIC I/O Monitor environment: Transfer File, Verify File, and Segment File.

4.1.1 Transfer File (T)

T performs basic data or file transfer from one I/O device to another. In the BASIC I/O Monitor environment T is used to copy paper tapes and list paper tapes or card decks on the teletype or line printer. T also provides the ability to create a source file by transferring from teletype to paper-tape punch. Paper tapes can be combined into one paper tape or segmented (IOPS ASCII tapes only) into several tapes.

4.1.2 Verify File (V)

The V operation allows parity/checksum verification of paper tapes. This function is particularly useful for verifying paper tapes copied with the T command.

4.1.3 Segment File (S)

The S operation provides a means for segmentation of bulk source paper tapes into two or more smaller tapes. All PIP operation commands are independent of other commands except Segment which is used prior to a Transfer command in order to specify at what points in the source file segmentation is to take place. The S command string can accommodate up to sixteen segmentation points or character strings (1-5 characters) at the beginning of lines at which segmentation is to take place. Each file is terminated with a .EOT just prior to the next segmentation point. Transfer continues to the next segmentation point, etc.

4.2 SWITCH OPTIONS UNDER THE BASIC I/O MONITOR

The data mode switches which may be used in the BASIC I/O Monitor environment are:

- (A) IOPS ASCII
- (B) IOPS Binary
- (I) Image Alphanumeric

Function switches for use under the BASIC I/O Monitor are:

- (G) Correct bad parity lines
- (E) Convert tabs to spaces
- (C) Convert multiple spaces to tabs
- (Y) Segment file
- (W) Combine files
- (F) Insert Form-Feed and Carriage -Return
- (T) Delete training spaces from ASCII files
- (Q) Delete sequence numbers from ASCII lines

Switch options can be used for some operations but are meaningless for other operations. Table 4-1 lists legal options by operation in the BASIC I/O Monitor environment. Furthermore, certain switch options conflict; e.g., combining the option to convert tabs to spaces (E) and spaces to tabs (C) is clearly a conflict. Table 4-2 lists legal switch combinations for the primary PIP operation, Transfer File.

Table 4-1
Legal Operation/Switch Combinations

| Operation | Legal Switches |
|-------------------|---------------------------------|
| Transfer File (T) | A, B, I, E, G, C, W, Y, F, T, Q |
| Verify File (V) | A or B |
| Segment File (S) | (None) |

Table 4-2
Legal Switch Combinations for Transfer File (T)

| Switches | Data Modes | | | Subsidiary Operations | | | | | | | |
|----------|------------|---|---|-----------------------|---|---|---|---|---|---|---|
| | A | B | I | E | G | C | W | Y | T | F | Q |
| E | ✓ | | | ✓ | | | ✓ | ✓ | | ✓ | |
| G | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| C | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| W | ✓ | ✓ | | ✓ | ✓ | | ✓ | | ✓ | ✓ | ✓ |
| Y | ✓ | | | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ |
| T | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| F | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Q | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

4.2.1 Image Alphanumeric (I)

The I data mode permits copying of any paper tape but, in particular, I must be used when copying tapes which are in Hardware Read-in Mode (HRM or RIM tapes). Thus, MACRO .ABS or .FULL tapes require the I data mode.

4.2.2 IOPS Binary (B)

Relocatable binary tapes are reproduced using the binary data mode B.

4.2.3 IOPS ASCII (A)

Source tapes are normally copied using the A data mode. It should be noted, however, that use of the A mode results in IOPS ASCII paper tapes having even parity in channel 8 of each frame. If for some reason this is undesirable to the user, a data mode of I is recommended. ASCII line buffers can accommodate the 120-character line printer length.

4.2.4 Bad Parity Correction (G)

Whenever data modes A or B are specified during a Transfer command, PIP automatically verifies the correctness of parity/checksum. The G switch, used with IOPS ASCII mode only, allows the user to modify erroneous input lines via teletype keyboard input. User intervention takes one of three forms: the line may be deleted, the line may be accepted, or the line may be replaced from the keyboard. The option to restart CTRL P is always available.

4.2.5 Tab to Space Conversion (E)

The E switch allows for conversion of horizontal tabs to spaces to allow off-line listing of ASCII tapes on Model 33 Teletypes. It is used with IOPS ASCII tapes. IOPS (Input/Output Programming System) follows a tenth position tab setting convention; thus, enough spaces are substituted for a tab to place the next printing character of the line in position 10, 20, 30, etc.

4.2.6 Space to Tab Conversion (C)

To condense an ASCII paper tape, the C switch is used to convert multiple spaces on an input file into horizontal tabs on the output file. Trailing spaces are simply deleted. Again, C is legal only when used with the A data mode.

4.2.7 Segment File (Y)

To apply the Segment operation during a Transfer file command, a Y switch is required in the T command string. On the basis of the Y switch, the IOPS ASCII input file is segmented into the number of output files specified in the preceding S command.

4.2.8 Combine Files (W)

Although combining files, or a series of paper tapes into one file, is most common when transferring from paper tape to a mass storage medium, it is possible to combine several small paper tapes into a single larger paper tape by indicating a W switch in a T command. Either IOPS binary or ASCII tapes can be so combined. For binary files, all but the final EOF block of the input tapes are discarded on output. Likewise, when combining a series of IOPS ASCII paper tapes, all .EOTs and .ENDs are stripped except that of the final input tape.

4.2.9 Insert Form Feed (F)

The F switch is used to automatically execute a form-feed (FF), followed by a carriage-return (CR), after each .EJECT statement or block of 57₁₀ lines if no previous FF had been encountered in the block. The F switch is used only with the Transfer T command in the data mode A.

4.2.10 Delete Trailing Spaces (T)

The T switch is used to delete trailing spaces from each line of an ASCII file. This switch is employed during a Transfer command T and data mode A.

4.2.11 Delete Sequence Numbers (Q)

The Q switch causes the deletion of all sequence numbers from input ASCII lines during the Transfer command. The switch is primarily intended for card conversion to some other medium. Columns 73-80 are deleted leaving a 72-character line with a carriage return as the 73rd character. The Q switch is legal only with A data mode in a Transfer command. It can be combined with C (spaces to tabs) or T (delete trailing spaces).

4.3 OPERATIONS UNDER THE ADVANCED OR BACKGROUND/FOREGROUND MONITOR

The presence of mass storage devices in a system configuration allows additional operations with PIP. In addition to the Transfer, Verify and Segment file operation, the following are available:

- | | |
|-------------------|-------------------|
| a. List Directory | d. Rename File |
| b. New Directory | e. Copy Tape, and |
| c. Delete File | f. Block Copy. |

(Additional switch options also become available as needed.)

4.3.1 List Directory (L)

The directory of any file-structured mass storage device can be listed on teleprinter or line printer with the L command. The filename, extension, starting block number, and number of blocks occupied are printed along with the number of system blocks and free blocks remaining.

4.3.2 New Directory (N)

The N command provides recording of a fresh directory on a mass storage device. In the case of DECtape, the File Bit Map blocks are cleared and the Directory block is initialized to indicate only the File Bit Map and Directory blocks as occupied.

4.3.3 Delete File (D)

To delete one or more named files from a mass storage device, the D operation is employed. Deletion implies removing references to the file from both the Directory and File Bit Map blocks.

4.3.4 Rename File (R)

Renaming one or more files requires an R command. Only the name and extension in the Directory are changed.

4.3.5 Copy Mass Storage Unit (C)

This function provides a convenient means of reproducing tapes (especially system tapes) in their entirety. Programmed read-after-write verification is performed.

4.3.6 Block Copy (B)

The block copy operation is used with DECtape or any mass storage device which is unit oriented to copy one or more blocks (e.g., when one or a few blocks on a tape seem suspect after a copy operation). There is no need to recopy an entire tape; blocks to be copied and verified are simply specified by their octal block number (0-1077).

4.4 SWITCH OPTIONS UNDER THE KEYBOARD MONITOR

Four additional switch options are available in an ADVANCED Monitor environment. Two are data modes: Image Binary (H) and Dump Mode (D). Two are subsidiary functions: New Directory (N) and New Directory with †QAREA (S).

Tables 4-3 and 4-4 summarize legal switch/operation combinations within an ADVANCED Monitor environment.

Table 4-3
Legal Operation/Switch Combinations

| Operation | Legal Switches |
|--------------------|-------------------------------|
| Transfer File (T) | A,B,I,H,D,E,G,C,W,Y,N,S,F,T,Q |
| Verify File (V) | A or B |
| Segment File (S) | (None) |
| List Directory (L) | N or S or None |
| New Directory (N) | S |
| Delete File (D) | (None) |
| Rename File (R) | (None) |
| Copy Tape (C) | N or S or H or None |
| Block Copy (B) | N or S or None |

Table 4-4
Legal Switch Combinations for Transfer File

| | | Data Modes | | | | | Subsidiary Operations | | | | | | | | | |
|--------------------------|---|------------|---|---|---|---|-----------------------|---|---|---|---|---|---|---|---|---|
| | | A | B | I | H | D | E | G | C | W | Y | N | S | T | F | Q |
| Subsidiary Operations | E | ✓ | | | | | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| | G | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | C | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | W | ✓ | ✓ | | | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Y | ✓ | | | | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | N | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | | |
| | T | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ |
| | F | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ |
| | Q | ✓ | | | | | | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ |

4.4.1 Image Binary (H)

The reader is referred to the applicable Software System manual (see Preface for list of applicable manuals) for a discussion of data modes. The use of Image Binary as a data mode on mass storage devices such as DECtape or disk implies the intent to retain the exact form of the binary data as it originally appeared in hard copy; e.g., paper tape such that, at a later time, the original data can be retrieved (once again onto paper tape) without alteration. It should be noted again that use of Image ASCII always exactly reproduces an identical tape, whether or not DECtape or disk has been used for intermediate storage (see Section 4.2.1).

The meaning of Image Binary as a switch option with the Copy (C) function is expanded beyond its customary meaning to imply a block-by-block DECtape copy. Later examples will illustrate this use of H mode.

4.4.2 Dump Mode (D)

Files recorded in dump mode reside on a mass storage device; thus, D is used as a data mode most frequently when transferring to and from mass storage. There is no restriction on its use from mass storage to paper tape, or vice versa, however.

4.4.3 New Directory (N)

The N switch option, like the N operation, initializes the Directory of the destination device. Permitting its use as a switch provides the added convenience of combining operations in a single command string.

4.4.4 New Directory With ↑QAREA (S)

As a function of installation core size, the S switch initializes the Directory of the destination device, reserving ↑QAREA blocks, as well as basic system blocks, for Directory and File Bit Map. For example, in an 8K system, 32₁₀ additional blocks are specified as occupied in the Directory.

SECTION 5

PIP COMMAND STRING

This section illustrates PIP commands and usage in detail. Basic I/O and ADVANCED Monitor environmental differences have previously been discussed, and no further mention is made here. Examples are given without the optional colon (:) and semicolon (;) delimiters for use with which the reader may refer to Section 3, page 3.

5.1 TRANSFER FILE (T)

The T command operations include listing, copying, creating, combining, and segmenting files. An input and an output device are required in the command string, as well as one of the five data modes. File names must be specified only for file-structured devices.

Filename extensions SRC or BIN can be omitted in the T command if data mode A or B is included in the command string (see NOTE of 5.2.). Conversely, data mode A is assumed if file extension SRC is used; B, if BIN is used.

5.1.1 Copying Files

The command:

```
T DT7 FILEA (A) + PR ,
```

copies a single tape from the paper-tape reader to DECtape unit 7 in IOPS ASCII mode.

The command:

```
T DT7 FA SRC,FB SRC,FC SRC +PR ,, ,
```

transfers three paper tapes as three separate files named FA SRC, FB SRC, and FC SRC.

The command:

```
T DT2 FILNEW (B) + DT1 FILOLD ,
```

not only transfers FILOLD BIN from DECtape unit 1 to DECtape unit 2 but also renames the file: FILNEW BIN.

5.1.2 Creating Files

Creating a file is normally an Editor function. However, a T command from teletype to any output device is perfectly legitimate. It should be kept in mind, however, that correction facilities provided by an Editor are not in PIP.

The command:

```
T PP (A) ← TT ↵
```

directs PIP to accept the input from teletype to be punched on paper tape. To terminate file creation, a final line consisting of ↑D (CTRL Key D) must be typed.

5.1.3 Listing Files

The command:

```
T LP ← RF FILNAM SRC ↵
```

lists FILNAM SRC on the line printer. IOPS ASCII is the only permissible mode to the line printer. Both IOPS and image ASCII are acceptable to the teletype, the alternate listing device.

5.1.4 Using the G Switch

PIP normally examines the correctness of parity and checksum when data mode A or B is specified. Transfer is discontinued after display of one of the two following messages on the teletype:

```
INPUT PARITY ERROR  
or  
INPUT CHECKSUM ERROR
```

The G switch, used only with data mode A, allows for user correction of an ASCII line with bad parity. Consider the following example:

```
T DT7 FILEA (AG) ←PR ↵
```

is typed. During transfer bad parity is encountered and the input parity error message is output on the teletype followed by the line in error. The user can:

- a. Accept the line by typing a carriage return.
- b. Delete the line by typing D ↵
- c. Retype the line, terminating with a carriage return.

- d. Abort the operation by typing $\uparrow P$ to restart PIP or $\uparrow C$ to reload the Monitor ($\uparrow C$ is ignored in BASIC Monitor environment).

5.1.5 Using the C or E Switches

The C or E switch are used only with A as the data mode. C and E can not be used together.

The command:

```
T DT7 FILEA SRC (C)  $\leftarrow$ PR  $\rightarrow$ 
```

effects a transfer from paper tape to DECtape, during which process all multiple spaces are converted to tabs and trailing spaces are deleted.

The command:

```
T PP (AE)  $\leftarrow$ DT2 FILEB  $\rightarrow$ 
```

effects a transfer of FILEB SRC from DECtape unit 2 to the paper-tape punch, during which process all tabs are converted to spaces allowing listing of the file on an off-line teletype which lacks a tabbing mechanism.

5.1.6 Using the N or S Switch

Initializing the directory of certain mass storage devices, e.g., DECtape, is a frequent operation. The N switch allows initialization within the context of a File transfer. S is the only switch which conflicts with N.

The command:

```
T DT4 FILEA IMG (IN)  $\leftarrow$ PR  $\rightarrow$ 
```

initializes the Directory and File Bit maps of DECtape unit 4 and, subsequently, transfers the paper tape file to DECtape in image ASCII mode.

The New Directory (N) command or switch causes 8 blocks to be reserved as file bit map blocks (required for DECtape 56-file capacity). These blocks are: the directory (block 100_8) and 7-file bit map blocks (71_8 through 77_8).

S switch - As a function of installation core size, (determined at system generation time), the S switch refreshes a directory, reserving $\uparrow Q$ area blocks as well as basic system blocks for directory and file bit maps. For example, in an 8K system, 32_{10} additional blocks (101_8 through 104_8) are specified as occupied in the directory (block 100_8).

S is a permissible switch (the only one) in a Newdir (N) command. Whereas,

N DT2

reserves only directory and file bit map blocks,

N DT2 (S)

additionally reserves \uparrow Q area blocks. S and N are mutually exclusive when both are used as switches in other PIP operations.

5.1.7 Using the W Switch

Source files are frequently of such size as to require several paper tapes. Although they may be maintained on a mass storage device in segmented form, it is more often desirable to combine the segments into one file. The W switch performs this function. It is legal with data modes A or B and conflicts with the Y switch.

The command:

T DT1 FILEA (AW)←PR ,,,,)

transfers five ASCII paper tapes to DECtape unit 1 as the single file, FILEA SRC. Because intermediate .EOT or .END pseudo ops are no longer useful, all but the one on the final tape are deleted during transfer.

Used with data mode B, the W switch provides a convenient way to combine several binary subprograms into a single file, e.g., a binary file.

The command:

T DT6 LIBRY BIN (W) ←DT1 A BIN, B BIN, C BIN)

combines the three binary files, A BIN, B BIN and C BIN into one file LIBRY BIN, deleting intermediate End-of-Files in the process.

5.1.8 Using the Y Switch

In contrast to the W switch which combines files, the Y switch is used when ASCII file segmentation is required. It is used only with data mode A and conflicts with the W switch. Given a sizable source file on mass storage which is to be segmented, the command:

T PP ,,,, (Y) ←DT1 FILIG SRC)

results in FILBIG SRC being split into six paper tapes where five segmentation points must have been specified in an S operation immediately preceding the current T command string (see Section 5.3).

The command:

```
T DT3 FA,FB ←RF FILBIG (AY) ↵
```

similarly segments the disk file FILBIG SRC into two smaller DECTape files, FA SRC and FB SRC. The preceding S operation will have specified one segmentation point.

As each output file is closed, PIP outputs ↑P, which is the restart request, on teletype. This is done to allow dismounting of tapes or removal of tape from the punch.

5.2 VERIFY FILE (V)

File verification is performed in either IOPS ASCII (A) or IOPS binary (B) data modes. No other switch options apply to the verify operation. There is no output; thus, only the input device (and filename if a file-structured device) need be specified.

The command:

```
V PR (B) ↵
```

requests parity and checksum verification of one binary paper tape. If a parity error occurs, the following message is typed:

```
INPUT PARITY ERROR
```

If there is a checksum failure:

```
INPUT CHECKSUM FAILURE
```

For an ASCII file, the error line is also printed. In either case, after the message is printed, verification continues until the entire file has been examined allowing the user to assess how many errors are present.

Multiple files may be verified in a single command string. For example, the command

```
V DT3 FILEA SRC,FILEB SRC ↵
```

requests verification of both FILEA SRC and FILEB SRC.

NOTE

To simplify even more the user interface to PIP, file name extensions SRC and BIN may be omitted in the Transfer (T) or Verify (V) commands if the appropriate data mode switch (A or B) appears in the command string, for example:

T DT2 (A) ←DT1 FILE 》

effects a transfer of FILE SRC to DT2. If program FILE (no extension) exists on DT1, it will be transferred.

V DT1 FILE (A) 》

causes FILE SRC to be verified.

Conversely: if SRC or BIN appears as an extension in the command string (T or V), data mode A or B (which-ever applies) is assumed, for example,

T DT2 ←DT1 FILE SRC 》

is the same as T DT2 (A) ←DT1 FILE SRC 》

is the same as T DT2 (A) ←DT1 FILE 》

5.3 SEGMENT FILE (S)

The S operation allows specification of up to 16 file segmentation points. Device names, file names, and switch options are all meaningless in the S command.

The segmentation points are specified as one to five character strings. In the subsequent T command, if a Y switch is specified, PIP examines the beginning of every line for the specified segmentation points in order of occurrence. Vertical form control characters at the beginning of a line are excluded from the string search. As each segmentation point is found, PIP closes the current output file segment, appending the pseudo op, .EOT, at the end of the segment. The next segment starts with the line which begins with the current segmentation point.

The command:

S TAGA,TAGB,TAGC,TAGD 》

sets up the four segmentation points TAGA,TAGB,TAGC, and TAGD for the immediately following command:

T PP ,,,, (AY) ←DT3 FILBIG SRC 》

the end result is five paper tapes; all but the last of which are terminated with .EOT. The last four begin with the lines TAGA---, TAGB---, etc.

As each segment is completed, ↑P is output on the teleprinter, allowing time to remove the paper tape segment, dismount tapes, etc. When ready to continue, the user simply types CTRL P and PIP resumes segmentation.

5.4 LIST DIRECTORY (L)


The Directory contents of any file-structured device can be listed by the L operation. The N or S switch options can be used.

The command:

```
L TT ←DT1 ↵
```

results in a printout such as the one below:

```
DIRECTORY LISTING
115 FREE BLKS
3 USER FILES
10 SYSTEM BLKS
MACRO ONE 4 226 ←# of Blocks Occupied
MACRO TWO 5 140
MACRO SRC 6 365
```

 1st Block of File

5.5 NEW DIRECTORY (N)

Although the N function can be performed as a switch option in the command string of another operation, it is useful to include it as a distinct operation. The S switch is the only switch option used with the N command, and only the destination device need be specified.

The command:

```
N DT4 ↵
```

results in a fresh directory on DECtape unit 4, a listing of which (as requested by an L operation) appears as follows:

```
DIRECTORY LISTING
1070 FREE BLKS
0 USER FILES
10 SYSTEM BLKS
```

The 10₈ system blocks are the Directory and seven File Bit Map blocks.

5.6 DELETE FILE (D)

File deletion is performed by the operation D. Only the destination device is specified. No switch options are used.

The command:

```
D DT3 FILEA,FILEB 》
```

causes PIP to delete both FILEA and FILEB from DECtape unit 3. If an extension is present, e.g., SRC or BIN, it must be typed in the Delete command.

5.7 RENAME FILE (R)

The R command is used to rename files on a file-structured device without data transfer of any kind. No other unit is needed, although the device name must appear with both source and destination data. A simple nine-character name substitution takes place into the directory entry section of the directory block. All switch options are illegal.

The command:

```
R DT2 NEWNAM BIN ←DT2 OLDNAM BIN 》
```

changes the name of the file OLDNAM BIN ON DECtape unit 2 to NEWNAM BIN.

5.8 COPY MASS STORAGE UNIT (C)

Copying the contents of one file-structured device onto another implies one of two tasks:

- a. incorporation of all information on the input device into the organization and content of the output device
- b. total replacement of all information on the output device by information on the input device.

The Copy (C) command with H switch specified (that is a direct unit copy in its entirety) uses all of the available core memory. The fastest possible copy (using H switch) is obtained by assigning DTD (for DECtape) to all possible .DAT slots prior to loading PIP.

The command:

```
C DT5 (H) ←DT3 》
```

replaces all data on DECtape unit 5 with data from unit 3 in a block-by-block copy and read after write.

Incorporation is effected in one of the following three ways:

- a. Absence of switch options in the C command:

C DT5 ←DT3

All files on DT3 will be incorporated into the file organization of DT5.

- b. Use of the N Switch:

C DT5 (N) ←DT3

Prior to transferring the files on DECtape unit 3 to unit 5, the Directory and File Bit Maps of unit 5 will be initialized.

- c. Use of the S Switch:

C DT5 (S) ←DT3

Prior to transferring the files on DECtape unit 3 to unit 5, the Directory and File Bit Map of unit 5 will be initialized and a ↑QAREA will be reserved.

5.9 BLOCK COPY (B)

To copy one or more blocks of one DECtape onto another, the B command is used. Switches N or S can be used within the command string also. Instead of specifying file names, actual octal block numbers (0-1077) are given in the command string. These block numbers can appear either with the destination or source data and are separated by commas.

The command:

B DT7 ←DT4 5, 15, 165, 1075

or

B DT7 5,15,165,1075 ←DT4

requests copy and verification of blocks 5,15,165 and 1075 from DECtape unit 4 to unit 7.

The command:

B DT2 10,15 ←DT1 4,3

requests blocks 4 and 3 of DECtape unit 1 to be copied (and verified) onto blocks 10 and 15 of DECtape unit 2, respectively.

SECTION 6

CORRECTION PROCEDURES

There are four correction procedures used with PIP:

- a. Aborting the task
- b. Deleting one or more characters in a command string
- c. Negating the entire command string
- d. Responding with corrected command string information after a PIP error message.

The procedure chosen is largely a function of what point in the PIP process the user decides to correct or to modify PIP action.

6.1 ↑P (CTRL KEY P)

A task can be aborted and PIP restarted by typing in the ↑P (CTRL P) character at any time. ↑P also indicates loading of the next in a series of paper tapes or output of the next in a series of files during segmentation. For example, at the end of each of several paper tapes to be combined into one output file, PIP types ↑P on the teleprinter, directing the user to load the next paper tape. When ready, the user types ↑P for PIP to continue.

6.2 RUBOUT (RO)

During typing of a command string, one or more characters can be deleted by use of the rubout (RO) key. For each character deleted, starting with the last one typed, a back slash (\) is echoed. For example:

```
T ZT3\\\ DT3 FILEA (A) ←PR ↵
```

The character Z is in error. Three rubouts have been used to back up to the erroneous character.

6.3 ↑U (CTRL KEY U)

At any point while typing a command string, that is, prior to the CR or ALTMODE, a ↑U can be typed to delete the entire command string up to that point. An "at" sign (@) is echoed. The user then starts from the beginning of the command string again.

The command:

```
T ZT3@T DT3 FILEA (A) ←PR ↵
```

demonstrates a ↑U correction.

6.4 PIP Error Detection and Correction

After a command string is completed, PIP checks for erroneous information. If erroneous information is found, an appropriate error message is output to the teleprinter and the questionable command string is output up to, but not including, the offending character or element followed by "?", requiring correct completion by the user. If the user prefers to retype the command, a carriage return or ↑P signals PIP to accept a new command from the beginning. The characters RO and ↑U should not be used, because it is the teletype handler (which no longer has access to the erroneous command string), not PIP, which interprets and acts upon RO and ↑U.

Appendix II contains a complete list of PIP error messages. Only two examples are cited here. Suppose a user intends to transfer an IOPS ASCII file from paper tape to DECTape. He types:

```
T DT2 FILEA SRC (P) ←PR ↵
```

Recognizing P as an illegal switch option, PIP types:

```
ILL SWITCH  
T DT2 FILEA SRC(? ↵
```

The user can either complete the command string from the erroneous character or element on to the end, or use an ↑P to indicate he prefers to restart the message.

If a DECTape handler is not assigned to any of the positive .DAT slots, PIP types in response to the above example:

```
DEV NOT IN + .DAT TABLE  
T ? ↵
```

to indicate that the command was in error (could not be honored due to absence of the necessary .DAT assignment) at the point of the device and unit specification code.

APPENDIX A
SUMMARY OF PIP COMMANDS

Basic I/O Monitor Environment

| Command | Abbrev. | Dest. Dev. | Source Dev. | File Names | Legal Switches |
|---------------|---------|------------|-------------|------------|---------------------------|
| Transfer File | T | Yes | Yes | No | A,B,I,E, G,C,W,Y,F,T,Q |
| Verify File | V | No | Yes | No | A or B |
| Segment File | S | No | No | No* | None |

ADVANCED and Background/Foreground Monitor Environment

| Command | Abbrev. | Dest. Dev. | Source Dev. | File Names | Legal Switches |
|----------------|---------|------------|-------------|------------|-----------------------------------|
| Transfer File | T | Yes | Yes | Yes | A,B,I,H,D,E, G,C,W,Y,N,S,F,T,Q |
| Verify File | V | No | Yes | Yes | A or B |
| Segment File | S | No | No | No* | None |
| List Directory | L | Yes | Yes | No | N or S |
| New Directory | N | Yes | No | No | S or None |
| Delete File | D | No | Yes | Yes | None |
| Rename File | R | Yes | Yes | Yes | None |
| Copy Tape | C | Yes | Yes | No | N or S or H |
| Block Copy | B | Yes | Yes | No** | N or S |

*Segmentation points instead of file names.

**Block numbers instead of file names.

SECTION 1
INTRODUCTION

Two utility programs are described in this section: a) DSKPTR, which performs the loading of Disk systems from paper tape; b) DSKSAV, which controls the saving and loading of Disk system software via DECtape.

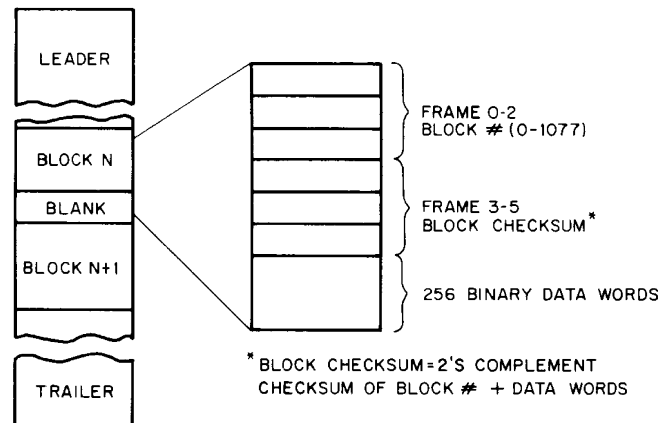
DSKPTR UTILITY PROGRAM

In PDP-15 systems which have a Disk but no DECtape units, the system software is provided to the user in the form of a group of 24 paper tapes. The DSKPTR Utility Program performs the operations needed to load the system from paper tapes.

In addition to its system load function, DSKPTR also enables any logical disk unit or 6K segment of these units to be either loaded from paper tape or its contents punched out onto paper tape. These additional functions are controlled according to the settings of the console AC Switches.

Paper Tape Data Format (See Figure 1)

The data format of tapes generated or loaded via DSKPTR is as follows: each tape consists of 6K or 24 logical 256_{10} word sequential disk blocks. Each data block on paper tape is preceded by 6 blank frames to aid visual identification.



15-0067

Figure 1-1 Paper Tape Block Format

SECTION 2

DSKPTR OPERATING PROCEDURE

The following steps describe the procedure required to load the Disk system via paper tape, the possible errors, and the action to be taken for each error condition.

| <u>Step</u> | <u>Procedure</u> |
|-------------|--|
| 1 | Set Disk unit control panel WRITE PROTECT switch to the OFF position (down). |
| 2 | Place the paper tape of DSKPTR in reader with address switches set to 17720. |
| 3 | Press I/O RESET and READIN. |
| 4 | When the program is loaded, it will display its title, DSKPTR and skeletal directions on teletype. <div style="margin-left: 100px;"> SET: ACS0=0[†] Paper tape to disk ACS15-17 = Unit# 0, 1, 2, 4, 5, 6 </div> |
| 5 | Place the first disk system tape in reader and press CONTINUE. |
| 6 | "HIT CONTINUE FOR NEXT TAPE" will be typed on the teletype when the tape has been read correctly. Load the remaining 23 tapes in a similar fashion. Note, any number of tapes (up to the full 24) can be loaded in any order. |

The following errors may be displayed on the teletype during loading:

- a. Unit Error: ACS15-17 = 3 or 7.
 Action: (1) Reset ACS15-17 = 0, 1, 2, 4, 5, 6
 (2) Press CONTINUE
- b. Reload Tape: Reader end-of-tape condition at illegal point.
 Action: (1) Reload tape
 (2) Press CONTINUE
- c. Input Checksum Error: Block checksum incorrect.
 Action: (1) Reload tape from beginning, OR position tape to the front of block in error, or leave tape in current position.^{††}
 (2) Press CONTINUE

[†]ACS0 - 17 = 0 for full system load onto disk.

^{††}Data block in error will be accepted as is.

| <u>Step</u> | <u>Procedure</u> |
|-------------|--|
| 7 | When the disk system is loaded, set the WRITE PROTECT switch and the three rightmost track switches to the ON position (up). |
| 8 | Load the Disk Bootstrap into the reader with address switches set to 17637. |
| 9 | Press I/O RESET and READIN. Monitor will be loaded from the disk and will type: |

\$ MONITOR Vnn

The system is now ready for operation. If the disk diagnostics have been run or if data on other areas of the disk is suspect, all user disk unit directories must be refreshed at this point. This is done with the MONITOR Newdir (N) command, for example,

\$ N 1

refreshes disk unit 1. This command should be issued for units 1, 2, 4, 5, and 6 before further operation.

DSKPTR can be used to punch out 6K areas of the disk, as follows:

| <u>Step</u> | <u>Procedure</u> |
|-------------|---|
| 1 | Repeat Steps 1, 2, and 3 of loading procedure. |
| 2 | SET: ACS0 = 1 ACS15-17 = Unit # ACS5-14 = Logical Block # 0, 30, 60... 1050 ^{††} |
| 3 | Press CONTINUE |
| 4 | Repeat Step 6 of loading procedure. |

The following errors may be displayed on teletype during disk to punch output:

- a. Unit Error: (See loading procedure)
- b. Reload Tape: Punch out of tape
 - Action: (1) Reload punch
 - (2) Press CONTINUE
 - (3) Splice Tape

^{††} ACS5-14 = 0 causes punching of all 2410 tapes for disk unit (ACS15-17). ACS5-14 = NON 0 causes punching of one 6K paper tape starting at the logical block specified. This feature is particularly useful when PATCH is used for system program modification and a single back up paper tape of the associated disk area is desired.

- c. Disk Error: Probably parity error. AC will contain disk status. Look at disk control panel for AC bit meaning.

Action: Since DSKPTR will have tried 8 times to read disk

- (1) Reset at 16000 or
- (2) Press CONTINUE to accept data.

SECTION 3

DSKSAV UTILITY PROGRAM

Once a Disk system has been generated, it is valuable to produce a backup system on DECTape to permit rapid disk restoration if it is needed. Utility program, DSKSAV, is used to read and store (save) the generated Disk Advanced Software System and/or the contents of other logical Disk units, and to load the stored information back onto the disk when necessary.

When used for Logical Disk Unit save/load functions, DSKSAV is controlled by the settings of the console AC switches.

DSKSAV Save/Load Operating Procedures

The following steps describe the procedure required for the program save/load operations, the errors which could occur, and the actions to be taken in response to each type of error.

| <u>Step</u> | <u>Procedure</u> |
|-------------|---|
| 1 | Place paper tape of DSKSAV in reader with address switches set to 17720. (Restart = 16000) |
| 2 | Press I/O RESET and READIN. |
| 3 | When the program is loaded, it will type its title, DSKSAV, and brief directions on the console teletype. The program is stopped to allow ACS settings. <div style="margin-left: 100px;"> Set: ACS0=0[†] DECTape to Disk (LOAD) ACS0=1 Disk to DECTAPE (SAVE) ACS15-17=Unit# 0, 1, 2, 4, 5, 6 </div> |
| 4 | Ensure that the DECTape unit selection is identical to ACS 15 through 17, and press CONTINUE. |

The following errors may be displayed on the teletype during loading:

- a. Unit Error: ACS15-17 = 3 or 7.
Action: (1) Reset ACS 15-17 = 0, 1, 2, 4, 5, 6
 (2) Press CONTINUE
- b. Disk Error: Probably illegal disk address (for example, attempt to load disk with
 WRITE PROTECT switch enabled).
 AC contains disk status. Examine disk control panel for AC bit meaning.
Action: Correct disk problem if possible and press CONTINUE.

[†]ACS0-17 = 0 for full system load onto disk.

- c. DECtape Error: Probably end zone error. AC contains DECtape status. Examine DECtape control panel for AC bit meaning.

Action:

- (1) If DECtape is in the forward end zone, press CONTINUE.
- (2) If DECtape is in the far end zone (beyond block 1077), position tape out of end zone, and press CONTINUE.
- (3) If parity or mark track error[†], press CONTINUE to restart transfer from the beginning, or set ACS0 through 17 = 0 and press CONTINUE to retry read one more time.

If DSKSAV is being used to load the Monitor system from DECtape to Disk, the following procedure should be followed for system startup:

| Step | Procedure |
|------|--|
| 1 | When the disk system is loaded, set the WRITE PROTECT switches for tracks 0 through 29. |
| 2 | Load the Disk Bootstrap into the reader with address switches set to 17637. |
| 3 | Press I/O RESET and READIN. The Monitor will be loaded from the disk and the message is typed out. \$ MONITOR Vnn |

The system is now ready for operation. If disk diagnostics have been run or if data on other areas of the disk is suspect, all user disk unit directories must be refreshed at this point. This is done with the MONITOR Newdir (N) command, for example,

\$ N 1 ↵

refreshes disk unit 1. This command should be issued for units 1, 2, 4, 5 and 6 before further operation.

NOTE

DSKSAV can also be used to save or load other logical units of the disk. This provides rapid DECtape backup of any or all disk data files. To do so, follow Step 1 and 2 above under loading procedure.

[†] DSKAV will have tried to read four times before the message is printed.

SECTION 1 INTRODUCTION

The PUNCH utility program permits the user to output, in an I/O Monitor environment, a selected core area and a .ABS loader onto paper tape. The three primary applications of this utility program are:

- a. System program modification or patching;
- b. .DAT slot reassignment;
- c. Production of an executable user program core load on a single paper tape. This application is useful for core loads which consist of a relocatable main program and several subprogram/library routines.

The PUNCH utility program is provided as a .FULL tape which is loaded starting at location 17720 of the highest available core bank.

The output of a PUNCH operation may be from one to nine .ABS tapes (each preceded by a .ABS loader) which are loaded in a Hardware READIN Mode (HRM) starting at 17720 of the highest available core bank. The areas of core which are punched out are 0 to the address in .SCOM+2 and the address in .SCOM+3 to the address just below the .ABS loader (17717 of the highest core bank).

1.1 EQUIPMENT REQUIREMENTS

An 8K system equipped with a high-speed reader/punch is the minimum hardware configuration for running PUNCH. The only option which affects the operation of PUNCH is the addition of core (up to 32K). Additional memory permits larger core areas to be punched out.

1.2 SOFTWARE REQUIREMENTS

The I/O Monitor system is required for proper operation of PUNCH. Although PUNCH does not require execution of code in the I/O Monitor, it does use parameters in the I/O Monitor to determine the amount and core limits of its punched output.

In particular, PUNCH references the following I/O Monitor cells:

| | |
|-----------|-----------------------------------|
| .SCOM | -first register below .ABS loader |
| .SCOM + 2 | -lowest free register |
| .SCOM + 3 | -highest free register |
| .SCOM + 5 | -system program start address |

| | |
|-----------|-----------------------------|
| .SCOM + 6 | -user program start address |
| .KM9PT | -I/O Monitor start address |
| TTUSRR | -↑P address in I/O Monitor |
| TTDDTR | -↑T address in I/O Monitor |
| TTIOSW | -Teletype I/O busy switch |

The .SCOM registers are fixed at cell 100 and following. The remaining four, however, are positionally variable as a function of I/O Monitor assembly. Hence, any reassembly of the I/O Monitor which changes the location of .KM9PT, etc., requires reassembly of PUNCH in which .KM9PT, etc., are redefined by parameter assignment from the keyboard at assembly time (i.e., a re-edit is not required).

1.2.1 Resident Programs

In addition to the I/O Monitor System, the system program or relocatable user programs must be in core for meaningful operation of PUNCH.

1.3 PROGRAM ORGANIZATION

The PUNCH utility program is divided into two parts:

- a. Part 1 of the program operates in core locations 17720 and above (where the .ABS loader normally resides). Part 1 has the function of loading and relocating Part 2 of the program.
- b. Part 2 resides in "free" core starting at the address pointed to by .SCOM+2; its function is to perform the desired output punch operations.

SECTION 2

OPERATING PROCEDURES

General descriptions of the procedures used in PUNCH operations are given in the following paragraphs. For detailed step-by-step procedures, refer to the 15/20 User's Guide.

2.1 LOAD AND PUNCH

The steps required to load PUNCH and output the desired tapes are as follows:

| <u>Step</u> | <u>Procedure</u> |
|-------------|--|
| 1 | Load required system/relocatable users program. |
| 2 | Load PUNCH at address 17720 of the highest core bank. |
| 3 | When loaded, the utility program outputs ↑P, ↑T or ↑S |

The user responds to this output by typing the appropriate letter to designate the starting location of the program to be punched:

- a. ↑T if DDT or DDTNP is part of the core load;
- b. ↑P for all other system or users programs which have already initialized the teletype (.INIT) with a restart address at the time PUNCH was loaded.
- c. ↑S only if the core load was output by PUNCH after linking loader operation at the moment at which the loader was expecting the ↑S.

On completion of the user response, the program causes a carriage-return/line-feed operation and outputs another >.

| <u>Step</u> | <u>Procedure</u> |
|-------------|---|
| 4 | The user must type the number of output tapes desired (i.e., 1 to 9). |

NOTE

A carriage-return is interpreted as the number 1 by the program.

Step

Procedure

5

On completion of Step 4, the program initiates the punching of the desired core load; halts at the end of each punch tape of the desired series (with the exception of the last tape) to permit the tape to be removed from the bin. Depress the control console CONTINUE switch to restart the output operations. After the final tape is punched, the PUNCH program restarts at the point of command string input.

2.2 ERROR DETECTION AND RECOVERY

Error detection is minimal in PUNCH. If any character other than tP, tT or tS is typed when those characters are expected; the program simply restarts. The value expected for number of tapes (1-9) is assumed correct. The low order bits of the ASCII character are simply ANDed off and used.

Punch-out-of-tape condition is always checked. If the punch is out of tape, "RELOAD PTP" is output to the teletype and PUNCH halts. Once the punch is reloaded, depressing CONTINUE causes the remainder of the tape to be punched. It may then be spliced to the earlier section.

SECTION 3

EXAMPLES

Listed below are three procedural examples for use of PUNCH.

3.1 PRODUCING AN EXECUTABLE USER PROGRAM TAPE

To produce a single paper tape of a relocatable user program, subprogram and/or library programs, the following method is used.

| <u>Step</u> | <u>Procedure</u> |
|-------------|--|
| 1 | Load Linking Loader and user programs, subprograms, etc. Refer to applicable system User's Guide (see Preface for list). |
| 2 | At the point where the Linking Loader expects ↑S, stop the computer and load PUNCH (HRM: 17720 of highest core bank). |
| 3 | When the PUNCH query: "↑P, T or S ?" appears on the teleprinter, type ↑S. |
| 4 | Type the number of output tapes desired. |
| 5 | To load the resultant tape, depress I/O Reset and READ-IN (17720 of the highest core bank) for each tape. The computer stops after loading each tape except the last. After the last tape is loaded, program execution begins automatically. |

3.2 SYSTEM PROGRAM CHANGES

Whenever system program changes or corrections are published, the user can implement them using PUNCH. MACRO is used as an example.

| <u>Step</u> | <u>Procedure</u> |
|-------------|--|
| 1 | Load the two .ABS MACRO tapes (HRM: 17720 of the highest core bank). |
| 2 | After MACRO is loaded and is awaiting a command from teletype, stop the computer and modify the required core locations. |
| 3 | Load PUNCH (HRM: 17720 of the highest core bank). |
| 4 | When the PUNCH query "↑P, ↑T or ↑S?" appears on the teleprinter, type ↑P. |
| 5 | Type the number of output tapes (2 for MACRO - more if desired). |
| 6 | Proceed as in Step 5 under 3.1. |

3.3 MODIFYING USER .DAT (Device Assignment Table) SLOTS

Although it is not possible to reassign negative .DAT slots (i.e., those used by system programs) at load time in the I/O Monitor environment (reassembly is required), it is possible to modify positive .DAT slots which are intended for relocatable user programs.

Table 3-1 lists the standard positive .DAT slot assignments. Two examples will be cited: a. reassignment for use of a device whose handler(s) is provided in the I/O Library[†]; b. reassignment for use of a device whose handler(s) is not provided in the I/O Library, i.e., a user generated handler.

Table 3-1
Standard Paper Tape .DAT[†] Slot Assignments

| .DAT Slot | Handler |
|-----------|---------|
| 1 | TTA |
| 2 | TTA |
| 3 | PRA |
| 4 | TTA |
| 5 | PPA |
| 6 | PRA |
| 7 | PPA |
| 10 | PRA |

3.3.1 Using An Alternate I/O Library Handler

Given a user who has written a relocatable program to perform drum I/O using handler DRA, the Linking Loader (or DDT or DDTNP if he plans to use DDT) must be repunched by PUNCH to allow loading of handler DRA.

| <u>Step</u> | <u>Procedure</u> |
|-------------|---|
| 1 | Load the Linking Loader (HRM 17720 of the highest available core bank). |
| 2 | Stop the computer, and modify the appropriate .DAT slot cell according to the Table 3-2 below. For example, suppose the user has chosen .DAT slot 7 for DRA ^{††} and intends to use drum unit 1. Cell 144 should be changed from 000004 (present value) to 100014 (the high-order 3 bits specify the unit #; the low-order bits, the loader code for DRA). |
| 3 | Proceed as in Step 3 of 3.2. |

[†].DAT table (cell 0) begins at location 135.

^{††}Note: The MACRO user program will require inclusion of .DAT slot 7 in the .IODEV statement.

Table 3-2
Loader - I/O Correspondence Table

| .DAT Slot Value | Handler |
|-----------------|---------|
| 1 | TTA. |
| 2 | PRA. |
| 3 | PRB. |
| 4 | PPA. |
| 5 | PPB. |
| 6 | PPC. |
| 7 | LPA. |
| 10 | CDA. |
| 11 | CDB. |
| 12 | CDC. |
| 13 | MTF. |
| 14 | DRA. |
| 15 | DRB. |
| 16 | DRC. |
| 17 | DRD. |

3.3.2 Using A Non-Standard I/O Handler

The following example is concerned with

- a. integration of a user generated I/O for a special or non-standard device[†] into the standard I/O Library;
- b. the procedure for modifying the paper-tape Linking Loader (or DDT or DDTNP) to recognize the new handler.

Appendix A illustrates the core memory map of the I/O Monitor skip chain when the Linking Loader (and ultimately the user program) is in memory. It is important to examine Appendix A because, in addition to .DAT slot modification, skip chain modification is required.

Given that a user is willing to apply one of the standard handler names to his own handler (see Table 3-2 above), no other changes to memory are required. He must position his own handler in the set of I/O Library tapes before the other one of same name. If, however, a user prefers a distinct name, he must change a radix 50 value in the Loader I/O Configuration (IOC) Table (see Appendix B and C). If changed, the new name (three alphabetic characters) must be terminated by a "." (for example, XYZ.). The procedural method is outlined below:

| Step | Procedure |
|------|---|
| 1 | Load the Linking Loader (HRM 17720 of the highest core bank). |
| 2 | Stop the computer, and modify the appropriate .DAT slot cell according to Table 3-2. Suppose .DAT slot 4 is chosen and the user wishes to substitute his plotter handler for handler LPA. Cell 141 should be changed from 000001 (present value) to 000007 (the Loader code for LPA). |

[†]For example, an incremental plotter, type 350.

Step

Procedure

2
(Cont)

The skip associated with LPA. is at location 1533 (see Appendix A). Change Cell 1533 from 706501 to 702401 (LPSF). If the plotter handler has been named LPA., proceed to Step 3 below.

If the user prefers a different name, e.g., PLA., he must modify the Loader IOC table (see Appendix B). Having chosen code 7 above for .DAT slot modification, he must modify the radix value in slot 7 of the IOC Table, i.e., location 4507 in the Linking Loader or DDTNP or location 5266 in DDT. To compute the radix 50 value of PLA.: find P in Appendix C, column 1 (062000), L in column 2 (000740), and A in column 3 (000001). The resultant radix 50 value = 062741. A sign bit of 1 indicates to the Linking Loader a fourth character = . Hence, Cell 4507 (or 5266 if DDT) should be changed to 462741.

3

Proceed as in Step 3 of 3.2.

APPENDIX A
I/O MONITOR SKIP CHAIN[†]

| Location | Contents | Mnemonic | Device Meaning |
|----------|----------|------------|--------------------------|
| 1511 | 703201 | SPFAL | POWER FAIL |
| 1512 | 741000 | SKP | |
| 1513 | 621577 | JMP* INT6 | |
| 1514 | 706101 | DRSF | DRUM DONE |
| 1515 | 741000 | SKP | |
| 1516 | 621603 | JMP* INT12 | |
| 1517 | 707341 | MTSF | MAGTAPE DONE |
| 1520 | 741000 | SKP | |
| 1521 | 621604 | JMP*INT13 | |
| 1522 | 700001 | CLSF | CLOCK OVERFLOW |
| 1523 | 741000 | SKP | |
| 1524 | 600476 | JMP CLKPIC | |
| 1525 | 706701 | RCSF | CARD COLUMN READY |
| 1526 | 741000 | SKP | |
| 1527 | 621575 | JMP* INT4 | |
| 1530 | 706721 | RCSD | CARD DONE |
| 1531 | 741000 | SKP | |
| 1532 | 621576 | JMP* INT5 | |
| 1533 | 706501 | LSDF | LINE PRINTER DONE |
| 1534 | 741000 | SKP | |
| 1535 | 621572 | JMP* INT1 | |
| 1536 | 700101 | RSF | PAPER TAPE READER DONE |
| 1537 | 741000 | SKP | |
| 1540 | 621573 | JMP* INT2 | |
| 1541 | 700201 | PSF | PAPER TAPE PUNCH DONE |
| 1542 | 741000 | SKP | |
| 1543 | 621574 | JMP* INT3 | |
| 1544 | 700301 | KSF | KEYBOARD READY |
| 1545 | 741000 | SKP | |
| 1546 | 601110 | JMP TIINT | |
| 1547 | 700401 | TSF | TELEPRINTER DONE |
| 1550 | 741000 | SKP | |
| 1551 | 601344 | JMP TOINT | |
| 1552 | 701741 | MPSNE | NON-EXISTENT MEMORY |
| 1553 | 741000 | SKP | |
| 1554 | 621600 | JMP* INT7 | |
| 1555 | 701701 | MPSK | MEMORY PROTECT VIOLATION |
| 1556 | 741000 | SKP | |
| 1557 | 621601 | JMP* INT10 | |
| 1560 | 702701 | SPE | MEMORY PARITY ERROR |

[†] This skip chain memory map applies only when the Linking Loader, DDT, DDTNP, CHAIN, EXECUTE, or re-locatable user programs are in core.

I/O MONITOR SKIP CHAIN[†] (Cont)

| Location | Contents | Mnemonic | Device Meaning |
|----------|----------|------------|-----------------------------|
| 1561 | 741000 | SKP | NOT DRUM ERROR (PDP-9 only) |
| 1562 | 621602 | JMP* INT11 | |
| 1563 | 706201 | DRNEF | |
| 1564 | 621605 | JMP* INT14 | |

[†] This skip chain memory map applies only when the Linking Loader, DDT, DDTNP, CHAIN, EXECUTE, or re-locatable user programs are in core.

APPENDIX B
LINKING LOADER IOC TABLE

| Location Loader or DDTNP/DDT | Contents | Handler |
|---------------------------------|----------|-----------|
| 4501 / 5260 | 500041 | TTA. (1) |
| 4502 / 5261 | 463321 | PRA. (2) |
| 4503 / 5262 | 463322 | PRB. (3) |
| 4504 / 5263 | 463201 | PPA. (4) |
| 4505 / 5264 | 463202 | PPB. (5) |
| 4506 / 5265 | 463203 | PPC. (6) |
| 4507 / 5266 | 446601 | LPA. (7) |
| 4510 / 5267 | 411541 | CDA. (10) |
| 4511 / 5270 | 411542 | CDB. (11) |
| 4512 / 5271 | 411543 | CDC. (12) |
| 4513 / 5272 | 452146 | MTF. (13) |
| 4514 / 5273 | 415721 | DRA. (14) |
| 4515 / 5274 | 415722 | DRB. (15) |
| 4516 / 5275 | 415723 | DRC. (16) |
| 4517 / 5276 | 415724 | DRD. (17) |

APPENDIX C
RADIX 50₈ VALUES

| X-- | | -X- | | --X | |
|-----|--------|-----|--------|-----|--------|
| A | 003100 | A | 000050 | A | 000001 |
| B | 006200 | B | 000120 | B | 000002 |
| C | 011300 | C | 000170 | C | 000003 |
| D | 014400 | D | 000240 | D | 000004 |
| E | 017500 | E | 000310 | E | 000005 |
| F | 022600 | F | 000360 | F | 000006 |
| G | 025700 | G | 000430 | G | 000007 |
| H | 031000 | H | 000500 | H | 000010 |
| I | 034100 | I | 000550 | I | 000011 |
| J | 037200 | J | 000620 | J | 000012 |
| K | 042300 | K | 000670 | K | 000013 |
| L | 045400 | L | 000740 | L | 000014 |
| M | 050500 | M | 001010 | M | 000015 |
| N | 053600 | N | 001060 | N | 000016 |
| O | 056700 | O | 001130 | N | 000017 |
| P | 062000 | P | 001200 | P | 000020 |
| Q | 065100 | Q | 001250 | Q | 000021 |
| R | 070200 | R | 001320 | R | 000022 |
| S | 073300 | S | 001370 | S | 000023 |
| T | 076400 | T | 001440 | T | 000024 |
| U | 101500 | U | 001510 | U | 000025 |
| V | 104600 | V | 001560 | V | 000026 |
| W | 107700 | W | 001630 | W | 000027 |
| X | 113000 | X | 001700 | X | 000030 |
| Y | 116100 | Y | 001750 | Y | 000031 |
| Z | 121200 | Z | 002020 | Z | 000032 |
| % | 124300 | % | 002070 | % | 000033 |
| . | 127400 | . | 002140 | . | 000034 |
| 0 | 132500 | 0 | 002210 | 0 | 000035 |
| 1 | 135600 | 1 | 002260 | 1 | 000036 |
| 2 | 140700 | 2 | 002330 | 2 | 000037 |
| 3 | 144000 | 3 | 002400 | 3 | 000040 |
| 4 | 147100 | 4 | 002450 | 4 | 000041 |
| 5 | 152200 | 5 | 002520 | 5 | 000042 |
| 6 | 155300 | 6 | 002570 | 6 | 000043 |
| 7 | 160400 | 7 | 002640 | 7 | 000044 |
| 8 | 163500 | 8 | 002710 | 8 | 000045 |
| 9 | 166600 | 9 | 002760 | 9 | 000046 |
| # | 171700 | # | 003030 | # | 000047 |

SECTION 1 INTRODUCTION

The DUMP utility program provides the user with the ability to output, on any available listing device, specified core locations which have been stored on a bulk storage device using the CTRL Q (tQ) command.

1.1 OPERATING PROCEDURES

1.1.1 Calling Procedure

The Dump program is called by typing DUMP 2 after the Monitor's \$ request. When the Dump program has been loaded, it types

```
DUMP
>
```

on the teletype and waits for a command from the user.

1.1.2 General Command Characters

| | |
|-------------------|--|
| RUBOUT (echoes \) | Delete last character in command string. Can be repeated n times to delete n characters. |
| CTRL U (echoes @) | Delete entire line |

1.1.3 Command String

The formats expected by the DUMP command string processor are as follows.

| <u>Command</u> | <u>Function</u> |
|---|---|
| ALL | The entire tQ area (from location 10 to the address in .SCOM) on the device associated with .DAT slot -14 (at tQ time, this device was the specified output device) is listed on the device associated with .DAT slot -12. |
| XXXXX-YYYYY (XXXXX > 10, and YYYYY ≤ C (.SCOM)) | The tQ area between absolute addresses XXXXX and YYYYY on the device associated with .DAT slot -14 is listed on the device associated with .DAT slot -12. At tQ time, this device (.DAT slot -14) was the specified output device and XXXXX and YYYYY were the absolute (octal) bounds of the core area to be dumped. |

| <u>Command</u> | <u>Function</u> |
|----------------|--|
| ZZZ# | The content of block #ZZZ on the device associated with .DAT slot -14 is listed on the device associated with .DAT slot -12. The block number is in octal radix. For DECTape systems, if the command ZZZ# is followed by a minus sign (-), block number ZZZ is read in the reverse direction and dumped. |

NOTE

If the Listing output (.DAT slot -12) is to be a file-oriented device, the file is named MEMORY and has the extension DMP.

1.2 ERROR CONDITIONS

Any unrecognizable command causes a question mark (?) to be typed on the teletype. Control is then returned to the command string processor which types > to indicate its readiness for a command.

1.3 RESTART PROCEDURES

If a command is terminated by a carriage return (↵), control returns to the command string processor after completion of the request.

```
DUMP
>
```

is printed on the teletype indicating readiness for another command.

If a command string is terminated by the ALT MODE character, control returns to the Monitor upon completion of the request.

1.4 EXAMPLE

To dump locations 16730 through 16750:

```
MONITOR
$ASSIGN DTD0 -14 ↵
$DUMP ↵
DUMP
>16730-16750↵
16730 000032 003740 013777 000000 000000 413420 013422 463356
16740 127400 463356 127400 000612 003766 003773 000000 020202
16750 000000
DUMP
>
```

SECTION 1 INTRODUCTION

1.1 GENERAL INFORMATION

The Dynamic Debugging Technique (DDT) program provides convenient on-line debugging assistance for MACRO and FORTRAN programmers. By typing simple commands on the teletype keyboard, programmers can make corrections and additions in symbolic code (or octal), suspend execution of the program at any predetermined point during the debugging run, and examine the status of any memory word in the program. The user's program is started and stopped by commands to DDT. Under normal conditions, the user is always able to stop a "runaway" program.

DDT operates as part of the Advanced Software System. It is loaded into memory (the top 1600₁₀ positions) along with the Linking Loader which, on command, loads the user's program (including the symbol table and any sub-programs) and the needed I/O handlers, FORTRAN Object Time System routines, and library subroutines.

All user communication with DDT is via the teletype (any model included in the standard system configurations).

DDT interprets all numeric input, and outputs all numeric data, in octal radix. The digits 8 and 9 are treated as alphabetic characters.

1.2 TERMINOLOGY USED

- ␣ A non-printing character used for text representation of the carriage return key.
- ␣ A non-printing character used for text representation of the line feed key.
- ↑ A text representation of the CTRL key, always used in conjunction with another key. It is also the printing character, up arrow.
- ↑T The non-printing character obtained by holding the CTRL key while striking the T.
- + Elements are to be added.
- Elements are to be subtracted.
- ␣ (Space) field delimiter, as between operation code and address.

The term C (R) represents the content of storage word R.

In examples, underscoring designates information typed by DDT.

A Transfer Vector is a word which contains the 15-bit address of another word. Bits 0 through 2 are meaningless and can be used for codes. Transfer vectors are also used in indirect addressing, by the Linking Loader for sub-routine calls, and are required in addressing to another memory bank.

SECTION 2 DEBUGGING WITH DDT

2.1 LOADING THE PROGRAM

In an I/O Monitor (paper tape) environment, the Linking Loader forms an integral part of the DDT tape.

In the Keyboard Monitor, the teletype command DDT (DDTNS) calls the Linking Loader as well as DDT. (DDTNS is used to prevent loading of the user symbol table to save memory.)

The first response from the teletype, in either system, is:

```
LOADER  
>
```

The user programs are then loaded in the usual manner (see Section 2.1 and 2.2 of the Linking Loader manual). Control can be transferred to DDT at this point without loading any user programs by typing control T. When loading is complete, DDT takes control and types:

```
DDT  
>
```

to indicate its readiness to accept DDT commands.

With the Keyboard Monitor, .DAT slots -4 (user programs) and -5 (user external library, if any) must be assigned to appropriate devices for proper loading.

If the input is file-oriented, the Loader assumes BIN (binary) as the filename extension.

2.2 USING THE BREAKPOINTS

A breakpoint provides a convenient means of interrupting a user program at any predetermined step to examine the program status. DDT inserts a breakpoint (on request) by replacing the indicated instruction with a jump to DDT. When the program reaches that point, control shifts to DDT, which types the number of the breakpoint; the address of the breakpoint; the contents of the AC; the status of the Link; and the go-ahead signal (>). The user then performs any of the debugging operations explained herein.

DDT allows the use of four breakpoints to facilitate debugging when there is uncertainty as to which path the program will follow.

The user can place a breakpoint at any point in this program, subject to the following limitations:

- a. Instructions which are program modified
- b. Instructions which are used as literals
- c. Breakpoints should not be placed in routines which operate with API active. If such a breakpoint is encountered, DDT types:

API 4XXXXX (API status register)

and the normal breakpoint information. The exclamation point (!) has no effect at that breakpoint (i.e., a restart from the breakpoint is not possible). The breakpoint will not be removed; any other DDT command is valid at this point including a command to remove this breakpoint.

Breakpoints can be placed on skip, jump, and JMS instructions. Breakpoints can also be placed on CAL or XCT instructions. CAL instructions may, however, contain arguments required by the called subroutine, as well as a variable number of subsequent arguments; consequently, DDT is unable to simulate the CAL (as it is able to simulate a JMS). Therefore, a breakpoint which has been placed on a CAL is removed by DDT before continuing (exclamation point command). DDT retains the request for a breakpoint at that location, and restores it if another breakpoint is entered and exited. XCT instructions may execute CAL instructions, and therefore, they are treated identically. If the user wants to place a breakpoint at a CAL, and restore it after each stop, a second breakpoint can be placed at the return from the CAL, as shown in this example.

```
LOC    CAL 3            (Breakpoint 1)
         12
         LAC BUFF       (Breakpoint 2)
```

On leaving the second breakpoint at LOC +2, the breakpoint on the CAL instruction is restored.

Operation of breakpoints requires one auto-index register; DDT initially assumes register 17. The user can specify any other auto-index register by modifying DDT's special register, AX\$, as follows:

AX\$/ 000017 10 2 (Modification procedure is explained later)

The commands controlling breakpoints are as follows:

k _ n" Causes a breakpoint to be inserted at location k. The number n (1-4) is assigned to that breakpoint.

- n" Causes the breakpoint assigned the number n to be removed.
- " Causes all existing breakpoints to be removed.

The insertion of a breakpoint takes place when control returns to the user program. The break occurs before execution of the instruction at the breakpoint address.

Examples:

- LOC + 1 □ 1" Inserts a breakpoint at LOC + 1
- TAG □ 2" Inserts a breakpoint at TAG
- 1" Removes breakpoint number 1

A breakpoint number can be reassigned without first removing the previous assignment.

To restart from a breakpoint, the user simply types an exclamation point (!). DDT restores the AC and Link and returns control to the user's program, starting with the instruction at the breakpoint address. An octal number typed before the exclamation point causes DDT to bypass that breakpoint n times. This is convenient when a breakpoint has been inserted in a program loop, and the user does not wish to stop every time through the loop.

If the user's program does not reach the breakpoint, the operator may stop the action and return control to DDT by typing control T (hold the CTRL key down while striking the T). DDT types the go-ahead (>). The program interrupt control must be "on" to perform this operation.

2.3 EXAMINATION AND MODIFICATION

DDT provides several variations of the procedure for examining and modifying the contents of any storage word. They are:

- k/ The slash, typed after an address (k) causes the addressed storage word to be opened and its contents displayed on the teleprinter. For example,

LOC/ TAD COUNT

where the instruction TAD COUNT is contained at the location labeled LOC. The storage word is now opened and may be modified by typing the desired content and issuing one of the commands described below.

The carriage return closes the storage word and resets DDT, enabling it to accept other commands. Any change which has been entered is incorporated, as shown below:

LOC/ TAD COUNT↵

TAG/ JMP LOC JMP LOC+1↵

- ↓ The line feed closes the storage word, then opens the next sequential storage word:

LOC/ TAD COUNT ↓

LOC+1/ CMA

- ↑ The up arrow closes the storage word, then opens the preceding storage word.
- LOC/ TAD COUNT ↑
LOC+1/ LAC A
- ↑Z Control Z allows the user to examine (and modify) a single storage word, out of sequence, and then return to the original sequence. This command closes the storage word, then opens the referenced storage word. A line feed will then open the next storage word in the original sequence, as shown:
- LABEL/ JMP LOC ↑Z
 LOC/ TAD COUNT TAD CNTR ↓
LABEL+1/ LAC HOLD
- ↑A Control A allows the user to examine a new sequence of storage words. This command closes the storage word, then opens the referenced storage word, establishing a new sequence. A line feed then opens the second storage word in the new sequence.
- LABEL/ JMP LOC ↑A
 LOC/ TAD COUNT TAD CNTR ↓
LOC+1/ CMA
- ↑X Control X is used, in conjunction with transfer vectors, to examine a new sequence of storage words. This command operates with a 15-bit address taken directly from the currently open word. (In contrast, the ↑Z and ↑A operations take 13 bits from the currently open word and the two memory bank bits from the address of the open storage word.)
- TAG/ 36307 ↑X
36307/ 000000

2.4 TYPE-OUT MODES

DDT allows the user to choose from several modes of representing the requested information. These modes, and their commands, are as follows:

- NUM\$ In this mode, DDT types memory word contents as six-digit octal numbers, including any leading zeroes.
- TV\$ In this mode, DDT interprets words as transfer vectors. Bits 0 through 2 are ignored, and bits 3 through 17 are interpreted according to the address modes as described below.
- SYM\$ In this mode, which is assumed initially, DDT interprets words as symbolic instructions. Bits 0 through 3 are first examined to determine the instruction class. If bit 4 (indirect addressing bit) of a memory reference instruction is set, an asterisk (*) is typed after the mnemonic op code. The address portion is handled according to the address mode as described below. Operate instructions are further examined for specific mnemonic codes. (See Appendix B for recognized codes.) Operate instructions not found in DDT's table are typed out as NOP+XXXX. Subroutine calls, extended arithmetic element, and input/output instructions are interpreted as CAL+XXXX, EAE+XXXX, and IOT+XXXX, respectively.

: The colon, typed after a word has been displayed in either numeric (NUM\$) or symbolic (SYM\$) mode, causes DDT to retype the word in the alternate mode.

LOC/ TAD LABEL : 340126

or LOC/ 340126 : TAD LABEL

= The equal sign, typed after a word has been displayed in either numeric or symbolic mode, causes DDT to retype the word as a transfer vector.

LOC/ CAL+126 = LABEL

2.4.1 Address Modes

The following commands set the address mode, which affects the handling of transfer vectors, address portions of memory reference instructions, and display addresses.

REL\$ In this mode, which is assumed initially, DDT types addresses which are relative to user defined symbols.

LOC/ TAD LABEL-3

If there is no symbolic label within $\pm 77_8$ positions, the address is typed as relocatable (see next paragraph below). Symbols defined in direct assignments are not recognized by DDT.

RLC\$ In this mode, DDT types addresses in relocatable form, as shown on the assembly listing. For example,

LOC/ TAD 147

ABS\$ In this mode, DDT types addresses in absolute form:

LOC/ TAD 13147

The difference between the results of RLC\$ and ABS\$ modes is the relocation factor (in this case, 13000). The relocation factor is found in the memory map output by the Loader.

The user may type modification input in whatever representation he finds most convenient. There are, however, two considerations:

a. If a memory reference mnemonic is entered with a numeric address, DDT assumes that address to be relocatable unless the address output mode has been set to ABS\$. For example,

LOC/ TAD COUNT TAD 147 ↓

(DDT adds the relocation factor before storing the information).

b. A requested address, typed numerically, is always considered absolute.

41/ Opens word 41 of the machine.

13000+41/ or 13041/ Opens word 41 of the program, where 13000 is the relocation factor.

2.5 STARTING AND RESTARTING

DDT receives control, initially, from the Monitor and normally regains control from the user's program by means of a breakpoint, as previously described. A CTRL T can be typed at any time (if the program interrupt control is enabled) to restore control to DDT.

The following commands shift control from DDT to the user's program:

- ' The apostrophe, typed alone, starts the user's program at its normal starting address. (i.e., that address given in the source .END statement, or the first physical location of the first program loaded).
- k' The user can start his program at any other point by simply typing that address ahead of the apostrophe.
- ! The exclamation point restarts the user's program after a breakpoint. The AC and the Link are restored before continuing.
- n! An octal number (n) entered before the exclamation point causes DDT to bypass that breakpoint n times before stopping again. This ability is useful when a breakpoint has been placed in a program loop.

2.6 SEARCHING OPERATIONS

DDT has a powerful searching operation to easily find every word in a user's program having particular characteristics. Two special locations, LO\$ and HI\$ (further explained in the next section), control the limits of the search, and a mask (MSK\$) allows the search to be based on all or any portion of the word. The mask is initially set at 777777, for a full word search; and the limits are initially set to encompass the entire user's program, including all subprograms and library routines.

There are three types of searches, as follows:

- k ⊆ EQ\$ Starts a search for all words, within the set limits, the contents of which, after masking by C(MSK\$), are equal to the expression k.
- k ⊆ UN\$ Starts a search for all words, within the set limits, the contents of which, after masking by C(MSK\$), are not equal to the expression k.
- k ⊆ ADR\$ Starts a search for all memory reference instructions, within the set limits, with effective addresses which, after masking by C(MSK\$), are equal to the address k. Indirect addressing is followed one step.

Examples:

LOC+1 ⊆ ADR\$ might produce

| | |
|---------------|-------------------|
| <u>TAG/</u> | <u>LAC LOC+1</u> |
| <u>POS/</u> | <u>XOR LOC+1</u> |
| <u>LABEL/</u> | <u>DAC* POINT</u> |

If the > is typed with no other output, the search routine has found no qualifying words.

2.7 SPECIAL LOCATIONS USED BY DDT

The following special locations contain information useful to the user, and which he may wish to change.

| | |
|-------|---|
| AC\$ | Holds C(AC) at a breakpoint. |
| LNK\$ | Holds status of the Link at a breakpoint. |
| MSK\$ | Contains the search mask, initialized at 777777. |
| LO\$ | Contains the address of the lower limit of the search operation. |
| HI\$ | Contains the address of the upper limit of the search operation. |
| PA\$ | Contains the address of the first position available for inserting patches. (Note that the initial contents of LO\$ show the last available position plus one.) |
| AX\$ | Contains the number of the auto-index register to be used by the breakpoint routines, initialized to 17. |
| RF\$ | Contains the current relocation factor. |
| SA\$ | Contains the normal starting address used by the apostrophe routine. |
| Bn\$ | Contains the address of breakpoint n. |

These words are stored sequentially as listed; the line feed may be used to step through them.

In the following example, the mask is set to examine instruction code bits (0 through 3) within the limits specified by LO\$ and HI\$.

| | | |
|--------|------------------|----------|
| MSK\$/ | <u>LAW 17777</u> | 740000↓ |
| LO\$/ | <u>CAL+11075</u> | BEGIN-1↓ |
| HI\$/ | <u>END+67</u> | END+1↵ |

After the mask and search limits have been set, the user can execute the search operation for the desired instruction class (all JMP instructions) by typing:

JMP ↵ EQ\$↵

2.8 SYMBOL DEFINITIONS

If the user finds, while debugging, that more symbols would be useful he can easily define them with the following DDT procedure:

- S) DDT assigns the symbol S to the current location.
- k(S) DDT assigns the symbol S to the location specified by the address k.

Example:

13627 (LOCAT)

Space is provided for approximately 25 additional symbols; the exact number depends on the length of the symbols entered. If an attempt is made to enter symbols beyond the allowable limit, DDT types the message OVERFLOW.

Symbols can be redefined as needed.

2.9 PATCH FILE OUTPUT (Paper-Tape Systems Only)

When the process of debugging extends to a number of sessions at the computer, it is convenient to be able to save those changes already checked out for use at later sessions. The commands described below control the output of a patch file onto paper tape.

| | |
|---------|--|
| PFO\$ | DDT outputs all registers within the limits set by LO\$ and HI\$ onto the patch file. PFO\$ may be given as many times as desired. |
| k PFO\$ | Put location k only onto the patch file. |
| SNS\$ | DDT puts all symbols defined during debugging onto the patch file, thus saving them for reference at later sessions. |
| PFE\$ | Close the patch file. |

As many files as desired can be produced by following the sequence of commands, as follows:

```
PFO$
.
.      (as many as desired)
.
.
PFO$
SNS$      (optional)
PFE$
```

2.10 PATCH FILE INPUT (Paper-Tape Systems Only)

Because of the patch file's format, it can be loaded only by DDT. This is done after the user's program has been loaded in the usual manner.

PFI\$ DDT reads in the patch file.

If a read error occurs, DDT stops reading and types the message ERROR followed by a right angle bracket (>).

Data up to the point of error is correctly in memory. At this point, typing PFI\$ (without repositioning the tape) causes patch loading to continue with the patch word after the word causing the error.

Repositioning the tape by moving the tape back one block causes PFIS to attempt to re-read the error word. (See Appendix C for format of the patch file.)

2.11 CO-RESIDENT SUBROUTINES

Identical symbols can be used in two or more separately assembled, or compiled, relocatable program segments that are loaded and run together; consequently the user must be able to specify which set of symbols DDT is to use. DDT initially assumes that the symbol table associated with the first program loaded (i.e., the main program) will be used. The relocation factor used by DDT comes from the symbol table and is, also, initially assumed to be that of the main program. The following DDT command changes both the symbol table search and the relocation factor to the named subroutine.

| | |
|---------|--|
| k;HDR\$ | Sets DDT to refer to that portion of the symbol table associated with the subroutine name k, and to use the relocation factor for that subroutine. (The memory map output by the loader shows all relocation factors.) Symbol tables are not loaded for IOPS, FORTRAN, and user library subroutines. |
| HDR\$ | If no program name is specified, DDT is reset to the initial condition, with main program symbol table and relocation factor assumed. |

2.12 INDIRECT ADDRESS REFERENCES

External global symbols in FORTRAN programs (those used within the program segment, but defined outside of it) are treated differently in the symbol table from those defined within the program segment. These symbols refer to a transfer vector pointing to the named register, not to the named register itself.

Example:

LAB/ 007603

7603 is the actual address of the storage word named LAB. This address must be used when any reference is made to LAB.

In FORTRAN programs, this condition also applies to symbols defined in DIMENSION statements.

2.13 MISCELLANEOUS FEATURES

| | |
|------|---|
| Q\$ | Q\$ represents the content of the currently open storage word. It makes it possible to make small changes without typing the entire contents. In the following example, Q\$ represents JMP LOC+3. |
| LOC/ | <u>JMP LOC+3</u> Q\$+4 |
| LOC/ | <u>JMP LOC+7</u> |

The period, typed alone, represents the address of the currently open, or the most recently opened, word.

```
LOC/   JMP LOC+3   JMP .+7
./      JMP LOC+7
```

& The ampersand causes DDT to bypass the mnemonic instruction lookup. It is necessary if the user has used a recognized mnemonic operator as a symbolic address.

```
JMP/                                     Is invalid, but
&JMP/                                   will open the word named JMP.
LOC/   JMP GO       JMP JMP          The second JMP, in this case, is
                                           interpreted as an address.
```

k# DDT executes the instruction k. The AC and Link are restored to their condition before the breakpoint (if one is in effect). If the instruction is not a JMP, control returns to DDT, and the new AC and Link (if affected) are stored. For example,

```
JMS □ SUBA#
```

causes subroutine SUBA to be executed. SUBA cannot look for subsequent arguments. Skip instructions cause the return pointer to be incremented by one.

↑U If the user makes a typing error, he can cancel the current line by typing CTRL U. DDT types @ as evidence of acceptance. Single character deletion (RUBOUT) is not allowed by DDT. If a RUBOUT is typed, it is treated as a CTRL U.

↑T The user can interrupt his program (or DDT) at any time by typing CTRL T. DDT then types:

```
DDT
0      C(PC)   C(AC)   S(L)
>
```

and waits for a command from the teletype.

APPENDIX A

SUMMARY OF COMMANDS

Linkage Characters

| | |
|---|-------------------------|
| + | Arithmetic plus |
| - | Arithmetic minus |
| ␣ | Field separator (space) |

Breakpoints

| | |
|------|--|
| k␣n" | Insert breakpoint at location k, assign number n (1-4) |
| n" | Remove breakpoint number n |
| " | Remove all existing breakpoints |
| ! | Restart from breakpoint |
| n! | Restart from breakpoint, wait n times before reentering breakpoint |
| ↑T | Restart DDT |

Examination and Modification

| | |
|----|--|
| k/ | Open location k |
| ↵ | (Carriage return) Close the location |
| ↓ | (Line feed) Close the location, open next location |
| ↑ | (up arrow) Close the location, open the preceding location |
| ↑Z | (CTRL Z) Close the location, open addressed location, continue original sequence |
| ↑A | (CTRL A) Close the location, open addressed location, start new sequence |
| ↑X | (CTRL X) Close the location, open the location addressed by 15-bit transfer vector, start new sequence |

Type-out Modes

| | |
|-------|--|
| NUM\$ | Type contents as six-digit octal numbers |
| TV\$ | Type contents as transfer vectors |
| SYM\$ | Type contents as symbolic instructions (assumed by default) |
| : | Retype in alternate mode (NUM\$, SYM\$) |
| = | Retype as transfer vector |
| REL\$ | Type addresses as relative to defined symbols (assumed by default) |

Type-out Modes (continued)

| | |
|-------|-------------------------------------|
| RLC\$ | Type address as relocatable numbers |
| ABS\$ | Type addresses as absolute numbers |

Starts and Restarts

| | |
|----|---|
| ' | Starts user's program at normal starting point |
| k' | Starts user's program at location k |
| ! | Restarts user's program from breakpoint |
| n! | Restarts user's program from breakpoint, waits n times before reentering breakpoint |
| †T | Restart DDT |

Searching Operations

| | |
|---------|---|
| k EQ\$ | Search for words equal to k |
| k UN\$ | Search for words not equal to k |
| k ADR\$ | Search for instructions with effective address equal to k |

Special DDT Locations

| | |
|--------|--|
| AC\$ | Holds AC at a breakpoint |
| LINK\$ | Status of Link at a breakpoint |
| MSK\$ | Contains search mask |
| LO\$ | Lower limit of search |
| HI\$ | Upper limit of search |
| PA\$ | First unused location in patch area |
| AX\$ | Number of auto-index used by breakpoints |
| RF\$ | Current relocation factor |
| SA\$ | Normal starting address |
| Bn\$ | Address of breakpoint n |

Symbol Definition

| | |
|------|---|
| S) | Assign symbol s to the current location |
| k(S) | Assign symbol s to location k |

Patch File Output†

| | |
|---------|-----------------------------------|
| PFO\$ | Patch file output |
| k PFO\$ | Single location patch file output |

†Patch file commands are used for Paper Tape Systems only.

Patch File Output† (continued)

| | |
|-------|-------------------------|
| SNS\$ | Save new symbols |
| PFE\$ | Close patch file output |

Patch File Input†

| | |
|-------|-----------------|
| PFI\$ | Read patch file |
|-------|-----------------|

Coresident Subroutines

| | |
|---------|--|
| k;HDR\$ | Use symbol table and relocation factor of subroutine k |
| HDR\$ | Use symbol table and relocation factor of main program |

Miscellaneous Features

| | |
|-----|--|
| Q\$ | Contents of currently open location |
| . | Address of currently open or most recently opened location |
| & | Bypass mnemonic instruction lookup |
| k# | Execute the instruction k |
| †U | Cancel the line |
| †T | Restart DDT |

†Patch file commands are used for Paper Tape Systems only.

APPENDIX B
MNEMONIC INSTRUCTION TABLE

| <u>Memory Reference</u> | | <u>Operate</u> | |
|-------------------------|--------|----------------|---------|
| CAL | 000000 | NOP | 740000† |
| DAC | 040000 | OPR | 740000 |
| JMS | 100000 | CMA | 740001 |
| DZM | 140000 | CML | 740002 |
| LAC | 200000 | RAL | 740010 |
| XOR | 240000 | RAR | 740020 |
| ADD | 300000 | SMA | 740100 |
| TAD | 340000 | SZA | 740200 |
| XCT | 400000 | SNL | 740400 |
| ISZ | 440000 | SKP | 741000 |
| AND | 500000 | SPA | 741100 |
| SAD | 540000 | SNA | 741200 |
| JMP | 600000 | SZL | 741010 |
| | | RTL | 742010 |
| | | RTR | 742020 |
| | | CLL | 744000 |
| | | STL | 744002 |
| | | RCL | 744010 |
| | | RCR | 744020 |
| | | CLA | 750000 |
| | | CLC | 750001 |
| | | GLK | 750010 |
| | | LAW | 760000 |
| <u>EAE Group</u> | | | |
| EAE | 640000 | | |
| <u>Input/Output</u> | | | |
| IOT | 700000 | | |

†DDT interprets 740000 as NOP.

APPENDIX C
PATCH FILE FORMAT

DDT punches the patch file in four-word blocks, including the two-word block header used by the IOPS system, with blank tape showing between the blocks. Each block carries the address and the contents of one memory word. (See Figure C-1.) The Save New Symbols command (SNS\$) punches the additional symbol table area in the same manner. The PFE\$ command punches an IOPS end-of-file block.

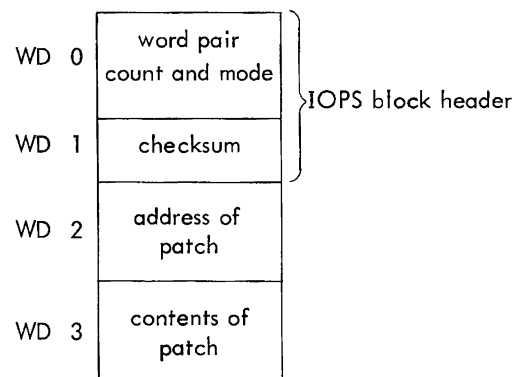


Figure C-1

SECTION 1 INTRODUCTION

1.1 GENERAL DESCRIPTION

SRCCOM (Source Compare Program) is a utility program in the ADVANCED Monitor software system which compares any two symbolic programs (written in IOPS ASCII) and indicates the differences between the compared programs. This utility program is particularly useful for such functions as:

- a. proofing an edited program by comparing it against the original to ensure that the desired changes were made;
- b. keeping track of developmental changes by comparing old and new versions of the same program;
- c. determining if two programs are the same (program identification).

In operation, the user must indicate to the processor, via the input keyboard, which program is to be regarded as the "original" against which the other is to be compared. During execution, SRCCOM outputs statements which indicate the type of modification made to the original (insertion, deletion, and changes) and the actual modification detected.

Three software switch options are provided in SRCCOM; one increases the efficiency of the comparison of programs coded in MACRO-15 (switch M), a second determines the form of SRCCOM output statements (switch A), and a third permits input of non-printing characters to be ignored (switch C).

1.1.1 Software Operating Environment

The SRCCOM utility program operates under control of the PDP-15 ADVANCED Monitor, requiring some form of bulk storage. It uses the Monitor's I/O device handlers to achieve device independence. DECtape to DECtape or Disk to Disk comparisons with output statements sent to a line printer effect an increase in SRCCOM operating speed. This utility program can operate with either PI or API and can use extended memory if it is available. The EAE option is not required.

1.1.2 Minimum Equipment Configuration

A PDP-15/20 basic system is required. SRCCOM will also operate in the basic 15/30 and 15/40 configurations.

1.2 REFERENCE MATERIAL

The sections within this manual which contain information applicable to the use of SRCCOM are referenced directly in the text. The PDP-15/20 ADVANCED Monitor manual (DEC-15-MR2A-D) and the 15/20 Users' Guide (DEC-15-MG2A-D) also contain information useful in understanding and using SRCCOM.

1.3 SPECIAL SYMBOLS

The following symbols are used throughout this program description to denote I/O teletype operations which do not result in a printed character:

| <u>SYMBOL</u> | <u>REPRESENTS</u> |
|---------------|---------------------------------------|
| ↵ | Carriage return - line feed operation |
| → | Tab |
| ␣ | Space |

SECTION 2 OPERATION

2.1 LOADING PROCEDURE

SRCCOM is loaded with the ADVANCED Monitor system Linking Loader. Specific procedures for the use of the Linking Loader are given in detail in another section of this manual; a summary of these procedures is presented in the PDP-15/20 Users' Guide.

2.2 DEVICE ASSIGNMENTS

Appropriate device assignments should be made, using the ADVANCED Monitor ASSIGN command, prior to initiating execution of the SRCCOM program. SRCCOM .DAT slot usage is as follows.

| <u>.DAT Slot</u> | <u>Use</u> |
|------------------|-----------------------|
| -14 | Original file input |
| -15 | New file input |
| -12 | Output Listing |
| -2 | Command string input |
| -3 | Control device output |

The teletype handler (TTA) should be assigned to .DAT slots -2 and -3; this is the control device. Only device handlers capable of handling .FSTAT, .READ, .SEEK, .ENTER, and IOPS ASCII data should be assigned to .DAT slots -14 and -15. Only handlers capable of handling IOPS ASCII data should be assigned to .DAT slot -12.

2.3 OPERATING SEQUENCE

When SRCCOM has been loaded into memory, it types

```
SRCCOM  
>
```

on the teletype and waits for a command string from the user.

The user should first ready the input devices for the programs to be compared, and then type his command string. It is important for the user to follow the proper command string format; otherwise, the compare phase will not proceed as desired, or an error message will be generated. The compare phase of SRCCOM begins on termination of the command string. During the compare phase, the user has control over SRCCOM via the keyboard commands:

| | |
|--------|------------------------------------|
| CTRL C | (Return to Monitor, printed as ↑C) |
| CTRL P | (Restart SRCCOM, printed as ↑P) |

These commands are formed by depressing the CTRL key and striking the appropriate letter key. Control is normally returned to the user at the end of the compare phase. At this point, the user can type a command string to initiate another compare, or return to the monitor by typing CTRL C. CTRL C causes a return to the Monitor at any point during the operation of SRCCOM.

2.4 COMMAND STRING

The SRCCOM command string is typed after the right angle bracket (>) in the following general format.

$$x,y,z \leftarrow \text{newfil} \left\{ \begin{array}{c} : \\ \text{or} \\ \square \end{array} \right\} \text{ext/oldfil} \left\{ \begin{array}{c} : \\ \text{or} \\ \square \end{array} \right\} \text{ext} \rangle$$

where

| | | |
|--------|---|-----------------------------------|
| s,y,z | = | switch options M,A,C |
| newfil | = | file name of new symbolic program |
| oldfil | = | file name of old symbolic program |
| ext | = | file name extension |

The switch options M,A, and C are defined as follows:

M Switch - Used in MACRO-15 comparisons to ignore comments and to perform space tab conversions.

A Switch - Used to specify abbreviated output format (affected lines are not printed).

C Switch - Used to ignore ASCII characters outside of the range 240-337 except for carriage return and horizontal tab characters.

Examples:

- a. To compare MACRO-15 program NEWFIL to MACRO-15 program ORGFIL (with M switch on) type:

M ← NEWFIL:SRC/ORGFIL:SRC >

- b. To compare MACRO-15 program NEWFIL to MACRO-15 program ORGFIL with all switches on, type:

C,M,A ← NEWFIL □ SRC/ORGFIL □ SRC >

c. To compare two programs with no switches on, type:

← NEWFIL □ SRC/ORGFIL □ SRC ▸

2.5 USING NONFILE-ORIENTED INPUT DEVICE

SRCCOM allows for comparison of a segmented paper tape with a DECtape file. When a nonfile-oriented input device (e.g., PR) is assigned to .DAT slots -15 or -14, SRCCOM detects a physical end-of-medium at the end of input. At this point SRCCOM types one of the following messages:

END OF NEW SRC--MORE? (Y OR N?)

or

END OF ORIG SRC--MORE? (Y OR N?)

If any character other than Y or N is typed, SRCCOM responds with a question mark (?) and waits for the user to type a Y or an N. If user desires more input, insert added medium into device (more paper tape into paper tape reader) and type Y.

2.6 FILE-ORIENTED SRCCOM LISTING

When a file-oriented device is assigned to .DAT slot -12, SRCCOM assumes the file name of the original program and supplies a COM extension. See example below:

M NEW EXT/OLD EXT

The file name and extension given to the SRCCOM listing would be:

OLD COM

SECTION 3 OUTPUT FORMATS

3.1 M SWITCH ON

The following paragraphs describe the general SRCCOM output formats for lines inserted, lines deleted, and lines changed with the M switch on. Examples are included in each paragraph for clarification. In each description, TAG represents the last label encountered in the program prior to the noted modification. The letter m represents the number of lines (decimal) from TAG to the last line before the modification. The letter n represents the number of lines (decimal) inserted, deleted, or changed.

3.1.1 Lines Inserted

The following is the general format of unabbreviated output in the case of an insertion in the new program (M switch only).

```
n LINES INSERTED BELOW TAG+m
first line inserted
second line inserted
      ⋮
```

If output is abbreviated (A Switch on also), only the following line is printed:

```
n LINES INSERTED BELOW TAG+m
```

Example

| | | | | |
|----------------------------------|------|-----------|---|---------------------------------|
| 2 LINES INSERTED BELOW TAGONE+20 | | | } | Unabbreviated (A Switch off) |
| DAC | SOME | /inserted | | |
| ISZ | TOOB | /inserted | | |

3.1.2 Lines Deleted

The following is the general format of unabbreviated output in the case of deletion in the new program (only M Switch on).

n LINES DELETED BELOW TAG+m

first deleted line
second deleted line

.
.
.
.

If output is abbreviated (A Switch on also), only the following line is printed:

n LINES DELETED BELOW TAG+m

Example

3 LINES DELETED BELOW TAGONE+20

| | | |
|------------|----------|-----------------------------------|
| DAC* POINT | /Deleted | } Unabbreviated (A Switch off) |
| LAW THREE | /Deleted | |
| DAS SWITCH | /Deleted | |

3.1.3 Lines Changed

The following is the general format of unabbreviated output in the case of lines changed in the new program (M Switch on only).

n LINES CHANGED BELOW TAG+m

line it was changed to in the new symbolic program
line in original symbolic program

line it was changed to in the new symbolic program
next line in the original symbolic program

.
.
.
.

If output is abbreviated (A Switch on also), only the following line is printed:

n LINES CHANGED BELOW TAG+m

Example

2 LINES CHANGED BELOW TAG+20

| | | | |
|---------|---------|-----------|-----------------------------------|
| DAC* | POINTER | /new line | } Unabbreviated (A Switch off) |
| DAC* | POINT | /old line | |
| TOO LAC | FIVE | /new line | |
| TO LAW | THREE | /old line | |

3.2 M SWITCH OFF

The following paragraphs describe the general SRCCOM output formats for lines inserted, lines deleted, and lines changed with the M Switch off. In each description, the letter n represents the number of lines (decimal) affected by the modification, and L represents the line number, also decimal. The first line of program is LINE 0.

3.2.1 Lines Inserted

The following is the general format of unabbreviated output in the case of an insertion in the new program (M Switch off).

n LINES INSERTED BELOW LINE L

line L

first line inserted

second line inserted

.
.
.
.

If output is abbreviated (A Switch on), only the following lines are printed.

n LINES INSERTED BELOW LINE L

line L

Example

12 LINES INSERTED BELOW LINE 58

600 DO 30 I = 1,10 /line 58

} Abbreviated
(A Switch on)

3.2.2 Lines Deleted

The following is the general format of unabbreviated output in the case of a deletion in the new program (both A and M Switches off).

n LINES DELETED BELOW LINE L

line L

first line deleted

second line deleted

.
.
.
.

If output is abbreviated (A Switch on), only the following lines are printed.

n LINES DELETED BELOW LINE L
line L

Example

| | | |
|--------------------------------|----------|-----------------------------------|
| 2 LINES DELETED BELOW LINE 300 | | } Unabbreviated (A Switch off) |
| GO TO 500 | /line L | |
| 100 A = B-C | /deleted | |
| WRITE (1,20) A | /deleted | |

3.2.3 Lines Changed

The following is the general format of unabbreviated output in the case of a change to the new program (both A and M Switches off).

n LINES CHANGED BELOW LINE L
line L
line it was changed to in the new program
line in original symbolic program
line it was changed to in the new program
next line in the original symbolic program
.
.
.
.

If output is abbreviated (A Switch on), only the following lines are printed.

n LINES CHANGED BELOW LINE L
line L

Examples

| | | |
|-------------------------------|-----------|-----------------------------------|
| 2 LINES CHANGED BELOW LINE 38 | | } Unabbreviated (A Switch off) |
| GO TO 500 | /line 38 | |
| 101 A = B-D | /new line | |
| 100 A = B-C | /old line | |
| READ (1,20) | /new line | |
| WRITE (1,20) | /old line | |

SECTION 4 ERROR RECOVERY

4.1 OPERATOR ERRORS

Operator errors that occur while loading SRCCOM are handled in the standard manner by the Linking Loader. Refer to the description of the Linking Loader Utility Program given in this manual.

Operator errors that occur while the user is typing a command string are detected by SRCCOM. SRCCOM outputs a carriage return and line feed, accompanied by one of the following messages, before returning control to the user:

- a. INVALID SWITCH
- b. TOO MANY CHARS IN FILE OR EXT
- c. BOTH FILES NOT SPECIFIED
- d. BAD INPUT DATA
- e. COM USED AS AN INPUT EXT
- f. FILENAMES NOT ON INPUT DEVICES
- g. IMPROPER DATA MODE

To recover from one of the above errors, the user must retype his command string in an acceptable form.

4.2 SOFTWARE ERRORS

If SRCCOM look-ahead capability is exceeded because of gross differences between the two programs being compared, it types

```
LOOK-AHEAD CAPABILITY EXCEEDED AT LINE L )  
(actual contents of line L) )
```

on the teletype, followed by

```
SRCCOM )  
>
```

to indicate that it is ready to accept a new command string.

If SRCCOM detects that there is not enough core available for its compare buffers, it outputs the following message and returns control to the monitor:

NO CORE FOR COMP BUFFS

4.3 DEVICE NOT ENABLED

If devices requested by the user in his command string are not enabled, the monitor outputs an IOPS 4 error message. To recover, enable the appropriate device, and type CTRL R on the teletype. (CTRL R is formed by depressing the CTRL key and striking the letter R and is printed as 1R).

SECTION 1
INTRODUCTION

Systems which have some form of mass storage (Disk or DECtape) are provided with a general-purpose software package which includes a System Generator (SGEN) program. During the initialization of a mass storage system, the user may re-configure the general-purpose software to develop a resident software system unique to his installation. This function is performed using the SGEN program.

SGEN operates in an interactive, conversational mode with the user to generate a unique resident software system; it uses a query/response technique to obtain needed information from the user concerning the system's:

- a. memory size
- b. required device handlers
- c. desired skip chain structure
- d. required .DAT slot assignments.

The SGEN program may also be used to:

- a. re-initialize a system, when necessary
- b. change the resident software system to meet the requirements of a change in the system equipment configuration
- c. change the resident software system to obtain a desired set of system software operating conditions.

The general-purpose software package provided with each mass storage system is in the form of:

- a. a MONITOR System DECtape reel for those systems which have DECtape units;
- b. a group of paper tapes (24) for systems which have Disk but no DECtape units (PDP-9, RB09 only).†
- c. a 7- or 9-channel magnetic tape reel for systems with Disk and magnetic tape, but no DECtape.†

A step-by-step procedure which details the operations required to load the general-purpose system and to develop and checkout a new system tape is provided with the software package. Additional procedures are also given in the respective system User's Guides.

†In each case a utility program is provided on paper tape to transfer the system from the input medium to Disk.

SECTION 2

GENERAL OPERATING PROCEDURES

The normal procedure to use SGEN to generate a resident software system is outlined in the following:

| <u>Step</u> | <u>Procedure</u> |
|-------------|--|
| 1 | The general-purpose software system provided with the equipment can be loaded from either DECtape, MAGtape or Paper Tape, whichever is included. |
| 2 | Prior to calling SGEN, the user must be sure there are available for a tQAREA (save area) at least 40g free blocks for 16K, 100g free blocks for 24K, 150g free blocks for 32K. (For the tQAREA, 40g blocks are required for each 8K of core, plus 10g basic system blocks.) |
| 3 | Use UPDATE, prior to using SGEN, to delete unwanted I/O routines from the library file, .LIBR BIN, to provide additional space. |
| 4 | On completion of Step 3, the user calls the system SGEN program. When SGEN is loaded, it types out a series of queries regarding the following: <ul style="list-style-type: none">a. system core sizeb. optionsc. type of teletype unit usedd. required device handler designationse. skip-chain formationf. default assumptiong. system device designationh. .DAT slot assignments |

The user's responses to these queries enable the SGEN program to generate a new software system configured to best meet the needs of the system as described by the user inputs.

A step-by-step procedure illustrating the use of SGEN is given in Appendix A of this description.

2.1 FORMATION OF SKIP CHAINS

The skip chain structures for standard DECtape, DECtape/Disk, and Disk systems are described in the following paragraphs.

2.1.1 DECtape or DECtape/Disk Systems

An 8K, non-EAE, non-API, KSR33 DECtape system is sent to all DECtape or DECtape/Disk customers. Each customer with a core configuration of greater than 8K, or who has either EAE or API or a KSR35 Teletype, should go through system generation to tailor his installation for optimum use. All customers who, on examining the .SCOM printout, discover devices or options listed that are not present in their system may want to eliminate the irrelevant skips from the chain. Those with non-standard devices (A/D, for example) should expand the chain.

Listed below is the skip chain as it appears in the standard 8K DECtape system:

- a. SPFAL Power Fail
- b. DTDF DECtape Done
- c. DSSF Disk Done
- d. DRSF Drum Done (PDP-9 only)
- e. MTSF Magnetic Tape Done on Error
- f. SPDF 339 Display Flag (PDP-9 only)
- g. LSDF Line Printer Done
- h. PCSF Card Column Ready
- j. PCSD Card Done
- k. CLSF Clock Done
- l. RSF Reader Done
- m. PSF Punch Done
- n. KSF Keyboard Done
- o. TSF Teleprinter Done
- p. DTEF DECtape Error
- q. MPSNE Non-Existent Memory Reference
- r. MPSK Memory Protect Violation
- s. SPE Memory Parity Error
- t. -DRNEF Drum No Error (Negative skip, see description below)

It is important that the above order remain intact even if deletions or additions are to be made. For example, given a system without the Power Fail, Parity or Memory Protect options and having either card reader, line printer or magnetic tape, the skip chain should be generated as follows:

- a. DTDF
- b. DSSF
- c. CLSF
- d. RSF

- e. PSF
- f. KSF
- g. TSF
- h. DTEF

The position of a skip to be added to the chain varies with the nature of the device. For example, high data rate devices are best placed at the top of the chain.

2.1.2 Negative Skips

Skip IOT DRNEF, skip on drum error flag not raised, is a good example of a negative skip, i.e., a skip on a flag not being raised. When specifying such a skip to .SGEN, a minus sign must precede the skip. It should be carefully noted that negative skips should only be included when the device is physically present in the system, because the skip IOT, otherwise, becomes an effective NOP, causing execution of the next instruction (a JMP to the Monitor error routine (IOPS 3)).

2.2 FORMATION OF DEVICE ASSIGNMENT TABLE (.DAT)

All I/O communication in a monitor environment is accomplished according to logical/physical device associations specified in the system software Device Assignment Table (.DAT).

During system generation, SGEN obtains information from the user on a query/response basis to build a .DAT table for the new system. Examples of standard .DAT slot assignments made for several possible system configurations are given in items a through d below.

- a. Listed below are the .DAT slot assignments as they appear in the standard 8K Paper Tape/RB09 Disk system (PDP-9 only):

| <u>.DAT</u> | <u>DEVICE</u> | <u>USE</u> |
|-------------|---------------|----------------------------|
| -15 | DKA6 | OUTPUT |
| -14 | DKA4 | INPUT |
| -13 | DKB5 | OUTPUT |
| -12 | TTA0 | LISTING |
| -11 | DKB4 | INPUT |
| -10 | TTA0 | INPUT |
| -7 | DKC0 | SYSTEM DEVICE FOR .SYSLD |
| -6 | NONE | OUTPUT |
| -5 | NONE | EXTERNAL LIBRARY FOR .LOAD |
| -4 | DKC5 | SYSTEM INPUT |
| -3 | TTA0 | TELEPRINTER OUTPUT |
| -2 | TTA0 | KEYBOARD INPUT |
| -1 | DKC0 | SYSTEM DEVICE FOR .LOAD |
| 1 | DKA4 | USER |
| 2 | DKA5 | USER |
| 3 | DKA6 | USER |

| <u>.DAT</u> | <u>DEVICE</u> | <u>USE</u> |
|-------------|---------------|------------|
| 4 | TTA0 | USER |
| 5 | PRA0 | USER |
| 6 | PPB0 | USER |
| 7 | DKA1 | USER |
| 10 | DKA2 | USER |

- b. Listed below are the .DAT slot assignments as they appear in the standard 8K DECtape system.

| <u>.DAT</u> | <u>DEVICE</u> | <u>USE</u> |
|-------------|---------------|----------------------------|
| -15 | DTA2 | OUTPUT |
| -14 | DTA1 | INPUT |
| -13 | DTB2 | OUTPUT |
| -12 | TTA0 | LISTING |
| -11 | DTB1 | INPUT |
| -10 | TTA0 | INPUT |
| -7 | DTC0 | SYSTEM DEVICE FOR .SYSLD |
| -6 | NONE | OUTPUT |
| -5 | NONE | EXTERNAL LIBRARY FOR .LOAD |
| -4 | DTC2 | SYSTEM INPUT |
| -3 | TTA0 | TELEPRINTER OUTPUT |
| -2 | TTA0 | KEYBOARD INPUT |
| -1 | DTC0 | SYSTEM DEVICE FOR .LOAD |
| 1 | DTA0 | USER |
| 2 | DTA1 | USER |
| 3 | DTA2 | USER |
| 4 | TTA0 | USER |
| 5 | PRA0 | USER |
| 6 | PPB0 | USER |
| 7 | DTA1 | USER |
| 10 | DTA2 | USER |

- c. The following examples are variations on .DAT slot assignments† as a function of either core size or different peripherals.

(1) Given an 8K system with line printer and card reader: LPA should be assigned to .DAT slot -12 and one of the positive slots, for example, 3, 7, or 10. CDB should be assigned to one of the positive slots.

(2) Given a 16K (or greater) Disk/DECtape system, a suggested list of assignments is as follows:

| | | | | | |
|-----|------|----|------|----|------|
| -15 | DKA6 | -6 | NONE | 2 | DKA5 |
| -14 | DKA4 | -5 | NONE | 3 | DKA6 |
| -13 | DKA5 | -4 | DKA5 | 4 | TTA |
| -12 | TTA | -3 | TTA | 5 | PRA |
| -11 | DKA4 | -2 | TTA | 6 | PPA |
| -10 | PRA | -1 | DKA0 | 7 | DTA1 |
| -7 | DKC0 | 1 | DKA4 | 10 | DTA2 |

†Installations with 16K or more core should assign the A versions of handlers to all .DAT slots.

(3) Given a 16K (or greater) DECtape system with magnetic tape, a suggested list of assignments is as follows:

| | | | | | |
|-----|------|----|------|----|------|
| -15 | DTA2 | -6 | NONE | 2 | DTA1 |
| -14 | DTA1 | -5 | NONE | 3 | DTA2 |
| -13 | DTA2 | -4 | DTA2 | 4 | TTA |
| -12 | TTA | -3 | TTA | 5 | PRA |
| -11 | DTA1 | -2 | TTA | 6 | PPA |
| -10 | PRA | -1 | DTA0 | 7 | MTF1 |
| -7 | DTC0 | 1 | DTA0 | 10 | MTF2 |

d. The following example is a variation of .DAT slot assignments as a function of both increased core size and additional peripherals.

Given a 16K Disk system with line printer and card reader, a suggested list of assignments follows:

| | | | | | |
|-----|------|----|------|----|------|
| -15 | DKA6 | -6 | NONE | 2 | DTA1 |
| -14 | DKA4 | -5 | NONE | 3 | DTA2 |
| -13 | DKA5 | -4 | DKA5 | 4 | LPA |
| -12 | LPA | -3 | TTA | 5 | PRA |
| -11 | DKA4 | -2 | TTA | 6 | PPA |
| -10 | PRA | -1 | DKA0 | 7 | CDB |
| -7 | DKC0 | 1 | DKA4 | 10 | DKA5 |

APPENDIX A
SYSTEM GENERATION, STEP-BY-STEP PROCEDURE

The following table illustrates the step-by-step query/response procedure developed while using SGEN with either DECtape or Disk.

Table A-1
Query/Response Procedure

| Step | Procedure | Printout |
|------|---|--|
| 1 | Type-- SGEN ↵ | \$SGEN |
| 2 | Program Title and explanatory message | SYSTEM GENERATOR Vnn THIS PROGRAM WILL GENERATE A NEW SYSTEM TAPE ON THE DEVICE SPECIFIED IN .DAT SLOT -15. IT WILL DETERMINE THE CHARACTERISTICS OF THIS SYSTEM TAPE BY ASKING YOU A SERIES OF QUESTIONS. IF IT CANNOT UNDERSTAND THE ANSWER YOU GIVE, IT WILL REPEAT THE QUESTION. HERE GOES! |
| 3 | Query | HOW MUCH CORE IS AVAILABLE? TYPE 8,16,24,OR 32. |
| 4 | Type system core size | > <input type="checkbox"/> ↵ |
| 5 | Query | IS AN API AVAILABLE? TYPE Y OR N. |
| 6 | Type Y (yes) or N (no) | > <input type="checkbox"/> ↵ |
| 7 | Query | IS AN EAE AVAILABLE? TYPE Y OR N. |
| 8 | Type Y or N | > <input type="checkbox"/> ↵ |
| 9 | Query | IS TELETYPE A MODEL 33? TYPE Y OR N. |
| 10 | Type Y or N | > <input type="checkbox"/> ↵ |
| 11 | Directive | INDICATE THE PRESENCE OR ABSENCE OF THE FOLLOWING HANDLERS BY TYPING Y OR N: |
| 12 | Query string; respond to each with either Y or N. | PRA? > <input type="checkbox"/> ↵ PRB? > <input type="checkbox"/> ↵ PPA? > <input type="checkbox"/> ↵ PPB? > <input type="checkbox"/> ↵ PPC? > <input type="checkbox"/> ↵ DTA? > <input type="checkbox"/> ↵ DTB? > <input type="checkbox"/> ↵ DTC? > <input type="checkbox"/> ↵ DTD? > <input type="checkbox"/> ↵ DKA? > <input type="checkbox"/> ↵ DKB? > <input type="checkbox"/> ↵ DKC? > <input type="checkbox"/> ↵ |

Table A-1 (Cont)
Query/Response Procedure

| Step | Procedure | Printout |
|--|---|---|
| 13 | Query | DKD? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> MTF? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> LPA? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> CDB? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DYA? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DRA? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DRB? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DRC? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> DRD? > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> } PDP-9 only ARE ANY OTHER DEVICE HANDLERS PRESENT? TYPE Y OR N. |
| NOTE If you HAVE device handlers NOT listed in the preceding query string (STEP 12) DO STEPS 14 through 22. If you DO NOT have additional device handlers, enter N (no) in answer to the STEP 13 query and GO TO STEP 23. | | |
| 14 | Type Y | > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |
| 15 | Query | HOW MANY? TYPE OCTAL NUMBER. |
| 16 | Type number of other handlers in octal form. | > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |
| 17 | Directive | TYPE THREE CHARACTER HANDLER NAME FOR NO. 01. |
| 18 | Type 3-character mnemonic for each handler (e.g., MTF) | > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |
| 19 | Query | HOW MANY SKIP IOTS SHOULD BE IN SKIP CHAIN FOR THIS DEVICE HANDLER? TYPE OCTAL NUMBER. |
| 20 | Type answer in octal notation | > <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |
| 21 | Directive, given for each handler | TYPE UP TO FIVE CHARACTER MNEMONIC FOR SKIP IOT NO. A COMMA, AND OCTAL SKIP IOT. |
| 22 | Type mnemonic, a comma and octal Skip IOT. | > <input type="text"/> STEPS 17 THROUGH 22 ARE REPEATED FOR EACH ADDITIONAL HANDLER AND HANDLER SKIP IOTS |
| 23 | Directive | THE FOLLOWING SKIP IOTS ARE TO BE INCLUDED IN THE SYSTEM SKIP CHAIN: |
| 24 | List of Skip Chain Input/Output Transfer Command Mnemonics. | CLSF MPSNE MPSK SPE SPFAL |

Table A-1 (Cont)
Query/Response Procedure

| Step | Procedure | Printout |
|------|---|--|
| 25 | Directive | KSF TSF RSF PSF DTFD DTEF MTSF LSDF RCSF RCSD TYPE THEM IN SKIP CHAIN ORDER, ONE PER NUMBER. TYPE DONE IF ALL DESIRED SKIPS HAVE BEEN ENTERED. (PRECEDE SKIP BY A MINUS IF REVERSE SKIP IOT.) NOTE: USE ↑P TO RETURN TO THIS POINT. |
| 26 | List of queries regarding formation of skip chain. Type IOT mnemonics using Section 1 as a guide. | NO. 01? > <input type="checkbox"/> ↵ NO. 02? > <input type="checkbox"/> ↵ NO. 02? > <input type="checkbox"/> ↵ NO. 03? > <input type="checkbox"/> ↵ NO. 03? > <input type="checkbox"/> ↵ NO. 03? > <input type="checkbox"/> ↵ NO. 04? > <input type="checkbox"/> ↵ NO. 05? > <input type="checkbox"/> ↵ NO. 06? > <input type="checkbox"/> ↵ NO. 07? > <input type="checkbox"/> ↵ NO. 10? > <input type="checkbox"/> ↵ NO. 11? > <input type="checkbox"/> ↵ NO. 12? > <input type="checkbox"/> ↵ NO. 13? > <input type="checkbox"/> ↵ NO. 14? > <input type="checkbox"/> ↵ NO. 15? > <input type="checkbox"/> ↵ NO. 16? > <input type="checkbox"/> ↵ NO. 17? > <input type="checkbox"/> ↵ |
| 27 | Query | SHOULD DEFAULT ASSUMPTION BE 7 CHANNEL MAG-TAPE? TYPE Y OR N. |
| 28 | Type Y or N | > <input type="checkbox"/> ↵ |
| 29 | Query | WHAT IS THE SYSTEM DEVICE? TYPE DT OR DK. |
| 30 | Type DT (DEctape) or DK (Disk). | > <input type="checkbox"/> ↵ |
| 31 | Directive | TYPE THE DEVICE HANDLER NAME (NON OR NONE) AND UNIT NO. FOR THE FOLLOWING .DAT SLOTS: NOTE: USE ↑P TO RETURN TO THIS POINT |
| 32 | List of queries regarding .DAT Slot Assignments. Type handler mnemonics. | -15? > <input type="checkbox"/> ↵ -14? > <input type="checkbox"/> ↵ -13? > <input type="checkbox"/> ↵ -12? > <input type="checkbox"/> ↵ -11? > <input type="checkbox"/> ↵ |

Table A-1 (Cont)
Query/Response Procedure

| Step | Procedure | Printout |
|------|---|--|
| 33 | Explanatory | -10? > <input type="checkbox"/>) -6? > <input type="checkbox"/>) -5? > <input type="checkbox"/>) -4? > <input type="checkbox"/>) -1? > <input type="checkbox"/>) 1? > <input type="checkbox"/>) 2? > <input type="checkbox"/>) 3? > <input type="checkbox"/>) 4? > <input type="checkbox"/>) 5? > <input type="checkbox"/>) 6? > <input type="checkbox"/>) 7? > <input type="checkbox"/>) 10? > <input type="checkbox"/>) THANKS, THIS WILL TAKE A FEW MINUTES. (1P ILLEGAL NOW) |
| | | NOTE APPROXIMATE TIME REQUIRED: a. DECTape to DECTape ... 15 to 20 min. b. DECTape to Disk 10 min. c. Disk to Disk 5 min. or less. |
| 34 | System generation complete, indicates on which Tape or Disk New System is stored. | ALL DONE - NEW SYSTEM ON |
| | | NOTE Printouts will be DT1 for DECTape only, DK0 for DECTape to Disk, DK1 for Disk only system. Each printout requires a different procedure. |
| 35 | a. If DT1 go to STEP 35 b. If DK0 go to STEP 37 c. If DK1 go to STEP 38 Do the following to DECTape Unit 8: a. Remove system DECTape. b. Mount new scratch DECTape. c. Change unit number Dial from 8 to 1. | |

Table A-1 (Cont)
Query/Response Procedure

| Step | Procedure | Printout |
|--|---|--------------------|
| 36 | <p>d. Set unit WRITE ENABLE/LOCK switch to WRITE ENABLE.</p> <p>Do the following to original DECTape Unit 1:</p> <p>a. Change unit number Dial from 1 to 8.</p> <p>b. Set unit WRITE ENABLE/LOCK switch to LOCK.</p> | |
| <p>NOTE</p> <p>On completing Step 36, go to Step 44.</p> | | |
| 37 | <p>Do the following to DECTape Unit 8:</p> <p>a. Remove system DECTape.</p> <p>b. Mount new scratch DECTape.</p> <p>c. Set unit WRITE ENABLE/LOCK switch to WRITE ENABLE.</p> <p>d. Change unit number dial from 8 to 2.</p> <p>e. Set Disk control panel WRITE PROTECT switch and toggle switches 10, 20, and 30 to ON (up).</p> | |
| <p>NOTE</p> <p>On completing Step 37, go to Step 44.</p> | | |
| 38 | Set Disk control panel WRITE PROTECT switch to OFF (down) position, and perform Steps 39 through 42. | |
| 39 | Enter .DAT SLOT assignment A DKA0 2/DKA1 3 . | \$ A DKA0 2/DKA1 3 |
| 40 | Enter PIP 2 . | \$ PIP |
| 41 | Response | PIP Vnn > |

Table A-1 (Cont)
Query/Response Procedure

| Step | Procedure | Printout |
|------|--|---|
| 42 | Enter C <input type="checkbox"/> DK0 <input type="checkbox"/> (H) <input type="checkbox"/> <input type="checkbox"/> ←DK1 ↵ to initiate transfer. | > C DK0 (H) <input type="checkbox"/> ←DK1 |
| 43 | Response (transfer complete) | |
| | <p>NOTE</p> <p>Protect contents of Disk Unit 0 by setting Disk Control Panel WRITE PROTECT switch and switches 10, 20, 30 to the ON (up) position (RB09 only).</p> | |
| 44 | Load paper tape BOOTSTRAP (Disk or DECTape, whichever is applicable) in paper tape reader. | |

APPENDIX B PIP ERROR MESSAGES

| <u>Error Message</u> | <u>Action</u> |
|--------------------------------|--|
| COMM STRING TOO LONG | Retype command string. |
| ILL FUNCTION | Retype from function character on. |
| ILL DEV OR UNIT | Retype from device name on. |
| ILL DEV TERMINATOR | |
| DEV ILL FOR OPTION OR FUNCTION | |
| DEV NOT IN + .DAT TABLE | Type tC to restore. |
| TOO MANY FILES OR BLKS | Retype command string. |
| FILE NAME TOO LONG | Retype from File Name on. |
| FILE NOT ON DEV | |
| TOO MANY SRC FILES | |
| TOO MANY DEST FILES | Check number of files actually transferred and type another command string to transfer remainder. |
| DATA MODE NEEDED | Type data mode in parentheses followed by carriage return. |
| SWITCH ILL FOR DEV | Retype from switch on. |
| ILL SWITCH | |
| SWITCH CONFLICT | |
| SWITCH ILL FOR FUNCTION | |
| ILL TERMINATOR | Retype from terminator on. |
| INPUT PARITY ERR | If binary, check data. If ASCII, retype command string using G switch. |
| INPUT CKSUM ERR | Check data. |
| INPUT LINE TOO LONG | |
| ILL BLK# | Retype from block # on. |
| READ-COMP ERR ON BLK:n | When operation complete, try B function on error block. |
| S FUNC NOT DONE | Execute S operation; then retype T command. |
| STRING 1 TO 16 ACCEPTED | Perform segmentation; then further segment last destination file. |
| TOO FEW DEST FILES | Retype command string with correct # of destination files. (1 more than # of segmentation points.) |

READER'S COMMENTS

PDP-15 UTILITY MANUAL
DEC-15-YWZA-D

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback – your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

Did you find errors in this manual? _____

How can this manual be improved? _____

DEC also strives to keep its customers informed of current DEC software and publications. Thus, the following periodically distributed publications are available upon request. Please check the appropriate boxes for a current issue of the publication(s) desired.

- ☐ Software Manual Update, a quarterly collection of revisions to current software manuals.
- ☐ User's Bookshelf, a bibliography of current software manuals.
- ☐ Program Library Price List, a list of currently available software programs and manuals.

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 Family
Digital Software News for the PDP-9/15 Family
PDP-6/PDP-10 Software Bulletin

These newsletters contain information applicable to software available from Digital's Program Library.

Please complete the card below to place your name on the newsletter mailing list.

Questions or problems concerning DEC Software should be reported to the Software Specialist at your nearest DEC regional or district sales office. In cases where no Software Specialist is available, please send a Software Trouble Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms, which are available without charge from the Program Library, should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

New and revised software and manuals, Software Trouble Report forms, and cumulative Software Manual Updates are available from the Program Library. When ordering, include the document number and a brief description of the program or manual requested. Revisions of programs and documents will be announced in the newsletters and a price list will be included twice yearly. Direct all inquiries and requests to:

Program Library
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

Digital Equipment Computer Users Society (DECUS) maintains a user Library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

Send Digital's software newsletters to:

Name _____

Company Name _____

Address _____

My computer is a

PDP-8/I ☐

LINC-8 ☐

PDP-9 ☐

PDP-10 ☐

PDP-8/L ☐

PDP-12 ☐

PDP-15 ☐

OTHER ☐

(zip code)

Please specify _____

My system serial number is _____ (if known)