DEC-11-EEDA-D

PDP-11

EDIT-11 TEXT EDITOR

PROGRAMMER'S MANUAL

for the

Disk Operating System

For additional copies, order No. DEC-11-EEDA-D from Digital Equipment Corporation, Direct Mail, Building 1-1, Maynard, Massachusetts 01754

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

First Printing, May 1971

Your attention is invited to the last two pages of this document. The "How to Obtain Software Information" page tells you how to keep up-to-date with DEC's software. The "Reader's Comments" page, when filled in and mailed, is beneficial to both you and DEC; all comments received are acknowledged and are considered when documenting subsequent documents.

This document is for information purposes and is subject to change without notice.

Associated Documents:

PDP-11 Disk Operating System Monitor,
Programmer's Handbook, DEC-11-MWDA-D

PDP-11 PAL-11R Assembler,
Programmer's Manual, DEC-11-ASDB-D

PDP-11 ODT-11R Debugging Program,
Programmer's Manual, DEC-11-OODA-D

PDP-11 Link-11 Linker and Libr-11 Librarian,
Programmer's Manual, DEC-11-ZLDA-D

PDP-11 PIP, File Utility Package,
Programmer's Manual, DEC-11-PIDA-D

PDP-11 FORTRAN IV Compiler and Object Time System,
Programmer's Manual, DEC-11-KFDA-D

The following are trademarks of Digital Equipment Corporation:

DEC	PDP
FLIP CHIP	FOCAL
DIGITAL (logo)	COMPUTER LAB
UNIBUS	OMNIBUS

P R E F A C E

This manual describes the features and operation of the Edit-11 Text Editor for the PDP-11 Disk Operating System (DOS). The manual assumes familiarity with the Disk Operating System Monitor (see PDP-11 Disk Operating System Monitor, Programmer's Handbook, DEC-11-MWDA-D).

In addition to the Edit-11 Text Editor and the Monitor, the Disk Operating System software includes:

PAL-11R Assembler
ODT-11R Debugging Program
PIP, File Utility Package
Link-11 Linker and Libr-11 Librarian
FORTRAN IV

TABLE OF CONTENTS

CHAPTER 1

EDIT-11

The PDP-11 Disk Editor (Edit-11) is a text editing program for use with the PDP-11 Disk Operating System. Operated by user commands from the keyboard, Edit-11 will read ASCII files from any device, make directed changes, and write on any device. In addition to basic editing functions, Edit-11 provides for command macros and multiple input and output files.

After initialization, Edit-11 indicates readiness to accept a command string by printing an asterisk (*). The user, through appropriate commands typed on the keyboard, causes Edit-11 to (1) read a "page" of text from the input file, (2) locate text in the buffer to be changed, (3) execute and verify changes, (4) output the page on the output file, and (5) proceed in the same manner to the end of his input file. After the last page of input is processed, the user closes his output file and returns to DOS Monitor control. The basic editing process above is quite flexible and made simple by Edit-11's large set of commands.

Each command to Edit-11 consists of one or two letters. Commands are typed in a string and sent to Edit-11 as a command string with appropriate arguments and text objects included. Edit-11 executes the command string, acting on the contents of its internal Page Buffer, in which the input source is stored.

The basic editing process can be divided into three sequential steps:

1. Reading of input text into a buffer internal to Edit-11.
2. Changing the text stored in the buffer.
3. Outputting the revised text to a new file.

CHAPTER 2

COMMAND MODE AND TEXT MODE

Whenever Edit-11 prints an * on the teleprinter, it is waiting to receive a command string, and is said to be in command mode. While most commands operate exclusively in command mode, there are eight commands that require additional text (the text object) to operate upon. Hence, although one can delete n characters by simply typing nD, the command to insert text, I, must be followed by the text to be inserted, the text object.

There are two ways to provide the required text object for a command. If the text object is small enough, Edit-11 can accept text in command mode if it is separated from the rest of the command string by delimiters. If the text object is long, or contains carriage return or line feed characters, then the user must cause Edit-11 to enter text mode.

Edit-11 processing begins in command mode. When you type a command string, no action occurs until you complete the string by typing the RETURN key (symbolized as <CR>). What happens next depends on the last character in the command string. If the last character in the string is a command requiring a text object, Edit-11 will enter text mode. If the last character is not a text command, Edit-11 expects to find any necessary test objects embedded in the command string itself, and will stay in command mode as it executes the command string.

Text can be accepted in command mode if the text contains no carriage return or line feed characters, and is small enough to fit on a single typed line with the command string. When Edit-11 finds a text command which is not the last command in the command string, it looks at the next character in the string as a text delimiter. All characters between this and the next occurrence of the delimiter are considered the text object for the command. Thus, any ASCII character which does not appear in the text object is a valid delimiter.

The following example inserts OBJECT , then prints the entire buffer on the teleprinter (B/L). Note the delimiters (#) of the text object (OBJECT) for the Insert command (I).

*I#OBJECT# B/L

If the text object is lengthy or contains carriage return or line feed characters, it can only be accepted if Edit-11 is in text mode. To enter text mode, type the text command of interest as the last character in the command string. Then type the RETURN key. When Edit-11 executes this command, it will enter text mode, and accept text input. It continues to accept text until the user terminates text mode and reenters command mode by typing the LINE FEED key (symbolized as <LF>).

Note that typing the RETURN key always causes the physical return of the printing head to the beginning of the line, and automatically generates a line feed, thereby advancing the carriage to a new line. In text mode, the RETURN key serves these mechanical functions, allowing you to continue typing at the beginning of a new line, and at the same time enters a carriage return and line feed character into the text.

These are both counted as characters even though they do not print. When you wish to terminate text mode and reenter command mode, you must type the LINE FEED key. A typed line feed is not considered to be part of the text unless it is the first character entered in text mode.

The following example inserts three lines of text into the buffer by entering text mode. Note that Edit-11 signifies its return to command mode by printing an *.

```
*I <CR>

 BLOOP:      MOV     #17ØØØ,(R3)+      <CR>
             BIT     (R4)+,MASK        <CR>
             BNE     BLOOP             <CR>

 <LF>
*
```

The commands which require text objects are Insert, Get, wHole, Edit wHole, Position, Edit Position, Change and eXchange. (Capitalized letters represent the actual commands typed.)

CHAPTER 3

COMMAND SYNTAX

3.1 THE CHARACTER LOCATION POINTER (DOT)

Almost all Edit-11 commands function with respect to a movable reference point, Dot. This character pointer is normally located between the most recent character operated upon and the next character; and, at any given time, can be thought of as "where Edit-11 is" in your text. As will be seen shortly, there are commands which move Dot anywhere in the text, thereby redefining the "current location" and allowing greater facility in the use of the other commands.

3.2 MARK

In addition to Dot, a secondary character pointer known as Mark also exists in Edit-11. This less agile pointer is used with great effect to Mark or "remember" a location by moving to Dot and conditionally remaining there while Dot moves on to some other place in the text. Thus it is possible to think of Dot as "here" and Mark as "there". Positioning of Mark, which is referenced by means of the argument @, is discussed below in several commands.

3.3 CHARACTER-ORIENTED COMMAND PROPERTIES

Many Edit-11 commands are character-oriented; that is, the argument to the command specifies the number of characters in the Page Buffer the command is to act on. Hence, the number of characters specified by the argument n is the same in the forward (n) and backward (-n) direction. Carriage Return and line feed characters embedded between text lines are counted in character-oriented commands, and are indistinguishable from other characters.

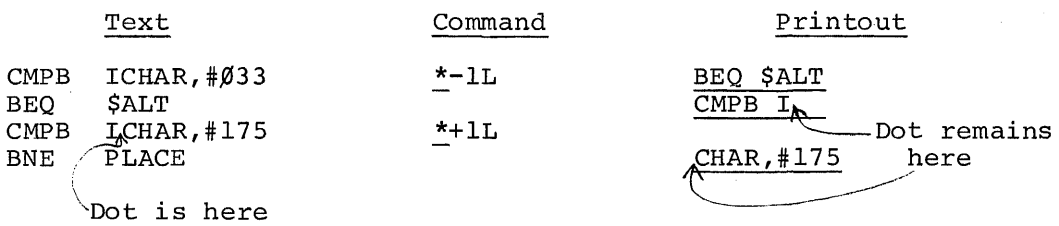
3.4 LINE-ORIENTED COMMAND PROPERTIES

Edit-11 recognizes a line as a unit by detecting a line terminator in the text. This means that ends of lines (line feed or form feed characters) are counted in line-oriented commands. This is important to know, particularly if Dot, which is a character location pointer, is not pointing at the first character of a line.

In such a case, an argument n will not affect the same number of lines (forward) as its negative (backward). For example, the argument

-1 applies to the character string beginning with the first character following the second previous end-of-line character and ending at Dot. Argument +1 applies to the character string beginning at Dot and ending at the first end-of-line character. If Dot is located, say, in the center of a line, notice that this would affect 1-1/2 lines back or 1/2 line forward respectively.

Example of List Commands -1L and +1L:



3.5 THE PAGE UNIT OF INPUT

Input files to Edit-11 are divided into smaller, more manageable segments called "pages". A page is terminated, and therefore defined, by a form feed (CTRL/FORM on keyboard) in the source text whenever a page division is desired. Although the unit of output is the line, the unit of input to Edit-11 is the page, and in order to make an ASCII file more usable, the user divides it into small segments by inserting form feeds in desired places before output occurs. Since more than one page of text can be in the buffer at the same time, it should be noted that the entire contents of the Page Buffer are available for editing.

3.6 ARGUMENTS

Some Edit-11 commands require an argument to specify the particular portion of text to be affected by the command or how many times to perform the command. In other commands this specification is implicit and arguments are not allowed.

The Edit-11 command arguments are described as follows:

1. n stands for any number from 1 to 32767 decimal and may, except where noted, be preceded by a + or -. If no sign precedes n, it is assumed to be a positive number.

   Where an argument is acceptable, its absence implies an argument of 1 (or -1 if a - is present). The role of n varies according to the command it is associated with.

2. Ø refers to the beginning of the current line.

3. @ refers to a Marked (designated) character location (see Section 3.2).

4. / refers to the end of text in the Page Buffer.

The roles of all arguments will be explained further with the corresponding commands which qualify them.

3.7 COMMAND STRINGS

A command string to Edit-11 consists of one or more commands typed on the same line. Spaces are not allowed between a command and its associated argument, although spaces may be inserted between commands themselves in the command string. The command string, including embedded text objects, must be less than 72 characters and is terminated by typing the RETURN key.

NOTE

Caution must be exercised when using spaces in text commands. If a space separates a text command from its object, the space is considered the delimiter. Hence:

1. I OBJECT B /L is legal.

2. I #OBJECT#B /L will use #OBJECT#B as the text object.

3. I !OBJECT!B/L is illegal because there is no second delimiter.

The following are all legal command strings:

```
*B /L B G#OBJECT# <CR>
*B/L BG#OBJECT#I#TEXT# B/L  <CR>
*BSKB3L  <CR>
*-3J -4C#TEXT# ØA L  <CR>
```

When the user types a command string to Edit-11, nothing happens until he types the RETURN key. Upon receipt of the return character, Edit-11 starts at the beginning of the command string and processes the commands one at a time. Should an error be encountered in the middle of the string, those commands before the error would be successfully completed, and those following the command in error would not be executed.

3.8 ERROR MESSAGES

Whenever Edit-11 receives a syntactically incorrect command, or cannot execute a command properly, it returns an error message of the form:

Xnnn	where X is either an S (System error) or W (Warning) and nnn is a numerical error code. (See Appendix A.)

If the error occurs after the first command in a command string, Edit-11 will follow the error code with a second line. The second line will be a copy of the command string being executed, with a ? following the character where Edit-11 found the error. The commands printed completely were executed and the commands not printed were not executed. Edit-11 then prints an * and awaits another command string.

Example:

```
*B /L G#TEXT# 6EF I#MORE#   <CR>
W314 ØØØØØØ
B /L G#TEXT# 6EF?
*
```

In most cases, Edit-11 performs no further action on receipt of an illegal command. In effect, the command string is truncated at the error point.

CHAPTER 4

INPUT/OUTPUT

When Edit-11 is initialized, it asks for a command string for the Command String Interpreter which serves to specify the input and output devices and files it will use. In this command string, the user must specify a primary output file, and may specify a primary input file and secondary input and output files if desired. (See section 6.1.)

4.1 PRIMARY INPUT AND OUTPUT

If two input files are specified, the first one specified is the primary input file. This file is the ASCII file you want to edit, and can only be read. The first output file specified is the primary output file and it is into this file that corrected text will be written. Essentially, Edit-11 will transform the primary input file into the primary output file under user direction. If the primary input and output files have the same name and are on the same device, Edit-11 creates the file EDITOR.TMP for output. When this file is closed (by the EX or EF command), the input file is renamed with .BAK as its extension. The output file, EDITOR.TMP is then renamed to the desired name. This backup facility is automatic, but it can be suppressed with the /B switch. (See Section 6.1.)

NOTE

Do not create a file of your own with the name EDITOR.TMP. It will be deleted the next time Edit-11 needs the name.

4.2 SECONDARY INPUT AND OUTPUT

If a second input or output file (or both) is specified, Edit-11 will allow text to be read or written from these files also. These files are utility files, designed to ease the process of editing. They are controlled by commands distinct from primary input and output commands. All secondary input/output commands begin with the letter E (ER, EW, EH, EP), while all primary input/output commands are one-letter commands.

CHAPTER 5

COMMANDS

5.1 INPUT AND OUTPUT COMMANDS

Five commands are available for reading in a page of text: The Read and Edit Read commands (section 5.1.1) are specialized input commands; the Next command (section 5.1.3) reads in a page after writing out the previous page; and the wHole and Edit wHole commands (sections 5.2.2.2 and 5.2.2.3) read in and write out text as part of a search for a specified character string.

Output commands either list text on the teleprinter or write text into an output file. The List command causes specified lines of text to be output to the teleprinter so that they may be examined. Write commands (Next, wHole, and Edit wHole also perform input), provide for the output of specified pages, lines, form feeds (for changing the amount of data that constitutes a given page), and leader/trailer for paper tape. Note that the process of outputting text does not cause Dot to move.

5.1.1 Read and Edit Read

Two ways of getting text into storage so that it can be edited are by means of the Read (R) and Edit Read (ER) commands. The command R causes a page of text to be read from the primary input file, while the command ER reads a page from the secondary input file. The read text is appended to the contents (if any) of the Page Buffer.

Text will be read in until either:

1. A form feed character is encountered;
2. The Page Buffer is 128 characters from being filled, or a line feed is encountered after the buffer has become 5ØØ characters from being full;
3. An end of data is detected on the input device.

Following executing of an R command, Dot and Mark will be located at the beginning of the Page Buffer.

An 8K system can accommodate about 4ØØØ characters of text. All

additional core memory is available for text storage, i.e., about 8ØØØ characters of text for each additional core memory bank.

NOTE

An attempt to overflow the storage area will cause the command (in this case, R) to stop executing. A W3Ø3 error message will be printed. No data will be lost.

5.1.2 List, Write and Edit Write

Output commands List, Write, and Edit Write can be described together, as they differ only in the device and file addressed. List (L) outputs to the teleprinter, Write (W) outputs to the primary output file, and Edit Write (EW) outputs to the secondary output file.

nL nW nEW	Lists Writes on primary output Writes on secondary output	the character string beginning at Dot and ending with the n^{th} end of line.
-nL -nW -nEW	Lists Writes on primary output Writes on secondary output	the character string beginning with the first character following the (n+1)th previous end of line and terminating at Dot.
ØL ØW ØEW	Lists Writes on primary output Writes on secondary output	the character string beginning with the first character of the current line and ending at Dot.
@L @W @EW	Lists Writes on primary output Writes on secondary output	the character string between Dot and the Marked location.
/L /W /EW	Lists Writes on primary output Writes on secondary output	the character string beginning at Dot and ending with the last character in the Page Buffer.

In addition to the above commands, there is a special list command that accepts no arguments.

V (Verify) lists the entire line containing Dot.

5.1.3 Next

Typing nN writes, onto the primary output file, the entire contents

of the Page Buffer (followed by trailer if paper tape is the output medium and a form feed is the last character in the buffer), deletes the contents of the buffer, and reads the Next page from the primary input file into the buffer. It performs this sequence n times. If there are fewer than the n pages specified, the command will be executed for the number of pages actually available, and error code W311 will be printed out. Following execution, Dot and Mark will be located at the beginning of the Page Buffer.

5.1.4 Form Feed and Trailer

F	Writes a Form feed character and four inches of null characters into the primary output file if the primary output is paper tape.
nT	Writes four inches of Trailer (null characters) n times on the primary output device if the primary output is paper tape.

5.2 EDITING COMMANDS

5.2.1 Commands to Move Dot and Mark

5.2.1.1 Beginning

B	moves Dot to the beginning of the Page Buffer

5.2.1.2 Jump and Advance

nJ	moves Dot forward past n characters.
nA	moves Dot forward past n ends-of-lines to the beginning of the succeeding line.
-nJ	moves Dot backward past n characters.
-nA	moves Dot backward to the first character following the (n+1)th previous end-of-line.
ØJ or ØA	moves Dot to the beginning of the current line
@J or @A	moves Dot to the Marked location.
/J or /A	moves Dot to the end of the Page Buffer

Notice that while n moves Dot n characters in the Jump command, its role becomes that of a line counter in the Advance command, However, because Ø, @, and / are absolute, their use with these commands overrides line/character distinctions. That is, Jump and Advance perform identical functions if both have either Ø, @ or / for an argument.

5.2.1.3 Mark

The M command Marks ("remembers") the current position of Dot for later reference in a command using the argument @. Note that only one position at a time can be in a marked state. Mark is also affected by the execution of the following commands:

C D H I K N R X EH ER

5.2.2 Search Commands

5.2.2.1 Get

The basic search command nG starts at Dot and Gets the n^{th} occurrence of the specified text object in the Page Buffer. If no argument is present, it is assumed to be 1. If the object is to follow the G command in the command string, it must be properly set off by delimiters. If G is the last character typed in the command string, Edit-11 will enter text mode and accept a search object of up to 72 characters.

This command sets Dot to the position immediately following the found character string.

If the search is unsuccessful, Dot will be at the end of the Page Buffer and W3Ø7 will be printed on the teleprinter before Edit-11 prints an *.

Examples:

1.

Text	Command	Effect
MOV @RMAX,@R5	2G <CR>	BEQ CK
ADD #6,(R5)+	CK <LF>	
CLR $CK3		
TST R2		
BEQ CKCR		
Dot was here		Dot is now here

2.

CMPB	ICHAR,#RUBOUT	G<CR>	BR
BEQ	SITE	TE<CR>	
BR	PUT	BR<LF>	
Dot			Dot

5.2.2.2 wHole

A second search command, nH, starts at Dot and looks through the wHole text file for the n^{th} occurrence of the character string you have specified as text object. It combines a Get and a Next such that if the search is not successful in the Page Buffer, the contents of the buffer

are written in the primary output file, the buffer contents are deleted, and a new page is read in from the primary input, where the search is continued. This will proceed until the search object is found or until the complete source text has been searched. In either case, Mark will be at the beginning of the Page Buffer.

If the search object is found, Dot will be located immediately following it. As in the Get command, if the search is not successful, Dot will be at the end of the buffer and a W3Ø7 will appear on the teleprinter. Upon completion of the command, Edit-11 will be in command mode. Note that an H command specifying a nonexistent search object can be used to close out an edit, i.e., copy all remaining text from the primary input file to the primary output file.

5.2.2.3 Edit wHole

A wHole search can be performed through the secondary input file with the EH command. The EH search is identical to the wHole command, except that the file searched is the secondary input, while the file written into is the primary output. Note that an EH search for a nonexistent object is a method for reading the entire secondary input file into the center or onto the end of the primary output file.

5.2.2.4 Position

A fourth search command, nP, is used to Position Edit-11 in the primary input file so that a Read command can read the desired page. Position combines the Get, Delete and Read commands so that if the search object is not found the Page Buffer is cleared and a new page is read from the primary input and searched. This process continues until the search object is found or the end of the file is reached. The Position search is a wHole search in which there is destruction of unsuccessfully searched pages.

The Position command works as follows:

1. The current buffer contents, if any, are output to the primary output device and the buffer is cleared.
2. A page is read into the buffer and searched.
3. If the search is unsuccessful, the Page Buffer is cleared and step 2 is repeated.
4. If the n^{th} occurrence is found, Edit-11 returns with the page in which the n^{th} occurrence resides in the buffer, with Dot following the search object and Mark at the beginning.

Note that before the search begins, the current Page Buffer is saved by Edit-11 automatically in the primary output file. If this buffer is not desired, it must be deleted by the user prior to searching.

Note also that the Position command is most useful as a means of placing Edit-11 in the input file. If the editing session wishes to create a new file out of the second half of the input file, a Position search will save time.

5.2.2.5 Edit Position

The nEP command is identical to the Position command, except that the input file read from and searched is the secondary input file. Note that any text in the buffer is output to the primary output file before the search begins.

5.2.3 Commands to Modify the Text

5.2.3.1 Insert

The Insert command (I) allows text to be inserted at Dot. After I is typed, Edit-11 needs the text object to be inserted. If text mode is used, up to 80 characters per line are acceptable, and execution of the command occurs when the LINE FEED key (which does not insert a line feed character unless it is the first character typed in text mode) is typed terminating text mode. At this point, Dot is located in the position immediately following the last inserted text character. If the Marked location was anywhere after the text to be inserted, Dot becomes the new Marked location.

As with the Read command, an attempt to overflow the Page Buffer will cause a W3Ø3 to be printed out followed by an * on the next line indicating that a command may be typed. All or part of the last typed line may be lost. All previously typed lines will be inserted.

Examples:

	Text	Command	Effect
1.	MOV #8.,EKOT Dot	I <CR> CN <LF>	MOV #8.,EKOCNT Dot
2.	Inserting a carriage return (and automatic line feed):		
	CLR R1CLR R2 Dot	I <CR> <CR> <LF>	CLR R1 CLR R2 Dot
3.	Inserting a single line feed:		
	LOOK WHAT HAPPENS HERE I<CR> Dot	<LF> <LF>	LOOK WHAT HAPPENS HERE Dot

5.2.3.2 Delete and Kill

The commands in this category are closely related to each other; they both erase specified text from the Page Buffer. The Delete command differs from the Kill command only in that the former accepts an argument, n, that counts characters to be removed, while the latter accepts an argument, n, that counts lines to be removed. Ø, @ and / are also allowed as arguments (see Section 3.6). After execution of either of these commands, Dot becomes the Marked location.

nD	Deletes the following n characters	nK	Kills the character string beginning at Dot and ending at the n^{th} end-of-line
-nD	Deletes the previous n characters	-nK	Kills the character string beginning with the first character following the (n+1)th previous end-of-line and ending at Dot.

ØD or ØK — Removes the current line up to Dot

@D or @K — Removes the character string bounded by Dot and Mark

/D or /K — Removes the character string beginning at Dot and ending with the last character in the Page Buffer.

Examples:

	Text	Command	Effect
1.	;CHECK THE MOZXDE Dot	-2D	;CHECK THE MODE Dot
2.	;IS IT A TAB OR ;IS IT A CR Dot	2K	;IS IT A TAB Dot

5.2.3.3 Change and eXchange

The Change (C) and eXchange (X) commands can be thought of as two-phase commands combining, respectively, an Insert followed by a Delete, and an Insert followed by a Kill. After the Change or eXchange command is typed, a text object is required to be inserted. If ±n is used as the argument, it is then interpreted as in the Delete (character-oriented) or Kill (line-oriented) commands, and accordingly removes the indicated text. Ø, @, and / are also allowed in arguments.

commands combining, respectively, an Insert followed by a Delete, and an Insert followed by a Kill. After the Change or eXchange command is typed, a text object is required to be inserted. If ±n is used as the argument, it is then interpreted as in the Delete (character-oriented) or Kill (line-oriented) commands, and accordingly removes the indicated text. Ø, @, and / are also allowed as arguments.

nC xxxx xxxx	Changes the following n characters	nX xxxx xxxx	eXchanges the character string beginning at Dot and ending at the n^{th} end-of-line
-nC xxx	Changes the previous n characters	-nX xxx	eXchanges the character string beginning with the first character following the (n+1)th previous end-of-line and ending at Dot

ØC xxxx xxxx	or	ØX xxxx xxxx	Replaces the current line up to Dot
@C xxx xxx	or	@X xxx xxx	Replaces the character string bounded by Dot and the Marked location
/C xxx	or	/X xxx	Replaces the character string beginning at Dot and ending with the last character in the Page Buffer.

Again, the use of absolute arguments Ø, @, and / overrides the line/character distinctions that n and -n produce in these commands.

If the Insert portion of these commands is terminated because of attempting to overflow the Page Buffer, data from the latest line may have been lost, and text removal will not occur. Such buffer overflow might be avoided by separately executing a Delete or Kill followed by an Insert, rather than a Change or eXchange, which does an Insert followed by a Delete or Kill.

Examples:

Text	Command	Effect
;A LINE FEED IS HERE ;THIS ;IS ON Dot ;FOUR ;LINES Dot	-9C<CR> TAB<LF> 2X<CR> PAPER<LF>	;A TAB IS HERE ;THIS ;IS ON ;PAPER Dot

5.2.4 Utility Commands

The following commands provide easier paths for more general editing functions.

5.2.4.1 Save

The nS command copies the n lines beginning at Dot into an external buffer called the Save Buffer. Dot does not change, nor is the saved text deleted. Any previous contents of the Save Buffer are destroyed. If there is not enough room in storage to copy the text specified, a W303 error message will be printed and none of the text will be saved. Note that Save is a line-oriented command, and operates only in the forward direction; that is, negative arguments are not allowed.

5.2.4.2 Unsave

The U command inserts the entire contents of the Save Buffer at Dot, and Dot moves to follow the last character Unsaved. The Save Buffer itself is retained, and the same Save Buffer can be Unsaved as often as desired. If the action of unsaving would result in overflow of the Page Buffer, a W31Ø error message is printed and the Unsave does not occur.

Note that Save and Unsave provide convenient tools for moving blocks of text or inserting the same block of text in several places.

5.2.4.3 Execute Macro

The nEM command is a means of performing the same Edit-11 command string n times. When the EM command is received, the contents of the Save Buffer up to the first <CR> character are interpreted as a command string and executed n times. The macro is subject to the same rules as any typed command string, and in addition, a macro string may not contain another macro call.

Thus, to execute a macro, it is required to insert the macro in the Page Buffer, save it, then execute.

Example:

The following sequence of commands will change the first 15 occurrences of .CSECT in the buffer to .ASECT.

```
* BI
  B G#.CSECT# -4J -C#A#<CR>
    <LF>
* BSK15EM
```

5.2.4.4 Edit Open

The EO command will close the secondary input file and re-open it at the beginning. Therefore, although Edit-11 is a one-pass editor, that is, capable of making only one trip through the primary input file per job, it is possible to make many passes through the subsidiary input file with the EO command. EO has no effect on the text.

5.2.4.5 End File

The EF command closes the primary output file to any further output, and renames it if a backup file is to be created. It also closes the primary input file to any further read operations. The secondary output and secondary input files remain open for further editing. Note that this command is useful to create a truncated output file from a large input file.

5.2.4.6 EXit

The EX command is the most common way to terminate an editing session. Exit copies the remainder of the primary input file into the primary output file, closes all files, and begins another edit.

If, however, an EF command was executed during the editing session, EX will not perform any input/output operations - it merely closes files. The last command of every editing session should be the EX command. Whether it involves any I/O or not depends on whether the primary I/O files are open or closed.

CHAPTER 6

OPERATING INSTRUCTIONS

6.1 STARTING

To start Edit-11, load the PDP-11 Disk Operating System (if it is not running already) and log in. Then type:

```
$RU(N) EDIT11
```

Upon receipt of this command, the DOS will load Edit-11 and begin its execution. The first thing Edit-11 needs is input/output specifications for the editing session. After being loaded and started, Edit-11 responds with #, indicating a request for a command string. The user responds with a command string of the form:

```
#dev:file1.ext[uic],dev:file2.ext[uic]<dev:file3.ext[uic]/B,dev:file4.ext[uic]
```

In the above specification, FILE1 is the primary output and FILE3 is the primary input, while FILE2 and FILE4 are the secondary output and input respectively. Note the following:

1. A minimum of one output must be specified, and a maximum of 2 input and 2 output may be specified. Hence, file2, file3, and file4 above are optional.

2. dev: is the device name and is optional; disk is assumed if none is specified. The assumed device changes, however, as soon as a new device is typed. See Disk Operating System Monitor Programmer's Handbook.

3. Filen.ext is the file name of the appropriate input or output file.

4. [uic] is the user code for the owner of the file, and need not be specified if the file in question is your file. Note that [(SHIFT/K) and] (SHIFT/M) are part of the uic specification and must be typed if a uic is specified.

5. /B is the backup switch. If /B is specified, no backup file will be created. Otherwise, the backup facility will be used (see section 4.1). Note that if the /B switch is used, it must follow the primary input file specification.

6. If file1 and file3 are the same file on one device, file3 will be renamed file3.BAK unless the /B switch is used. file2 and file4 cannot be the same file.

When Edit-11 receives a syntactically correct specification, it performs internal initialization, then types an *, indicating its readiness to receive the first command.

6.2 RESTARTING

To restart Edit-11, use the Monitor REstart command after issuing a CTRL/C.

```
*CTRL/C
.RE
*
```

Note that the above method of restarting enables you to stop a command execution if desired.

6.3 FINISHING AN EDIT

To finish an editing session, use the EX command. This will restart Edit-11 for the next job unless the keyboard (KB:) has been specified as a primary input device. In such a case, the EX must be followed by typing CTRL/C, EN, the RETURN key, and the LINE FEED key.

NOTE

Any other way of leaving an editing session will result in the input and output files not being properly closed. A file not properly closed cannot be used by any system program until correctly closed by the (Unlock) option in PIP. Therefore, if an edit is terminated without an EF or EX command (the plug for the computer was pulled out, the Monitor crashed, or you executed another Monitor command after a CTRL/C), the input and output files must be restored by PIP before attempting to edit again.

6.4 ERROR RECOVERY

In the course of editing a page of the program, it may become necessary to correct mistakes in the commands themselves. There are two special commands which do this:

1. Typing the RUBOUT key removes the preceding typed character, if it is on the current line.

2. The CTRL/U combination (holding down the CTRL key and typing U) removes all the characters in the current line.

If, in the course of a command string execution, a serious error is discovered, the restart capability can be used to terminate the command. (See section 6.2.)

6.5 PROCEDURE WITH LOW-SPEED PUNCH

If the low-speed punch is one of the output devices, Edit-11 pauses before executing any command to write on the punch. The punch must be turned on at this time, after which typing the LINE FEED key on the keyboard initiates the output (the key typed does not echo on the tape). Following completion of the operation, Edit-11 pauses again to allow you to turn the punch off. When the punch has been turned off, type the LINE FEED key and Edit-11 will return to command mode.

CHAPTER 7

IMPLEMENTATION NOTES

7.1 MACRO USAGE

Use of the EM macro is most efficient if the text involved is small enough to fit within the macro itself. Large amounts of text can be inserted by a macro, however, if use is made of the fact that the macro itself is only the first line of the Save Buffer.

For example, suppose it is desirable to insert

```
JSR     R5,RSAVE
JSR     R5,RELOAD
```

after every occurrence of EMT 4Ø in your program. The text to be inserted is too long to enter in your command mode, and there is no way to enter text mode from a macro. Use can be made, however, of the Unsave command.

To accomplish the above, the commands would look like:

```
*BI
G;EMT 4Ø; A U -3A K
     JSR    R5,RSAVE
     JSR    R5,RELOAD
<LF>
*B3S3K 5ØØØEM
```

The first command inserts the 3 lines we need to save into the Page Buffer. Note that the first of the 3 lines is the macro, while the last two are the common code. The macro itself contains the Get command to look for the EMT 4Ø, followed by an Advance command to advance Dot to the next line. The next command is the Unsave command, which will Unsave the three lines in the Save Buffer. Since the macro is by definition the first line of the Save Buffer, it is Unsaved along with the other two and has to be deleted. The -3A and Kill commands accomplish this. The second command string saves the macro and common text, deletes them from the Page Buffer, and executes the macro 5ØØØ times. Five thousand is an arbitrarily large number which assures that we get all the occurrences of EMT 4Ø in the buffer. When the search fails, the macro will be halted and an error returned as Edit-11 prepares to accept another user command string.

7.2 DELIMITER USAGE

When entering text in command mode, any ASCII character is acceptable as a delimiter. Some, however, are better than others, and here are some suggestions when using delimiters.

1. Use the same delimiter all the time (except when not possible because it appears in the text itself). If you pick an uncommonly used character that is easy to type, such as Q or ;, typing the delimiter will become second nature.

2. Avoid delimiters that are valid arguments or commands. Use of /, @ and valid commands as delimiters is an error-prone practice. Avoid spaces most of all, as forgetting that there is a space in the text string will result in the execution of undesired commands, and this is a common error.

   Example:

   ```
   *G SAVE /DUMMY/ I$TEXT$
   ```

   Although the user wanted to get "SAVE /DUMMY/", he will actually Get "SAVE", Delete to the end of the buffer, Unsave, Mark, Mark again, then receive an error message for the illegal Y command.

7.3 SUBSIDIARY I/O

Subsidiary I/O can serve several purposes. Here are some suggestions on how it might be used.

1. If your secondary output file is a line printer or scope, you can look at the entire Page Buffer very quickly. After typing in a long Insert, look at it via the B/EW command and check for mistakes.

2. Use of two output files is ideal for dividing a single input file into two smaller files.

3. If you want to move a very long section of text, or you want to save text for insertion later but also want the Save Buffer free, use the HSR and HSP as your subsidiary I/O. Punch the desired text, then read it in wherever desired.

4. Two input files are ideal for concatenation also; wHole search for a nonexistent object through the primary input, then do the same through secondary input.

Remember that to close subsidiary files, the EX command must be used.

7.4 SAVE AND UNSAVE

Edit-11 saves text during the Save command by setting aside a buffer large enough to accommodate the saved text. This decreases the total core available to the Page Buffer, resulting in the possibility that there might not be enough free core left to Unsave the text. Edit-11 guards against this situation by allowing you to save only text that is short enough to guarantee at least enough room to insert it again. That is, you are guaranteed there will be enough room to Unsave at least once following every Save command that completes successfully. Of course, if you read or Insert a large amount of text between Save and Unsave, you might decrease the space available too much and not be able to Unsave. In this case, part of the buffer will have to be written into the output file to make room. You will be safe if you do all your Unsaves immediately following the corresponding Save commands.

7.5 CREATING A NEW FILE

If the purpose of an editing session is to create a file for which there is no input file, a primary input device need not be specified. Inserts are used to create the file and EX to close it. However, if the keyboard is specified as a primary input device, the file should be closed by the following sequence:

1. Write the file
2. Issue the EF command
3. If the editing session is to be terminated, type the EX command

If you should accidentally issue a read command such as R, N, EX, you can terminate the read with a form feed character. If you type EX without a previous EF and the keyboard is your input device, you must type CTRL/C, wait for a . to be printed out, and then type EN, the RETURN key, and the LINE FEED key in order to proceed successfully.

7.6 .USING EXIT

Always end the editing session with the EX command. EX will guarantee that all files are closed correctly. Recall that if you do not want EX to perform any input file copying or buffer writing, precede it with the EF command.

APPENDIX A

ERROR DIAGNOSTICS

Error Code	Meaning	Cause
S202	Device Full	Output device does not have sufficient room to continue.
S203	Switch Error	Too many switches or illegal switch. A switch appeared which was not equal to /B or which followed a file other than primary input; or more than one switch appeared.
S204	Too Many Output Files	More than two output files were specified or a switch appeared after secondary output.
S205	Too Many Input Files	More than two input files were specified or a switch followed the secondary input.
S227	Illegal File Specification	Argument indicates specific violation. 1 - No primary output specified 2 - Secondary input equals secondary output 3 - Secondary input equals primary output 4 - Primary input equals secondary output 5 - Primary input equals secondary input 6 - Primary output equals secondary output

Command Syntax Errors - Command syntax errors are reported by printing the command up to and including the character at which the scan terminated, followed by a question mark and vertical tab. This does not mean the last character typed is the cause of the syntax error.

W303	Buffer Overflow	Command Input Buffer, Text Input Buffer, Save Buffer or Page Buffer overflow.
W304	Macro Overflow	Macro as stored in Save Buffer is too long to execute.
W305	Recursive Macro	Macro contains an EM command.
W306	Empty Save Buffer	An EM or U command was issued with nothing in the Save Buffer.
W307	Search Failure	n^{th} occurrence of search object was not found in available text.

Error Code	Meaning	Cause
W310	No Room to Unsave	Not enough available room to unsave required text.
W311	End of Data	End of input medium or end of input file reached during read. Last page read was last in file.
W312	Illegal Line Feed	A line feed character was encountered in command string.
W313	Illegal Negative Argument	The command specified does not accept negative arguments.
W314	No Arguments Allowed	The command specified does not recognize any arguments.
W315	Illegal Argument	Command does not accept given argument.
W316	Illegal Text String	Usually caused by missing second delimiter.
W317	Illegal Command	Edit-11 cannot execute command as requested. Usually caused by secondary I/O commands when no secondary I/O was specified at initialization time.
W320	Page Buffer Almost Full	Page Buffer within 128 characters of being full.
W321	File Closed	Attempt to Read or Write primary files after EF.

APPENDIX B

COMMANDS

In the following table, # represents any legal text delimiter. <CR> represents a return character, and <LF> represents a line feed character.

Command	Format	Result
Read	R	Read from primary input file until form feed encountered.
Edit Read	ER	Read from secondary input file until form feed encountered.
Write	nW	Write n lines into primary output file.
Edit Write	nEW	Write n lines into secondary output file.
Form feed	F	Write form feed into primary output file.
Trailer	nT	Write 4Ø null characters as trailer on the primary output device if device is paper tape.
Next	nN	Write the contents of the Page Buffer onto the primary output file, kill the buffer, and read a page of text from the primary input file. Repeat n times. Equivalent to B/W /D R.
Beginning	B	Move Dot to the beginning of the Page Buffer.
Advance	nA	Advance Dot n lines. Leaves Dot at beginning of line.
Jump	nJ	Move Dot over n characters.
Delete	nD	Delete n characters from text.
Kill	nK	Kill n lines of text.

Command	Format	Result
Mark	M	Mark the current location of Dot.
Save	nS	Save the next n lines in the Save Buffer.
Unsave	U	Copy the contents of the Save Buffer into Page Buffer at Dot.
List	nL	List n lines on teleprinter
Verify	V	Verify the present line via teleprinter.
Get	nG#XXXXX# or nG<CR> XXXX<CR> <LF>	Search for the n^{th} occurrence of XXXXX. Return with Dot following XXXXX.
wHole	nH#XXXXX# or nH<CR> XXXXX<CR> <LF>	Search for the n^{th} occurrence of XXXXX. If found, return with Dot following XXXXX. If not found, execute an N command and continue search.
Edit wHole	nEH#XXXXX# or nEH<CR> XXXX <LF>	Perform a wHole search for the n^{th} occurrence of XXXXX, using the secondary input and primary output files.
Position	nP#XXXXX# or nP<CR> XXXX <LF>	Perform a Next command, then search for the n^{th} occurrence of XXXXX. If found, return with Dot following XXXX. If not found, clear the buffer, read another page, and continue search.
Edit Position	nEP#XXXXX# or nEP<CR> XXXX <LF>	Perform a Position search using secondary input rather than primary input file.
Insert	I#XXXXX# or I<CR> XXXXX<CR> <LF>	Insert the text XXXXX at Dot. Move Dot to follow XXXXX.
Change	nC#XXXXX# <LF> or nC<CR> XXXXX<CR> <LF>	Change n characters to XXXXX. Equivalent to Insert followed by n Delete
eXchange	nX#XXXXX# or nX<CR> XXXXX<CR> <LF>	eXchange n lines for XXXXX. Equivalent to Insert followed by n Kill.

Command	Format	Result
Execute Macro	nEM	Execute the first line of the Save Buffer as a command string n times.
Exit	EX	Perform consecutive Next commands until EOM or EOF reached. Close all files, and return to Monitor.
Edit Open	EO	Move to the beginning of the secondary input file. Must follow a W311 error message before an ER can be executed.
End File	EF	Close the primary output file to any further output and close the primary input file.

INDEX

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 & PDP-12
Digital Software News for the PDP-11
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance 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 provided in the software kit 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.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

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, Bldg. 3-5
Maynard, Massachusetts 01754

PDP-11 Edit-11 Text Editor
Programmer's Manual
DEC-11-EEDA-D
May 1971

READER'S COMMENTS

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? If so, specify by page.

How can this manual be improved?

Other comments?

Please state your position. _____ Date: _____

Name: _____ Organization: _____

Street: _____ Department: _____

City: _____ State: _____ Zip or Country _____

— — — — — — — — — — — — — — — — — — Fold Here — — — — — — — — — — — — — — — — — —

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

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754