SYMBOLIC EDITOR PROGRAMMING MANUAL

PDP-8

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

DEC-08-ESAB-D

PDP-8 SYMBOLIC EDITOR PROGRAMMING MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Revised November 1967
Reprinted February 1968

PREFACE

The PDP-8 comes to the user complete with an extensive selection of system programs and routines making the full data processing capability of the new computer immediately available to each user, eliminating many commonly experienced initial programming delays.

The programs described in these abstracts come from two sources, past programming effort on the PDP-5 computer, and present and continuing programming effort on the PDP-8. Thus the PDP-8 programming system takes advantage of the many man-years of program development and field testing by PDP-5 users.

Although in many cases PDP-8 programs originated as PDP-5 programs, all utility and functional program documentation is issued in a new, recursive format introduced with the PDP-8.

Programs written by users of either the PDP-5 or the PDP-8 and submitted to the users' library (DECUS - Digital Equipment Corporation Users' Society) are immediately available to PDP-8 users.

Consequently, users of either computer can take immediate advantage of the continuing program developments for the other.

CONTENTS

CONTENTS (cont.)

INTRODUCTION

The PDP-8 Symbolic Editor allows the user to prepare and edit symbolic tapes on-line in ASCII code with the Teletype and/or high-speed reader/punch. The tedious task of correcting symbolic program tapes using the Teletype off-line is thereby avoided. Proper use of the PDP-8 Symbolic Editor can substantially ease the labor and reduce the number of passes necessary to correct symbolic program tapes.

The Editor reads a page, or section, of symbolic tape into a buffer in core storage, where it is available for examination and correction. The page buffer occupies all of core not taken up by the Editor itself and has a capacity of approximately 5000_{10} characters. When the Editor has finished reading a page into the buffer, a bell rings to signal the user that he may begin editing. The user may then call for a listing of individual (numbered) lines, in any order, and insert desired changes and corrections. In addition, text may be added to the buffer, or inserted between specified lines. Groups of lines or individual lines may be moved or deleted by a single command, or the entire page may be erased if desired. Upon command, the Editor will then either list or punch out the corrected lines or page on paper tape. The Editor can also be used to generate a new symbolic tape by typing new text directly on the console keyboard.

SECTION 1
OPERATING FEATURES

By convention, paper tape information is organized into lines, in variable sized blocks, called pages. Pages are separated on paper tape by ASCII form feed codes, and lines are separated by carriage return/line feed pairs. A page of text may contain about 60 lines of heavily commented text, or approximately 5000_{10} characters; it can hold about 340 lines without comments or formatting. Each line of text includes the terminating carriage return/line feed combination. All lines in the text buffer are implicitly numbered in decimal notation starting with 1. This implicit enumeration is continually updated by the Editor to take account of line insertions, moves and deletions. For editing and listing purposes, each line is referred to by its current implicit decimal number on the page.

Since the form feed code is not stored in core memory, there are no page divisions in the text buffer; the entire contents of the buffer are treated as a single page. The user may output the contents of the text buffer as several pages, however.

1.1 MODES OF OPERATION

To distinguish between editing commands and the actual text to be entered in the buffer, the Editor operates either in command mode or text mode. In command mode all input typed on the Teletype will be interpreted as commands to the Editor to perform some operation, or to allow the operator to perform some operation, on the text stored in the buffer. In text mode, all typed input is interpreted as text to replace, be inserted into, or to be appended to the contents of the text buffer.

1.1.1 Transition Between Modes

After being loaded into core memory the Editor is in command mode; that is, the program is waiting for a command. The user types the desired command code and terminates it by striking the carriage return (RETURN) key. This nonprinting character (hereafter represented by ⤶) tells the Editor to carry out the command. The Editor then enters text mode and responds with a line feed character (represented by ↓) as soon as it has processed the command and begun the operation.

With the Editor in text mode, the user types the desired corrections or insertions to his text. To terminate the text he enters a form feed (CTRL/FORM combination) to tell the Editor to return to command mode. The Editor answers by ringing a bell to indicate the transition back to command mode.

1.2 COMMAND STRUCTURE

A command directs the Editor to perform a desired operation. Each command consists of a single letter, preceded by zero, one, two or three arguments. The command letter tells the Editor what to do; the arguments usually specify which numbered line or lines of text are affected. (Some arguments consist of special characters, see section 1.3). Commands to the Editor must take one of the following forms, where E represents any command letter and the symbol ⤵ represents the nonprinting character for carriage return (RETURN key).

Type of Command	Command Format	Meaning
No Argument	E ⤵	Perform operation E
One Argument	nE ⤵	Perform operation E on the referenced line.
Two Arguments	m,nE ⤵	Perform operation E on lines m through n, inclusive.
Three Arguments	m,n$jE ⤵	Used by MOVE command only.

The arguments m and n, which refer to numbered lines in memory, must be positive and n must be greater than m.

Two arguments must be separated by a comma, but no comma is allowed between the argument(s) and the command. Note also that in order to be executed a command must be followed by a carriage return (⤵).

1.3 SPECIAL CHARACTERS AND FUNCTIONS

A number of keys have special operating functions. These keys and their associated functions are listed below. The nonprinting characters are noted; the symbols for these are shown in parentheses. All other echo the character in parentheses.

1.3.1 Carriage Return (⤵ nonprinting)

In both command and text modes, striking the carriage return key (RETURN) signals the Editor to process the information just typed. In command mode, it allows the Editor to execute the command just typed. A command will not be executed until it is terminated by striking the RETURN key (with the exception of = which needs no CR). In text mode, it causes the line of text which it follows to be entered in the text buffer. A typed line is not actually part of the buffer until terminated by a carriage return.

1.3.2 Back Arrow (←)

The back arrow (←) is used for error recoveries in both command and text modes. When used in text mode, ← cancels everything to the left of itself back to the beginning of the line. The user then continues typing on the same line. When used in command mode, ← cancels the entire command and the Editor issues a "?" and a carriage return/line feed (CR/LF). Back arrow cannot cancel past a CR/LF in either command or text mode.

```
A ←  ? (CR/LF)
THIS ←  "HERE IS A TEXT MODE EXAMPLE" (CR/LF)
     only the part in quotes is entered in the buffer
```

1.3.3 Rubout (\ )

Rubout (\ ) is also used in error recovery in both command and text modes with one exception. When executing a READ command from either the paper tape or Teletype reader, rubouts are ignored completely and do not go into the buffer.

It is necessary for the READ command to disable the rubout function since all tab characters on paper tape are, for timing purposes, followed by rubouts which would destroy the tabs. Rubouts are not stored in the text buffer but are inserted by the Editor following all tab characters on the output tape.

At any other time in text mode (specifically if text mode was entered via the APPEND, CHANGE, INSERT or SEARCH command) typing rubout echoes a back slash (\ ) and deletes the last typed character. Repeated rubouts delete from right to left up to, but not including the CR/LF combination, separating the current line from the previous one.

Example:

```
THE QUUICK\\\\ICK BROWN FOX (CR/LF) will be entered in the buffer as
THE QUICK BROWN FOX
```

When used in command mode, rubout is equivalent to back arrow and cancels the entire command. The Editor then issues a "?" and a CR/LF combination.

1.3.4 Form Feed (CTRL/FORM nonprinting)

Form feed signals the Editor to return to command mode. A form feed character is generated by depressing and holding the CTRL key and hitting the FORM key. This combination is typed while in text mode to indicate that the desired text has been entered and the Editor should now return to command mode. The Editor rings a bell in response to a CTRL/FORM to indicate the transition back to command mode. If the Editor is already in command mode when CTRL/FORM is typed, no bell will sound. CTRL/BELL is equivalent to CTRL/FORM except in the case of a SEARCH command (see editing commands).

1.3.5 Period (.)

The Editor keeps track of the implicit decimal number of the line on which it is currently operating. At any given time the symbol period (.) stands for this number and may be used as an argument to a command. Example: .L ↵ means list the current line. .−1,.+1L ↵ means list the line preceeding the current line, the current line, and the line following it.

After a READ or APPEND command, the current line counter (.) is the number of the last line in the buffer. After an INSERT or CHANGE command, (.) is equal to the number of the last line entered. After a LIST command, (.) is the number of the last listed line. After a DELETE command, (.) is the number of the line immediately before the deletion. After a KILL command, (.) is 0. After a GET command, (.) is the number of the line typed by the GET. After a MOVE or SEARCH command the current line counter (.) is not updated and remains at whatever it was before the command.

1.3.6 Slash (/)

The symbol slash (/) has a value equal to the decimal number of the last line in the buffer. It may also be used as an argument to a command. Example: 10,/L↵ means list from line 10 to the end of the buffer.

1.3.7 Line Feed (↓ nonprinting)

Commands are terminated by a carriage return/line feed (CR/LF) combination and the lines on each page of text are separated by a CR and LF. The user need only strike the RETURN key, however, to terminate a command or input line, since the Editor automatically generates a line feed to follow each carriage return.

On input from paper tape, line feed characters are completely ignored. On output the Editor automatically punches a line feed following each carriage return.

Typing a line feed while in command mode is equivalent to typing ".+1L↵" and will cause the Editor to type out the line following the current one and increment the value of (.) by one.

1.3.8 ALT Mode (ALT nonprinting)

Hitting the ALT mode key (ALT) while in command mode will also cause the line following the current line to be typed out and (.) to be incremented by one. If the current line is also the last line in the buffer, typing either ALT mode or Line Feed will be answered by a "?" from the Editor indicating there is no "next" line. Some Teletypes have an Escape (ESC) key in place of the ALT mode. The function is identical for Escape or ALT mode.

1.3.9 Left Angle Bracket (<)

Typing Left Angle Bracket (<) while in command mode is equivalent to typing ".−1L↵" and will cause the Editor to echo < and then type out the line preceding the current line. The value of (.) is decreased by one so that it still refers to the last line typed out.

1.3.10 Equal Sign (=)

Equal Sign is used in conjunction with the pointers period (.) and slash (/). When typed in command mode it causes the Editor to print out the decimal value of the argument preceding it, followed by a CR/LF. In this way the number of the current line may be found (.=XXX) or the total number of lines in the buffer (/=XXX) or the number of some particular line (/−8=XXX) may be determined without counting from the beginning.

1.3.11 Colon (:)

Colon is a lower case character with exactly the same function as (=).

1.3.12 Blank Tape and Leader/Trailer

Both Blank Tape and Leader/Trailer (code 200) are completely ignored on an input tape as are line feed characters and rubouts. Line feeds and rubouts are automatically replaced wherever necessary on output, blank tape and leader/trailer are not.

1.3.13 Tabulation (⇥ nonprinting)

The Editor is written in such a way as to simulate "tab stops" at ten space intervals across the carriage. When the user holds the CTRL key and strikes the TAB key the Editor produces a tabulation. A tabulation consists of from one to ten spaces, depending on the number needed to bring the carriage to the next tab stop. Thus the user may use the Editor to produce neat columns on the hard copy.

This tab function is attached to two switch register options (one for input, one for output) to allow the user to produce and control tabulations in the text buffer during input and output operations (see Switch Options). On input (under a READ command) the Editor can replace a group of two or more spaces with a tabulation if the user chooses to set bit 0. On output it will produce either a tab character followed by a rubout (for timing purposes) or enough spaces to reach a "tab stop", depending on the setting of bit 1. The Editor cannot output tab characters unless tabulations have been entered in the buffer either from the keyboard or through setting bit 0 on input.

NOTE: Location 0002 contains the negative (2's complement) of the number of spaces used to simulate "tab stops". To change the tabulation simply change the constant in location 0002 after loading the Editor.

1.4 SWITCH REGISTER OPTIONS

The Editor uses five switches in conjunction with the actual input and output commands to control the reading and punching of paper tape. It is sometimes desirable to be able to interrupt a command before it finishes. For example, if the user mistakenly gave a LIST command in place of a PUNCH command, he does not wish to wait for the Teletype to list a large section of text. Bit 2 allows him to interrupt any output command and return immediately to command mode.

Bit	Position	Action and Explanation
0	0	Read the input tape exactly as it is.
	1	Read the input tape taking note of spaces. Each time two or more successive spaces are found, substitute in the buffer a tabulation for that whole group of spaces. This option affects only the READ command.
1	0	On punching (or listing) text from the buffer, tabulations are to be interpreted as an appropriate number of spaces.
	1	Tabulations are interpreted as a tab character followed by a rubout (211/377).
2	0	Normal operation. All output commands completed as specified.
	1	Suppress output. If at any time during execution of an output command this switch is set to 1, the output will cease and the Editor will return to command mode in a few seconds. The switch should then be set to 0 to continue using the Editor.
10	0	Low speed output. All punching will be done via the Teletype punch.
	1	High speed output. All punching will be done via the high speed punch.
11	0	Low speed input. The READ command expects the source tape to be in the Teletype reader. DO NOT use the APPEND command to read tapes.
	1	High speed input. The source tape will be read from the photoelectric reader.

SECTION 2
COMMAND REPERTOIRE

Commands to the Editor are grouped under three general headings:

Input Commands
Output Commands
Editing Commands

Explanation of the three types of commands is given in the following sections. Each command description will state if the Editor returns to the command mode after completing the operation specified by the command.

The Editor will print an error message consisting of a question mark whenever the user has requested nonexistent information or used an inconsistent or incorrect format in typing a command. For example, if a command requires two arguments, and only one (or none) is provided, the Editor will print "?", issue a carriage return and line feed, and ignore the command as typed. Similarly, if a nonexistent command character is typed, the error message "?" will be typed, a carriage return and line feed issued and the command will be ignored. However, if an argument is provided for a command that does not require one, the argument will be ignored and the normal function of the command performed.

Examples:

Message	Explanation
L↵ ?↵	The buffer is empty. The user is asking for nonexistent information.
7,5L↵ ?↵	The arguments are in the wrong order. The Editor cannot list backwards.
17$10M↵ ?↵	This command requires two arguments before the "$", only one was provided.
H↵ ?↵	Nonexistent command letter.

2.1 INPUT COMMANDS

Command	Action and Explanation
R↵	READ a page of text from the paper tape reader. Depending on the position of switch register bit 11 (SR 11) reading will be done from either the photoelectric or the Teletype reader. The Editor will read information from the input tape until a form feed character (CTRL/FORM key combination) is detected. All incoming text except the form feed is appended to the contents of the text buffer. Information already in the buffer remains there.

	In the case of input via the photoelectric reader, the end of the tape will be interpreted as a form feed and the Editor returned to command mode, if an actual form feed character does not appear on the tape. In the case of input via the Teletype reader, a form feed must be entered via the keyboard to return the Editor to command mode if an actual form feed character does not appear on the tape. If this is not done, the READ command is still in effect and all subsequent commands will be interpreted erroneously as text and appended to what was just read from tape. Any rubout encountered during a READ command will be ignored. (See RUBOUT)
A↲	APPEND the incoming text from the teleprinter keyboard to the information already in the buffer (the buffer may be empty initially). The Editor will enter the text mode upon receiving this command and the user may then type in any number of lines of text. The new text will be appended to the information already in the buffer, if any, until the form feed (CTRL/FORM) key combination is struck. By giving the APPEND command with an empty buffer, a symbolic program tape may effectively be generated on-line by entering the program via the keyboard. Any rubout encountered during execution of an APPEND command will actually delete the last typed character. Repeated rubouts will detele from right to left up to, but not beyond the beginning of the current line. The APPEND command must not be used to read paper tapes from the Teletype reader since every rubout on the tape will detele a character.

NOTE: In both of these commands, the Editor returns to the command mode only after the form feed character.

2.2 OUTPUT COMMANDS

Output commands are subdivided into list and punch commands. List commands will cause the printout on the Teletype of all or any part of the contents of the text buffer to permit examination of the text. Punch commands provide for the output of leader trailer, form feeds, corrected text or for the duplication of pages of an input tape. List or punch commands do not affect the contents of the buffer.

2.2.1 List Commands

The following commands cause part or all of the contents of the text buffer to be listed on the Teletype.

Command	Action and Explanation
L↲	LIST the entire page. This causes the Editor to list the entire contents of the text buffer.

Command	Action and Explanation
nL↵	LIST line n. This line will be typed out, followed by a carriage return and a line feed.
m,nL↵	LIST lines m through n, inclusive. Lines m through n will be printed on the Teletype.

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

2.2.2 Punch Commands

The following commands control the punching onto paper tape of leader/trailer, text and form feeds.

Note that the PUNCH and NEXT commands halt the computer before executing the command. This halt is intended to give the user a chance to be sure that the control switches are set correctly before any tape is punched. Pressing the CONTINUE key on the console will allow the Editor to proceed with the command. The editor remains in command mode after execution of any command which punches tape.

The Editor is designed to minimize the possibility of illegal or meaningless characters being punched into a source tape, therefore the illegal codes 340-376 and 140-177, and most illegal control characters will not be punched. In this way a tape containing illegal characters may be corrected by simply reading it into the Editor and punching it out.

Depending on the position of switch register bit 10 (see explanation of control switches) punching will be done by either the Teletype punch or the high-speed punch.

Command	Action and Explanation
P↵	PUNCH the entire contents of the text buffer.
nP↵	PUNCH line n only.
m,nP↵	PUNCH lines m through n, inclusive. (where m must be less than n).

None of the above commands outputs a form feed character following the text.

NOTE TO LOW SPEED PUNCH USERS: The F and T commands do not halt the computer before punching tape. The user must therefore, turn on the punch immediately after typing the carriage return.

F↵	FORM FEED. This command causes the punching of four blanks, a form feed character, and approximately two inches of blank tape. If using low-speed punch, turn punch off before typing command then turn on immediately after typing carriage return.

Command	Action and Explanation
T↲	TRAILER. This command causes about four inches of blank tape to be punched. If using low-speed punch, turn punch off before typing command then turn on immediately after typing carriage return.
N↲	NEXT. This is a utility command which combines the functions of four commands. It punches the contents of the buffer, punches some blank tape, a form feed and more blank tape, kills the buffer, and reads in the next page of text from the reader specified by switch register bit 11 (i.e., it executes P↲ , F↲ , K↲ , R↲).
nN↲	Executes the above sequence n times. The Editor halts only before the first punching. If n is greater than the number of pages of input tape the command will proceed in the specified sequence until it reads the end of the input tape, then it will return to command mode. (If using Teletype reader, when tape runs out type CTRL/FORM to return to command mode).

2.3 EDITING COMMANDS

The following commands permit deletion, alteration, or expansion of text in the buffer.

Command	Action and Explanation
K↲	KILL the entire page in the buffer. The values of special characters "/" and "." are set to zero. The Editor remains in command mode.
nD↲	DELETE line n. Line n is removed from the text buffer. The numbers of all succeeding lines are reduced by one, as is the line count.
m,nD↲	DELETE lines m through n, inclusive. The line following n becomes the new line m and the rest of the lines are renumbered accordingly. The value of the current line counter, ".", is equal to the number of the line proceeding the deleted line or lines. The Editor remains in command mode after all DELETE operations.
nI↲	INSERT the typed text before line n, until a form feed (CTRL/FORM) is encountered. The Editor enters text mode to accept input. The first line typed becomes the new line n. Rubouts are recognized. Both the line count and the numbers of all lines following the insertion are increased by the number of lines inserted. The value of "." is equal to the number of the last line inserted. To reenter the command mode, the CTRL/FORM key combination must be entered to terminate text mode. If this is not done, all subsequent commands will be interpreted erroneously as text and entered in the program immediately after the insertion.
I↲	INSERT without an argument will insert text before line 1.
nC↲	CHANGE line n. Line n is deleted, and the Editor enters text mode to accept input. The user may now type in as many lines of text as he desires in place of the deleted line. Rubouts are recognized during any CHANGE operation. If more than one line is inserted, all subsequent lines will be automatically renumbered and the line count will be updated appropriately.

Command	Action and Explanation
m,nC↲	CHANGE lines m through n, inclusive (m must be numerically less than n). Lines n through m are deleted and the Editor enters text mode allowing the user to type in any number of lines in their place. All subsequent lines will be automatically renumbered to account for the change and the line count will be updated. After any CHANGE operation, return to command mode is accomplished by entering a form feed (CTRL/FORM key combination) to terminate input. After a CHANGE the value of the current line counter, ".", is equal to the number of the last line of the change. Lines which are changed or deleted do not physically disappear from the buffer area, thus the space they occupied is not recovered upon completion of the command. This being true, it is possible to overflow the buffer by changing, or deleting and inserting lines. This possibility may be effectively eliminated by logically segmenting a program on paper tape into "pages" of 50 to 60 lines. This is done by punching groups of 50 lines followed by a form feed character (see output commands). There is a way to retrieve lost space in some cases by use of the SEARCH command (see below).
m,n$kM↲	MOVE lines m through n inclusive to before line k (m must be numerically less than n and k may not be in the range between m and n). Lines m through n are moved from their current position and inserted before line k. The lines are renumbered after the move is completed although the value of the current line pointer, ".", is unchanged. Moving lines does not use any additional buffer space. A line or group of lines may be moved to the end of the buffer by specifying k as "/+1". Example: 1,10$/+M). Since the MOVE command requires three arguments, it must have three arguments to move even one line. This is done by specifying the same line number twice. Example: 5,5$23M). This will move line 5 to before line 23. The Editor remains in command mode after a move command.
G↲	GET and list the next line which begins with a tag. The Editor begins with the line following the current line (line .+1) and tests for a line which does not begin with a tab, slash or a space. This will most often be a line beginning with a tag. It might also be a line containing an origin.

```
Example:        TAD - this is the current line
                DCA -
           /THIS IS A COMMENT
            HERE,0      - this line would be typed out by the command G↲
                TAD  -
                ISZ  -
              *5000        this line would also be typed if another G↲ were
                           typed.
```

Command — Action and Explanation

nG↲

GET and list the first line after line n which begins with a tag. The Editor begins with line n and tests it and each succeeding line as described above.

Both G and nG update the current line counter after finding the specified line. However, if either version of the GET command reaches the end of the buffer before finding a line beginning with other than a tab, slash or space, the current line counter retains the value it had before the GET was issued and a "?" is typed to indicate that no tagged line was found. The Editor remains in command mode after a GET command.

nS↲

SEARCH line n for the character specified after the carriage return. Allow modification of line when character is found.

The SEARCH command is one of the most useful functions in the Editor. It is also structured somewhat differently from the other Editor commands. After terminating the command nS with a carriage return the user has told the Editor to SEARCH line n, but he hasn't specified what to search for. The Editor is, therefore, waiting for the user to type a character. The character he types is taken as the object of the search but is not echoed. The Editor instead immediately begins typing out the specified line. After typing the character for which it is searching the Editor stops. All of the editing features are then available to the user. He may proceed using any of the following.

a. Delete the entire typed portion of the line by typing " ←—— " (back arrow).

b. Delete the entire untyped portion and terminate the line and the search by typing ↲ (carriage return).

c. Delete from right to left one of the typed characters for each \ (rubout) typed.

d. Insert characters after the last one typed simply by typing them.

e. Insert a carriage return and line feed, thus dividing the line into two, by typing ↓ (line feed).

f. Continue searching to the next occurrence of the search character by typing CTRL/FORM. When typing stops all options are again available.

g. Change the search character and continue searching by typing CTRL/BELL followed by the new search character.

Each time the Editor types the character for which it is searching, typing stops and all or any combinations of the above operation may be carried out.

m,nS↲

SEARCH lines m through n inclusive in the same way as described above. The search character is input after the carriage return and all of the options are available. The only difference is in point b. Typing ↲ (carriage return) deletes the entire untyped portion and terminates that line, but the search continues on the next line.

By typing CTRL/BELL to change search characters, all editing of a single line may be done in one pass. Clearly, typing CTRL/BELL twice will cause the search to proceed to termination, since the search character will now be BELL which is not stored in the buffer.

Command	Action and Explanation
S↲	An additional feature is available to the more sophisticated user: by typing S with no arguments the entire buffer may be searched for occurrences of a single character. It must be remembered, however, that as with every CHANGE command, every SEARCH command uses additional buffer space for storage of the new line. This is obviously necessary, since the program can have no prior knowledge of whether the size of the line will be less than, greater than, or equal to that of the old line, and it must therefore assume that it will be larger. As the entire buffer is searched, a new image of the text is created in core that is guaranteed to occupy the same or less space than previously, since all deleted spaces have been removed. The Editor recognizes this and immediately moves the text image back to the top of the buffer space. Thus, the only prerequisite to condensing the text image is that there be enough core space left to contain another image of the edited text.

SECTION 3
OPERATING PROCEDURE

This chapter describes the sequence of operations necessary to load, edit, and punch out a corrected symbolic program tape, and gives examples of the use of the Editor.

After the Editor has been loaded, it may be used to read into the text buffer a page of the symbolic program to be corrected. When the page has been read in and a form feed code has been encountered, the Editor rings a bell to signal the user that the Editor is in the command mode. Corrections and additions may then be either typed in from the Teletype keyboard or inserted from paper tape via the reader. Individually numbered lines may be listed in any order permitting insertions, deletions, or changes. Text may be inserted between specified lines, moved or appended to the end of a section. Individual lines, groups of lines, or an entire page may be deleted upon a single command. To ensure that a tape is correct, desired portions or an entire page may be listed before punching. Finally, the corrected lines, groups of lines, or the entire page may be listed and/or punched. The original text remains available in the core buffer in case further corrections are necessary.

The following paragraphs give the detailed procedure for loading the Editor and a symbolic tape, making required corrections, and punching the corrected symbolic tape.

3.1 LOADING SEQUENCE

Before editing can begin, the Symbolic Editor program must be loaded into core with the Binary Loader, and the symbolic program tape to be corrected must be read into the core text buffer. The loading of the symbolic tape is performed by the Editor itself under keyboard control.

3.1.1 Loading the Binary Loader

The Binary Loader is loaded into core by the Read-In Mode (RIM) Loader. The RIM Loader itself is initially placed into core memory by PDP-8 console keys and switches. When this has been done, the RIM Loader is used to load the Binary Loader into core by means of the following procedure.

a. Put the Binary Loader tape in the Teletype reader.

b. Set the SWITCH REGISTER (SR) to 7756, the starting address of the RIM Loader.

c. Press the LOAD ADDRESS key and then the START key.

d. Turn on the reader and wait until the tape is completely read in. When the reader stops, the Binary Loader is in memory.

3.1.2 Loading The Symbolic Editor With The Binary Loader

To load the Symbolic Editor with the Binary Loader:

a. Place the Editor tape in the proper reader, either Teletype or photoelectric.

b. Set the SR to 7777 (the starting address of the BIN Loader) press the LOAD ADDRESS key, and turn on the reader.

c. (If using the high-speed reader set SR bit 0 to 0) then press the START key.

d. When the reader stops, the Editor is in core memory.

If the ACCUMULATOR (AC) does not contain zero when the reader halts, a checksum discrepancy exists which indicates that the Editor tape has been read in incorrectly. Load the tape again by repeating the procedure described above.

To start the Editor program initially set the SR to 200, press the LOAD ADDRESS key and then the START key. The loading of the symbolic tape to be edited and all subsequent operations are performed through the use of the Editor by giving appropriate commands from the Teletype keyboard.

3.1.3 Loading A Symbolic Tape Using The Editor

a. Set switches as indicated in Section 1.4 on page 1-6.

b. Place the symbolic tape of the program to be corrected in the appropriate paper tape reader.

c. At the keyboard, type the READ command followed by a carriage return (i.e., type R ↲). If using the Teletype reader, turn it on now. The symbolic tape will be read into the text buffer.

d. The Editor will continue reading the tape until the form feed code is encountered at the end of the tape (see Input Commands). If the tape contains no form feed code, and the Teletype reader is being used for input, strike the FORM/CTRL key combination after the tape has been read in. Upon recognizing the form feed character, the Editor enters the command mode and rings a bell to indicate that it is ready for the first command.

CAUTION

> When using the Teletype reader, if the form feed code is encountered before the symbolic tape has completely read in (as indicated by the ringing of the bell), turn off the paper tape reader. Otherwise, characters on tape will be interpreted as commands to the Editor. The section of tape read in up to the form feed code should then be edited first before proceeding with the remainder of the tape.

3.2 EDITING A SYMBOLIC TAPE

The actual editing procedure depends, of course, on a particular user's requirements. The general procedure is illustrated in the example that follows. For input, editing, and output commands that may be given to the Editor, refer to the detailed explanation of the Command Structure, Command Repertoire and Special Characters and Functions under Operating Features (see pages 1-1, 1-2 and 1-6) or see the corresponding summaries of commands and special characters in the appendix. Observe also the following operating notes and precautions.

a. Terminate each command to the Editor by striking the RETURN (carriage return) key. This directs the Editor to execute the command.

b. After a command to insert, change, or append text to the symbolic program has been executed, the Editor remains in the text mode until the operator hits the FORM/CTRL key combination on the teletypewriter. This combination generates the form feed code character, which tells the Editor to return to the command mode.

c. If a great deal of text is added during the course of editing, the text buffer may overflow (full capacity is approximately 60 lines of heavily commented text or 340 uncommented). When the buffer storage limit has been exceeded the program will continue operation and insertions and changes may still be made but an additional bell will be rung for every location that is used beyond the buffer limit. In order to extend the buffer so that more characters may be read, the user may change location 1 to contain the address of the last location in the buffer to be used. It is suggested that this number not be larger than 7570 to protect the binary loader. At this point the buffer is capable of holding over 5000 decimal characters.

d. The Editor may be stopped at any time by pressing the STOP key; to continue press the CONTINUE key. If it is desired to restart the Editor without disturbing the buffer, place octal address 177 in the SR and press the LOAD ADDRESS and the START keys. If it is desirable to clear the buffer and then restart, place octal address 176 in the SR and press the START key.

3.2.1 Error Messages

The proper rules for giving commands must be observed during editing, as is explained under Operating Features (page 1-1). If commands are given in an incorrect format, or if arguments are either missing, erroneous or extraneous, the Editor will respond by typing out a question mark. Notice that some commands can legitimately take from zero to two arguments, and one takes three. In general, if an argument is either missing or extraneous, the Editor types out ? and ignores the command. Similarly, if a negative argument is encountered or an illegitimate command string is typed, the Editor again responds with the error message ?.

3.3 PUNCHING THE CORRECTED SYMBOLIC TAPE

The procedure for punching out the corrected symbolic tape depends to some extent on the user's requirements. The general sequence of steps is given below.

a. As desired, give the output command to punch out either line n of the text (nP↵), lines m through n (m,nP↵), the entire text (P↵), form feeds (F↵) or blank tape (T↵).

b. After the carriage return (↵) following the punch command has been received, the computer will halt. Turn on the punch and check the switches.

c. Press the CONTINUE key at the console to start punching.

If a tabulation has been produced by striking the CTRL/TAB key combination spaces or a tab character will be generated on output depending on switch register bit 1 (see SWITCH OPTIONS).

d. If using the telepunch, after a tape has been punched, turn off the punch before typing any further commands. If this is not done, the control codes typed in will be punched on the symbolic tape.

d. Punching the symbolic program does not delete it from memory. The page remains in the text buffer in core until the KILL command (K↵) is given to erase it. If it is desired to read another tape into the buffer, the user must first delete the entire page of text (K↵). Remember that the recommended page length, as delimited by the form feed, is approximately 60 lines. However, the Editor will accept more text if necessary.

3.4 EXAMPLE OF USE

The following detailed example of the editing of a page of text is intended to familiarize the user with the basic operations of the Symbolic Editor. Where details of the loading sequence and operating procedures are not shown, it is assumed that the user has followed the correct procedures previously explained.

This example concerns a program for adding up numbers stored in locations 200_8 through 207_8 of the computer, with the answer to be stored in location 410_8. The program is to start in location 600. The program listing is shown in Figure 3-1.

```
/ADD UP NUMBERS
*600
BEGN,     HLT
/TO START THE PROGRAM.  HIT "CONTINUE" ON CONSOLE
/
/THE NEXT FIVE INSTRUCTIONS INITIALIZE THE ROUTINE
          CLA               /CLEAR THE ACCUMULATOR
          TAD M10           /LOAD AC WITH THE NUMBER - 10
          DCA COUNTR        /PUT INTO COUNTER
          TAD TWOHUN        /LOAD AC WITH FIRST ADDRESS
          DCA POINTR        /PUT IT INTO POINTER
```

Figure 3-1 Program Listing of Addition Routine

```
/
/THE NEXT SEVEN INSTRUCTIONS ARE THE PROGRAM ITSELF
BEGN,     TAD I POINTR          /ADD NEXT NUMBER
          ISZ POINTR            /INDEX POINTER
          ISZ COUNTR            /INDEX COUNTER, IS IT ZERO?
          JMP BEGN              /NO; CONTINUE ADDING
          DCA I ANSWER          /YES; STORE ANSWER
          HLT                   /HALT
          JMP BEGN+1
/
/THE NEXT THREE REGISTERS CONTAIN THE CONSTANTS
M10,      -10                   /NEGATIVE TALLY NUMBER
TWOHUN,   200                   FIRST ADDRESS IN BUFFER
ANSWER    410

/THE NEXT TWO REGISTERS ARE RESERVED FOR VARIABLES
COUNTR,   0
POINTR,   0

$
```

Figure 3-1 Program Listing of Addition Routine (continued)

Let us assume that we have attempted to assemble this program using the PDP-8 Symbolic Assembler (PAL III). On PASS1, however, the Assembler typed the following:

```
DT   BEGN        AT    0606

UA   ADDRES      AT    0616

UA   ANSWER      AT    0612
BEGN      0600

UA   BUFFER      AT    0616
COUNTR    0620

UA   FIRST       AT    0616

UA   IN          AT    0616

M10       0615
POINTR    0621
TWOHUN    0616
```

The message DT BEGN AT 0606 signifies that the programmer has mistakenly used identical tags to specify two different addresses. An inspection of the program listing (Figure 2-1) shows that the tag BEGN has, indeed, been duplicated. It appears in line 3 of the listing as BEGN, HLT, then in line 14, starting with BEGN, TAD I POINTR. (Since the line numbers are implicit only, they are not shown in the example; they may be obtained by counting from the top down in Figure 2-1). To correct the situation, the Symbolic Editor was read in with the Binary Loader, as explained under Loading Sequence in this section. The symbolic tape to be corrected was then loaded through the Editor by means of the READ (R↵) command. A series of commands were given. (The lines typed by the Editor have been underlined for clarity.)

```
R
14L
BEGN,               TAD I POINTR    /ADD NEXT NUMBER
14C
ADDR,               TAD I POINTR    /ADD NEXT NUMBER
17L
                    JMP BEGN        /NO; CONTINUE ADDING
17C
                    JMP ADDR        /NO; CONTINUE
13, 18L
/THE NEXT SEVEN INSTRUCTIONS ARE THE PROGRAM ITSELF
ADDR,               TAD I POINTR    /ADD NEXT NUMBER
                    ISZ POINTR      /INDEX POINTER
                    ISZ COUNTR      /INDEX COUNTER, IS IT ZERO?
                    JMP ADDR        /NO; CONTINUE
                    DCA I ANSWER    /YES; STORE ANSWER
25L
ANSWER   410
25C
ANSWER,  410
24L
TWOHUN,  200                         FIRST ADDRESS IN BUFFER
24C
TWOHUN,  200                        /FIRST ADDRESS IN BUFFER
.-1,.+2L
M10,                                /NEGATIVE TALLY NUMBER
TWOHUN,  200                        /FIRST ADDRESS IN BUFFER
ANSWER,  410
```

Having made the desired corrections, the programmer finally asks the Editor to list the entire text by giving the LIST (L↵) command, but still withholds the PUNCH command (P↵), pending final corrections. A new program listing is printed out and the entire text is preserved in the buffer. The programmer now punches the entire text onto paper tape by giving the PUNCH (P↵) command and hitting CONTINUE on the console.

```
/ADD UP NUMBERS
*600
BEGN,     HLT
/TO START THE PROGRAM. HIT "CONTINUE" ON CONSOLE
/
/THE NEXT FIVE INSTRUCTIONS INITIALIZE THE ROUTINE
          CLA                 /CLEAR THE ACCUMULATOR
          TAD M10             /LOAD AC WITH THE NUMBER - 10
          DCA COUNTR          /PUT INTO COUNTER
          TAD TWOHUN          /LOAD AC WITH FIRST ADDRESS
          DCA POINTR          /PUT IT INTO POINTER
```

```
/
/THE NEXT SEVEN INSTRUCTIONS ARE THE PROGRAM ITSELF
ADDR,           TAD I POINTR    /ADD NEXT NUMBER
                ISZ POINTR      /INDEX POINTER
                ISZ COUNTR      /INDEX COUNTER, IS IT ZERO?
                JMP ADDR        /NO; CONTINUE
                DCA I ANSWER    /YES; STORE ANSWER
                HLT             /HALT
                JMP BEGN+1

/THE NEXT THREE REGISTERS CONTAIN THE CONSTANTS
M10,            -10             /NEGATIVE TALLY NUMBER
TWOHUN,         200             /FIRST ADDRESS IN BUFFER
ANSWER,         410

/THE NEXT TWO REGISTERS ARE RESERVED FOR VARIABLES
COUNTR,         0
POINTR,         0

$
```

The PASS1 result of assembling this program is:

```
    ADDR        0606
    ANSWER      0617
    BEGN        0600
    COUNTR      0620
    M10         0615
    POINTR      0621
    TWOHUN      0616
```

APPENDIX A
SUMMARY OF SYMBOLIC EDITOR OPERATIONS

A1 SPECIAL KEY FUNCTIONS

Carriage Return (RETURN Key)	Text mode - Enter the line in the text buffer.
	Command mode - Execute the command
Back Arrow (⟵)	Text mode - Cancel the entire line of text, continue typing on same line.
	Command mode - Cancel command. Editor issues a ? and carriage return/line feed.
Rubout (\ )	Text mode - Delete from right to left one character for each rubout typed. Does not delete past the beginning of the line. Is not in effect during a READ command.
	Command mode - Same as back arrow.
Form Feed (CTRL/FORM Key Combination)	Text mode - End of inputs return to command mode.
Period (.)	Command mode - Current line counter used as argument alone or in combination with + or − and a number (., .+5L).
Slash (/)	Command mode - Value equal to number of last line in buffer. Used as argument (/-5,/L).
Line Feed (↓)	Text mode - Used in SEARCH command to insert a CR/LF combination into the line being searched.
	Command mode - List the next line (equivalent to .+1L).
ALT Mode (ALT key) Escape (ESC key)	Command mode - List the next line (equivalent to .+1L).
Left Angle Bracket (<)	Command mode - List the previous line (equivalent to .−1L).
Equal Sign (=)	Command mode - Used in conjunction with . and / to obtain their value (.=27).
Colon (:)	Command mode - Lower case character, same function as = .
Tabulation (CTRL/TAB Key Combination)	Text mode - Produces a tabulation which on output, is interpreted as spaces or a tab character/rubout combination depending on a switch option.

A2 SWITCH OPTIONS

Switch	Position	Meaning
0	0	Read input tape as is
	1	Convert spaces to tabulations on input
1	0	Output tabulations as spaces
	1	Output tab character rubout combination for each tabulation
2	0	Normal operation
	1	Suppress output
10	0	Low speed output
	1	High speed output
11	0	Low speed input
	1	High speed input

A3 COMMAND SUMMARY

Command	Format(s)	Meaning
READ	R↵	Read incoming text from reader and append to buffer until a form feed is encountered.
APPEND	A↵	Append incoming text from keyboard to any already in buffer until a form feed is encountered.
LIST	L↵	List the entire buffer.
	nL↵	List line n.
	m,nL↵	List lines m through n inclusive.
PUNCH	P↵	Halt. Upon striking CONTINUE key on console, punch the entire buffer.
	nP↵	Halt. Upon CONTINUE, punch line n.
	m,nP↵	Halt. Upon CONTINUE, punch lines m through n inclusive.
FORM FEED	F↵	Punch trailer, punch a form feed (214), punch trailer.
TRAILER	T↵	Punch four inches of trailer.
NEXT	N↵	Punch the entire buffer and a form feed, Kill the buffer and Read the next page.
	nN↵	Repeat the above sequence n times.
KILL	K↵	Kill the buffer.
DELETE	nD↵	Delete line n of the text.
	m,nD↵	Delete lines m through n inclusive.
INSERT	I↵	Insert before line 1 all the text from the keyboard until a form feed is entered.
	nI↵	Insert before line n until a form feed is entered.

Command	Format(s)	Meaning
CHANGE	nC↵	Delete line $\underline{n}$, replace it with any number of lines from the keyboard until a form feed is entered.
	m,nC↵	Delete lines $\underline{m}$ through $\underline{n}$, replace from keyboard as above until form feed is entered.
MOVE	m,n$kM↵	Move lines $\underline{m}$ through $\underline{n}$ inclusive to before line $\underline{k}$.
GET	G↵	Get and list the next line beginning with a tag.
	nG↵	Get and list the next line after line $\underline{n}$ which begins with a tag.
SEARCH	S↵	Search the entire buffer for the character specified (but not echoed) after the carriage return. Allow modification when found.
	nS↵	Search line $\underline{n}$, as above, allow modification.
	m,nS↵	Search lines $\underline{m}$ through $\underline{n}$ inclusive, allow modification.

A4 REFERENCES

Binary Loader - DEC 08-LBAA-D

RIM Loader - DEC 08-LRAA-D

digital

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Printed in U.S.A.