

PERKIN-ELMER

OS/32 EDIT

User Guide

48-008 F01 R02

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

The software described in this document is furnished under a license, and it can be used or copied only in a manner permitted by that license. Any copy of the described software must include the Perkin-Elmer copyright notice. Title to and ownership of the described software and any copies thereof shall remain in The Perkin-Elmer Corporation.

The Perkin-Elmer Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Perkin-Elmer.

The Perkin-Elmer Corporation, Data Systems Group, 2 Crescent Place, Oceanport, New Jersey 07757

© 1980, 1983, 1984 by The Perkin-Elmer Corporation

Printed in the United States of America

TABLE OF CONTENTS

PREFACE

v |

CHAPTERS

1	OPERATING SYSTEM/32 (OS/32) EDIT	
1.1	INTRODUCTION	1-1
1.2	EDIT REQUIREMENTS	1-1
1.3	SYSTEM ENVIRONMENTS	1-1
1.3.1	Interactive Environment	1-1
1.3.1.1	Prompts	1-2
1.3.1.2	Control Characters	1-2
1.3.2	Batch Environment	1-3
1.4	FILES	1-3
1.4.1	File Descriptors	1-3
1.5	EDIT FEATURES	1-4
1.5.1	ASCII and Hexadecimal Editing	1-4
1.5.2	Identifying Data	1-4
1.5.2.1	Line Numbers	1-5
1.5.2.2	Character Strings	1-5
1.5.2.3	Current Line	1-6
1.5.2.4	Delimiters	1-6
1.5.2.5	Ranges	1-6
1.5.2.6	Carriage Return	1-8
1.5.3	Formatting Data	1-8
1.5.3.1	Line Length	1-8
1.5.3.2	Text Display	1-8
1.5.3.3	Tab Settings	1-9
1.5.3.4	Column Display	1-9
1.6	VERIFICATION	1-9
1.7	COMMANDS	1-9
1.7.1	Edit Command Syntax	1-9
1.7.2	Statement Syntax Conventions	1-10
1.8	MESSAGES	1-11

CHAPTERS (Continued)

2	BUILDING, LOADING, AND STARTING THE EDITOR	
2.1	BUILDING EDIT AS A SYSTEM IMAGE LOAD MODULE	2-1
2.2	LOADING EDIT FROM AN MTM TERMINAL	2-1
2.3	LOADING EDIT FROM THE SYSTEM CONSOLE	2-2
2.4	STARTING EDIT	2-3
3	OS/32 EDIT COMMANDS	
3.1	INTRODUCTION	3-1
3.2	AFTER COMMAND	3-2
3.3	AGAIN COMMAND	3-4
3.4	ALTER COMMAND	3-5
3.5	APPEND COMMAND	3-10
3.6	BEFORE COMMAND	3-11
3.7	BOTTOM COMMAND	3-13
3.8	CHANGE COMMAND	3-14
3.9	COLUMN COMMAND	3-16
3.10	DELETE COMMAND	3-18
3.11	DONE COMMAND	3-20
3.12	END COMMAND	3-21
3.13	FIND COMMAND	3-22
3.14	GET COMMAND	3-24
3.15	HELP COMMAND	3-26
3.16	INCLUDE COMMAND	3-27
3.17	INSERT COMMAND	3-30
3.18	LINE NUMBER COMMAND	3-32
3.19	MOVE COMMAND	3-33

CHAPTERS (Continued)

3.20	NOFIND COMMAND	3-35
3.21	NOTRUNCATE COMMAND	3-37
3.22	OPTION COMMAND	3-38
3.23	PAUSE COMMAND	3-44
3.24	REPLACE COMMAND	3-45
3.25	RULER COMMAND	3-47
3.26	SAVE COMMAND	3-48
3.27	SCREEN COMMAND	3-51
3.28	SEND STOP COMMAND	3-52
3.29	SUBSTITUTE COMMAND	3-53
3.30	TOP COMMAND	3-55
3.31	TRUNCATE COMMAND	3-56
3.32	TYPE COMMAND	3-57
3.33	PLUS SIGN COMMAND	3-59
3.34	MINUS SIGN COMMAND	3-60
4	SAMPLE EDIT SESSION	4-1

APPENDIXES

A	COMMAND SUMMARY	A-1
B	MESSAGES	B-1
C	ASCII-EBCDIC CONVERSION TABLES	C-1
D	BIDIRECTIONAL INPUT/OUTPUT CONTROL (BIOC) FUNCTIONS APPLICABLE TO OS/32 EDIT	D-1

TABLES

1-1	EDIT CONTROL CHARACTERS	1-2
-----	-------------------------	-----

TABLES (Continued)

B-1	SVC1 ERROR CODES	B-1
B-2	SVC7 ERROR CODES	B-2
D-1	LINE DISPLAY COMBINATIONS	D-3
	INDEX	Ind-1

PREFACE

This manual describes the OS/32 Edit Utility, its commands, error messages, and the basic editor concepts and is especially written for the new user.

Chapter 1 introduces Edit and outlines the system requirements. Chapter 2 describes building, loading, and starting the editor from the system console and a multi-terminal monitor (MTM) terminal. Chapter 3 describes all the Edit commands and gives examples of their use. Chapter 4 guides a user through a sample editing session.

Appendix A summarizes Edit commands. Appendix B is a message summary. Appendix C contains ASCII-EBCDIC conversion tables. Appendix D contains bidirectional input/output control (BIOC) function control codes applicable to users of the OS/32 Edit program.

Revision F01 R02 of this manual contains new parameters (CONTINUE/EXIT and WINDOW/NOWINDOW) for the OPTION command.

This manual is intended for use with the OS/32 R07.2 software release and higher.

For information on the contents of all Perkin-Elmer 32-bit manuals, see the 32-Bit Systems User Documentation Summary.

CHAPTER 1 OPERATING SYSTEM/32 (OS/32) EDIT

1.1 INTRODUCTION

OS/32 Edit is a disc-based editor that operates on an OS/32 system and on an OS/32 system with MTM. Edit can be executed in an interactive or batch environment and editing is performed on a line-by-line basis. Data can be appended to a file, altered, and saved via the commands described in Chapter 3.

1.2 EDIT REQUIREMENTS

These system resources are required for Edit:

- The minimum hardware required for OS/32
- Disc devices
- Temporary file support

The pure segment of the editor requires 32.50kb of memory. Each impure segment requires a minimum of 9.25kb of memory. This minimum amount includes 4kb for a buffer in which the editor maintains tables.

1.3 SYSTEM ENVIRONMENTS

Edit can be executed in an interactive or batch environment which is determined by the user-specified command input device in the START command. See Section 2.4.

1.3.1 Interactive Environment

In an interactive environment, a dialogue is carried on between the user and the system. Edit waits for your command, processes it, then prompts you for the next command. Commands are entered interactively through such devices as a CRT or teletype. There are two modes of operation in an interactive environment:

- Command mode
- Input mode

During command mode, editing is performed on a copy of your file residing in memory. You can change, alter, delete, and locate data that has been previously contained in the file.

During input mode, you can change existing data and enter new data in a file. Entering a null line (a carriage return being the only new information on an input line) transfers you from input mode to command mode.

1.3.1.1 Prompts

In command mode, a greater than sign (>) is displayed to the list device. It is a command prompt informing the user to enter a command:

>

When a command is typed in, followed by a carriage return, Edit interprets the command and performs the operation. Some commands are repeated every time the carriage return is depressed until a new command is entered.

Input mode causes a line prompt to be displayed. The prompt character is preceded by a line number:

1 >
2 >

In input mode, Edit will continue to display line prompts until a null line is entered. Edit will then return to command mode.

1.3.1.2 Control Characters

Some interactive devices allow you to modify data before it is transmitted to the buffer via control characters. Table 1-1 lists the control characters and their functions.

TABLE 1-1 EDIT CONTROL CHARACTERS

CHARACTER	FUNCTION	DEVICE
CTRL X	Voids line being typed	CRT, TTY, MODEL 1100, 1200 VDU, Carousel
<--	Deletes character directly preceding symbol	CRT, TTY, Carousel
CTRL H	Moves cursor back 1 space, deleting character in that space	CRT, MODEL 1100, 1200 VDU, Carousel

1.3.2 Batch Environment

In a batch environment, a sequence of Edit commands is input from a disc file, card reader, or magnetic tape device and is processed as a background task. The list device is specified in the START command. User commands, Edit responses and messages are output to the list device.

If an error occurs during batch processing, execution of the current job is halted and this message is displayed:

```
-END OF TASK CODE = 2
```

Normal completion is indicated by the following message:

```
-END OF TASK CODE = 0
```

1.4 FILES

Data is retained by saving it to a file residing on a disk device. Files are identified by a unique name and specified by file descriptors according to the standard OS/32 format.

1.4.1 File Descriptors

File descriptors, abbreviated as fd, are entered in a standard format.

Format:

```
[ voln: ] [filename] [.[ext]] [/ file class]
[ dev: ]
```

Parameters:

voln: is a 1- to 4-character alphanumeric string specifying the name of a disc volume. The first character must be alphabetic and the remaining, alphanumeric. If the volume name is omitted, the default is the system or user volume.

dev: is a 1- to 4-character alphanumeric string specifying a device name. The first character must be alphabetic and the remaining, alphanumeric.

filename is a 1- to 8-character alphanumeric string specifying the name of a file. The first character must be alphabetic and the remaining, alphanumeric. If a filename is specified when a device name is specified, the filename is ignored.

.ext is a 1- to 3-character alphanumeric string specifying the extension to a filename.

file class is a 1-character alphabetic string specifying the type of file class. The file class types are:

P for private file

G for group file

S for system file

If the file class is omitted, the default is P in an MTM environment, and S in an operating system environment.

1.5 EDIT FEATURES

| This section describes editing modes and the methods used to
| identify lines of data during an editing session.

1.5.1 ASCII and Hexadecimal Editing

Edit provides two modes of editing:

- ASCII mode
- Hexadecimal (hex) mode

In ASCII mode, Edit interprets character strings as they are represented. In hexadecimal mode, Edit interprets character strings as pairs of hexadecimal digits. All editor commands can be used in hexadecimal as well as ASCII mode. Choice of modes is specified in the OPTION command. ASCII-EBCDIC conversion tables are in Appendix C.

1.5.2 Identifying Data

| Lines of data can be identified by specifying line numbers or
| character strings; by assuming the current line; by using
| delimiters; or by specifying ranges of line numbers.

1.5.2.1 Line Numbers

Line numbers provide a convenient way to identify lines of data during an editing session. Edit assigns line numbers beginning with 1, incremented by one, up to line number 99999. The APPEND command assigns line numbers in increments of one, and is used to enter new data. The INSERT, REPLACE, and INCLUDE commands insert new information into existing data at a specified position. Line numbers for inserted lines have a decimal part following the integer number.

The decimal part is generated by incrementing the existing line number by .01. The line number of each inserted line is .01 greater than the previous line number. Incrementing stops if a new line number is the same as an existing line number. Thus, several lines can exist with the same line number. In this situation, if a line number parameter is used to specify a line, the first line with the duplicate number following the current line is assumed. Other lines can be located by using the plus (+) or minus (-) commands to position the desired lines. Enter a SAVE command to renumber the entire file.

1.5.2.2 Character Strings

A line of data can be identified by specifying a sequence of unique characters within the line. When a string is specified in a command format, the string identifies a line to the editor. When the specified string is not unique to one line being edited, the first line containing the string is assumed to be the desired line.

Format:

/string/[col]

Parameters:

/	is a non-alphanumeric character delimiting a string. See section 1.5.2.4.
string	is a unique, alphanumeric sequence of characters.
col	is a decimal number from 1 to 256 specifying the starting column location of a character string.

1.5.2.3 Current Line

During editing, the editor maintains a line pointer positioned at the current line. Edit commands with no range parameter specified operate on the current line. When changing from input mode to command mode, the current line is the last line entered from the command device; when changing from command mode to input mode, the current line is the line following the last line edited.

1.5.2.4 Delimiters

A delimiter is a non-alphanumeric character that separates character strings in an input line. Delimiters allow the editor to recognize the beginning and end of a string. Any non-alphanumeric character is valid except:

- semicolon
- comma
- minus sign
- blank
- a character that appears in the string itself

The slash (/) is used to represent a delimiter in this manual.

1.5.2.5 Ranges

A range refers to specific line numbers in a file, or unique character strings that identify specific line numbers in a file. A minus sign is used to separate range arguments.

Format:

$$\left[\left\{ \text{line no}_1 \right\} \right] \left[- \left[\left\{ \text{line no}_n \right\} \right] \right]$$

$\left[\left\{ \text{/string}_1 / [\text{col}] \right\} \right] \left[- \left[\left\{ \text{/string}_2 / [\text{col}] \right\} \right] \right]$

Parameters:

- | | |
|----------------------|---|
| line no ₁ | is the line or beginning line of a range to be operated on. |
| line no _n | is the ending line of the range to be operated on. |

/	is a non-alphanumeric character delimiting a string.
string ₁	is a unique, alphanumeric character string or beginning string of a range to be operated on.
string ₂	is a unique, alphanumeric character string or ending string of a range to be operated on.
col	is a decimal number from 1 to 256 indicating the column locations of string ₁ and string ₂ .

Functional Details:

Any combination of range arguments is permitted. Edit searches the file for the specified lines or strings. After a match is made, the actual function of the command is performed.

Data is operated on beginning with the first line or string specified, and continuing through the last line or string specified. If the second argument is omitted, all lines from the line or string specified to the end-of-file are operated on. If the first argument is omitted, all lines from the current line to the line or string specified are operated on.

If the entire range parameter is omitted, the default is the current line.

Examples:

1-16	indicates lines 1 through 16 inclusive.
range parameter omitted	specifies current line only. An exception to this is the SAVE command where, if the range is null, all lines of data are still saved.
1	indicates line 1 only.
-10	specifies current line through line 10 inclusive.
-	specifies current line through end-of-file inclusive.
1-	indicates line 1 through end-of-file.

/CHARS/-5

is the line where the character string CHARS appears to line 5.

/CHARS/-/DELN/

are the lines identified by the first occurrence of CHARS to the first occurrence of DELN.

1.5.2.6 Carriage Return

The carriage return (CR) transmits a line of data to be processed and switches an Edit user from input mode to command mode.

1.5.3 Formatting Data

Edit presets options that control the data format of a file. These defaults allow the user to create a file without having to specify line length, line numbers, etc. See Section 3.22.

1.5.3.1 Line Length

The maximum number of characters allowed in each line is called line length. When appending to create a new file; i.e., no GET command is issued prior to the APPEND and nottruncate is in effect, the record length defaults to 80 bytes. If a GET is issued to an existing disc file and OPTION LENGTH (see OPTION command) was not specified, the length defaults to the record length of the file. The maximum characters per line is 256. You can specify line length via the length parameter in the OPTION command.

If the number of bytes per record is less than the editor line length, each line is padded with blanks. You can specify a column number, via the TRUNCATE command, indicating the column at which truncation will occur. Characters displayed beyond the specified column are lost when a line of data is expanded. The default is column 73. The TRUNCATE command remains in effect until a NOTRUNCATE command is entered.

1.5.3.2 Text Display

Several commands cause data to be displayed with the appropriate line number. The decimal portion of the integer is only output if it is not equal to zero. Trailing blanks are not output. Data is continued on the next line, up to 80 characters, when necessary.

When the tab option is in effect, data is displayed according to column settings defined by the tab option. See Section 1.5.3.3. When the mode specified in the OPTION command is hexadecimal, all data is displayed in hexadecimal format.

1.5.3.3 Tab Settings

Tab control causes data to be aligned at preset positions and frees the user from manually spacing data.

The tab parameter of the OPTION command allows the user to set tabs and designate a tab control character. The tab settings are used only when text is entered at the command device. The editor expands the line entered according to the tab control character and settings, and retains the line in its expanded form. See Section 3.22.

1.5.3.4 Column Display

Edit displays a ruler at the user's console via the RULER command. Use this as a guide to determine where columns begin and end.

Example:

```
>RULER
```

```
      0           1           2           3           4           5           6
1234567890123456789012345678901234567890123456789012345678901234567
```

When hexadecimal is entered as the mode in the OPTION command, column numbers are output in hex. For example:

```
>RULER
```

```
0102030405060708090A0B0C0D0E0F0101112131415161718191A1B1C1D1E1F202122
```

1.6 VERIFICATION

Lines that are located, changed, or altered are automatically displayed to the list device. Editing verification can be turned off via the noverify option in the OPTION command.

1.7 COMMANDS

Commands are entered following the command prompt and are transmitted for processing by entering a CR.

1.7.1 Edit Command Syntax

Multiple commands can be entered on one line if they are separated by semicolons (;). An exception to this is the AGAIN command which must appear by itself on a command line. When multiple commands are entered on the same line, they are executed sequentially. If an error occurs, any subsequent commands on the line are ignored.

In a batch environment parameters can be continued by entering a comma as the last character and continuing the parameters on the following line.

1.7.2 Statement Syntax Conventions

These statement syntax conventions are used in all command and instruction formats:

Capital letters, parentheses, and punctuation marks must be entered exactly as shown

Lowercase letters represent parameters or information provided by the user
n

Underlining indicates only the underlined portion of the entry is required
PAUSE

Ellipsis represents an indefinite number of parameters or a range of parameters
...
param₁, ..., param₅

Lettering with shading represents a default option
n

Braces represent required parameters from which one must be chosen
{ }

Brackets represent an optional parameter that can be chosen
[]

Commas

separate parameters and substitute missing positional parameters

Braces inside brackets

[} }

represent optional parameters from which one can be chosen

Comma preceding braces inside brackets

[, } }

must be entered if one of the optional parameters is chosen. However, if the parameters are not positional and the first parameter of the statement is not chosen, the parameter specified as the first is not preceded by a comma.

Comma inside brackets

[,]

must be entered if the optional parameter is chosen

Comma outside brackets except last parameter

[], [], [], [],

must be entered in place of missing positional parameters and to separate optional parameters that are chosen. Commas are omitted for trailing parameters and a comma must be entered with the last specified parameter.

Equal sign separating keyword from parameters

must be entered to associate parameter with keyword

KEYWORD=param

1.8 MESSAGES

Four types of messages are output to the command device during an Edit session:

- Messages pertaining to incorrect syntax
- Messages pertaining to incomplete execution of a command
- Messages pertaining to incomplete command execution because of a more serious error
- Information messages

Messages indicating incorrect syntax are preceded by a question mark (?) and are generated when the editor cannot recognize a command or its parameters. The editor does not attempt to execute the specified command.

Execution messages are preceded by an exclamation point (!) and are output during execution. Determine where in the file execution stopped, correct the error, and re-execute the command.

Messages preceded by two exclamation points (!!) denote a serious problem. In some cases, the only way to recover is to attempt to save the current data to a file different from the source file, end the session, and restart.

Information messages are output to help you in an editing session. Some examples of these messages are reminders to save text and indications that a function has completed normally.

For a complete list of messages and meanings see Appendix B.

CHAPTER 2 BUILDING, LOADING, AND STARTING THE EDITOR

2.1 BUILDING EDIT AS A SYSTEM IMAGE LOAD MODULE

This sequence of commands builds the shared HELP segment and Edit task as a system image load module:

```
TITLE 'EDIT32 - SHARED SEGMENT'  
ESTABLISH IMAGE,ADDR=C0000  
RESOLVE HELPRO62.SEG,SIZE=(X400,X1000),ADDR=E0000  
INCLUDE EDIT32,EDIT32A  
MAP EDITR062.MAP  
BUILD EDITR062.SEG  
END
```

```
TITLE 'EDIT32 TASK'  
ESTABLISH TASK  
RES EDITR062.SEG,ADDR=C0000,ACC=RE  
RES HELPRO62.SEG,ADDR=E0000,ACC=RE  
INCL EDIT32,DR  
INCL EDIT32,EDIMP  
MAP EDIT32.MAP,ALPHA,XREF,ADDR  
BUILD EDIT32.TSK  
END
```

For an explanation of these commands, see the OS/32 Link Reference Manual.

2.2 LOADING EDIT FROM AN MTM TERMINAL

After building the image load module, EDIT32.TSK, the module can be loaded into memory with this command:

Format:

```
LOAD fd [,segsize increment]
```

Parameters:

fd is the file descriptor of the Edit load module to be loaded into memory.

segsizesize is a decimal number in kb specifying the
increment additional memory to be added to the task's
impure segment.

Functional Details:

When the Edit load module is loaded into memory, the system displays an asterisk (*) prompt, indicating the editor can be started.

In order to successfully load the task, the shared HELP segment must be on the volume under the account number from which the task is to be loaded, or on the default system volume under account number 0.

| Under MTM, the editor should run with task executing prompts
| disabled (command PREVENT PROMPTS).

2.3 LOADING EDIT FROM THE SYSTEM CONSOLE

The LOAD command loads Edit from the system console.

Format:

LOAD taskid [, [fd] [segsizesize increment]]

Parameters:

taskid	specifies the name of the module after it is loaded into the foreground segment of memory as the currently-selected task.
fd	is the file descriptor of the editor image load module to be loaded into memory. If this parameter is omitted, the default is taskid.TSK.
segsizesize increment	is a decimal number in kb specifying the additional memory to be added to the task's impure segment.

Functional Details:

The system sets Edit as the currently-selected task. When the system displays the asterisk prompt, the editor can be started.

2.4 STARTING EDIT

The START command starts Edit.

Format:

```
START [,COMMAND=fd1] [,LIST=fd2]
```

Parameters:

COMMAND= fd₁ specifies the input device from which commands are to be entered. If this parameter is omitted, the default is the console device (CON:). If the specified command input device is interactive and the LIST parameter is omitted, all output data and messages are sent to the command device. If the command input device is batch, the LIST parameter must be specified.

LIST= fd₂ specifies the output device to which output data is sent. If this parameter is omitted and the command device is interactive, the output data and all messages are sent to the command device. If the command device is batch, the LIST parameter must be specified.

Functional Details:

If both parameters in the START command are omitted, the default is the console device (CON:).

Error Messages:

!INVALID COMMAND DEVICE

The file or device specified as the command device of the START command is syntactically incorrect.

!INVALID LIST DEVICE

The file or device specified as the list device is syntactically invalid.

!DUPLICATE START OPTION

A command or list parameter was entered more than once.

!SYNTAX ERROR IN START OPTIONS

The command or list parameter was entered incorrectly.

!LIST OPTION OMITTED - BATCH MODE

A non-interactive device was specified as the command device,
and the list parameter was not specified.

!UNABLE TO ASSIGN LIST DEVICE

The file or device specified as the list device cannot be
assigned.

--END OF TASK CODE=1

The editor cannot proceed without valid START parameters.
Code 1 indicates abnormal job termination.

CHAPTER 3
OS/32 EDIT COMMANDS

3.1 INTRODUCTION

This chapter describes the following OS/32 Edit commands:

- AFTER
- AGAIN
- ALTER
- APPEND
- BEFORE
- BOTTOM
- CHANGE
- COLUMN
- DELETE
- DONE
- END
- FIND
- GET
- HELP
- INCLUDE
- INSERT
- LINE NUMBER
- MOVE
- NOFIND
- NOTRUNCATE
- OPTION
- PAUSE
- REPLACE
- RULER
- SAVE
- SCREEN
- SEND STOP
- SUBSTITUTE
- TOP
- TRUNCATE
- TYPE
- (+) PLUS SIGN
- (-) MINUS SIGN

AFTER

3.2 AFTER COMMAND

The AFTER command inserts a new character string immediately after a user-specified character string in a line of data.

Format:

$$\text{AFTER /string/newstring} \left[\left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 / [\text{col}] \end{array} \right\} \right] \right. \\ \left. \left[- \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 / [\text{col}] \end{array} \right\} \right] \right] \right]$$

Parameters:

- / is a non-alphanumeric character delimiting a string. The blank, minus sign, comma, semicolon, and any character that appears in the string itself cannot be used.
- string is a sequence of unique alphanumeric characters. The new string is inserted after this string.
- col is a decimal number from 1 to 256 indicating the starting column location of string₁ and string₂.
- newstring is a sequence of unique alphanumeric characters to be inserted after the specified character string.
- line no₁ is the line number or first line number of a range of lines where the newstring will be inserted after the specified string.
- line no_n is the ending line number of a range where the newstring will be inserted after the specified string.
- string₁ is a unique character string or first unique character string of a range which determines where in the file the newstring will be inserted after the specified string.

string₂ is the ending unique character string of a range which determines where in the file the newstring will be inserted after the specified string.

Functional Details:

If the inserted string produces a line longer than the established line length, or if the portion of data to the left of the truncate column is expanded, the following message is displayed when any non-blank characters are lost:

!NON-BLANK CHARACTERS LOST

Examples:

>AFTER/DISC/FILE/2

After the character string DISC, insert character string FILE in line 2.

>AFTER/DISC/FILE/2-/TAPE/

After the character string DISC, insert character string FILE from line 2 to the first occurrence of the character string TAPE.

>AFTER/DISC/FILE/2-

After the character string DISC, insert the character string FILE from line 2 to end-of-file.

AGAIN

3.3 AGAIN COMMAND

The AGAIN command re-executes a previously specified command line.

Format:

AGAIN

Functional Details:

Commands that are not re-executed are DELETE, GET, END, and a line of data entered with a line number. When the AGAIN command is entered following these commands, the following message is displayed:

?REPEAT DISALLOWED

The AGAIN command must be the only command on the input line.

Examples:

```
>TY 1-4
  1 This is an example
  2 of how
  3 the AGAIN
  4 command works.
>AG
  1 This is an example
  2 of how
  3 the AGAIN
  4 command works.
```


`%/string/` is the percent sign followed by a slash, a character string, and a slash, that inserts the character string preceding the percent sign.

`x` indicates any character other than a blank, `\`, `@`, and `%`. The character replaces the current character in that position.

Functional Details:

| All local CRTs are sysgened by default with the bidirectional
| input/output control (BIOC) driver. The capability to use BIOC
| editing features depends on certain BIOC-related features that
| are made available to the user at sysgen time. See the OS/32
| System Generation/32 (SYSGEN/32) Reference Manual.

| Operation of the ALTER command is dependent upon the above
| conditions plus the use of the BIOC option. If the BIOC option
| is enabled, the ALTER command displays the specified line and
| places the cursor at the end of the line. At this point, the
| line may be edited using the BIOC line-editing features. See
| Appendix D.

| After the line has been changed to the desired appearance, a
| carriage return (CR) is entered to terminate the ALTER command.
| Edit proceeds to display the next line of the range, if any. If
| this was the last or only line specified, Edit returns to the
| command mode.

| If the terminal is sysgened with other than the BIOC driver, the
| BIOC option should be disabled (OPTION NOBIOC). Then the ALTER
| command works as described in the following paragraphs.

After the ALTER command is entered, Edit responds by printing the specified line as it currently exists. The line number is then printed again. Changes to the original line now can be entered in the corresponding character positions.

The alter line can be terminated at any character position. All remaining characters in the line are retained. If OPTION VERIFY is selected, Edit responds by printing the altered line. The line number is then printed again, giving another opportunity to alter the line. When OPTION NOVERIFY is selected, Edit does not print the altered line or offer another chance to alter the line. In this case, if another change is desired, ALTER must be reentered. A carriage return in the first character position will terminate ALTER.

| As soon as one line of characters is typed on a terminal, the
| line automatically wraps around to the next line. To alter the
| second part of a line, space through the first part of the line
| in order to access the second part.

The ALTER command is rejected in hexadecimal and batch mode.

Examples:

>T1-3

```
1 This text is part of a new document.
2 The first thing we must do is revise it.
3 This is an example of a long line that wraps
  around when it is displayed.
```

>AL 1

```
1 This text is part of a new document.
1 > old
1 This text is part of a old document.
1 > n
1 This text is part of anold document.
1 > %
1 This text is part of an old document.
1 > %/one /
1 This text is one part of an old document.
1 >
```

>AL 2

```
2 The first thing we must do is revise it.
2 > too@@@@@
2 The first thing toodo is revise it.
2 > \
2 The first thing to do is revise it.
2 >
```

>AL 3

```
3 This is an example of a long line that wraps
  around when it is displayed.
3 >
3 > print@@
3 This is an example of a long line that wraps
  around when it is printed.
3 >
```

Assume the terminal is sysgened with the BIOC driver, and OPTION BIOC is enabled. In the following examples, the cursor is indicated by █ and the file consists of only one line.

>AP

```
1 >THIS IS A TEST. █
2 >
```

>AL 1

```
1 >THIS IS A TEST. █
```

| To delete the period, enter backspace or CTRL-H; line now looks
| like this:

| 1 >THIS IS A TEST.

| Now enter:, AND A GOOD ONE. The line looks like this:

| 1 >THIS IS A TEST, AND A GOOD ONE.

| Enter RETURN and Edit will prompt for the next command. The
| current line will not be redisplayed because it is exactly as it
| appears above.

| To change the AND to BUT:

| >AL

| 1 >THIS IS A TEST, AND A GOOD ONE.

| Enter CTRL-W 5 times: once for the period, then 4 times to
| backspace 4 words, leaving the cursor before the A in AND:

| 1 >THIS IS A TEST, AND A GOOD ONE.

| To move the cursor forward to the A, enter CTRL-F or just SPACE
| once to overwrite the existing space. Now type: BUT. The line
| will look like this:

| 1 >THIS IS A TEST, BUT A GOOD ONE.

| You could hit RETURN at this point, leaving the line as you see
| it. However, let's try an insert. Enter CTRL-^, then a space
| followed by NOT. The line now looks like this:

| 1 >THIS IS A TEST, BUT NOT A GOOD ONE.

Notice that the space at the cursor is shifted right with the rest of the line as the characters are inserted. Now, hit RETURN and the cursor will "zoom" to the end of the line, restoring the remainder, before Edit prompts for the next command.

To change BUT NOT back to AND:

>AL 1

1 >THIS IS A TEST, BUT NOT A GOOD ONE.

Enter CTRL-] followed by a comma. The cursor will search backwards to the nearest comma:

1 >THIS IS A TEST, BUT NOT A GOOD ONE.

Now enter CTRL-F twice to position the cursor at the B in BUT. Hit CTRL-_ 4 times to delete BUT and the space which follows, leaving the cursor at the N in NOT:

1 >THIS IS A TEST, NOT A GOOD ONE.

Finally, enter AND overwriting NOT and hit RETURN. Edit will prompt for the next command. Enter:

>TYPE

1 >THIS IS A TEST, AND A GOOD ONE.

APPEND

3.5 APPEND COMMAND

The APPEND command enters data to a file from the command input device.

Format:

APPEND

Functional Details:

Enter APPEND to either create a new file or add data to an existing file.

In an interactive environment, Edit prompts you for data by displaying line numbers. The first line of data entered is assigned line number 1. Subsequent line numbers are incremented by 1. The maximum lines allowed in a file is 99999. If data already exists, the new information is entered on the line following the last line of data. Tab settings are recognized, if set during the current editing session.

In a batch environment, data is entered immediately following the APPEND command. Terminate data with the character string previously specified in the OPTION command. See Section 3.22.

Examples:

```
>AP
  1  >To create
  2  >a new file
  3  >enter APPEND.
  4  >
>TY 1-
  1  To create
  2  a new file
  3  enter APPEND.
```

3.6 BEFORE COMMAND

The BEFORE command inserts a new character string immediately before a user-specified character string in a line of data.

Format:

$$\text{BEFORE /string/newstring} \left[\left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 / [\text{col}] \end{array} \right\} \right] \right. \\ \left. \left[- \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 / [\text{col}] \end{array} \right\} \right] \right] \right]$$

Parameters:

- / is a non-alphanumeric character delimiting a string. The blank, minus sign, comma, semicolon, and any character that appears in the string itself cannot be used.
- string is a sequence of unique alphanumeric characters. The new string is inserted before this string.
- col is a decimal number from 1 to 256 indicating the starting column location of string₁ and string₂.
- newstring is a sequence of unique alphanumeric characters to be inserted before the specified character string.
- line no₁ is the line number or first line number of a range of lines where the newstring will be inserted before the specified string.
- line no_n is the ending line number of a range where the newstring will be inserted before the specified string.
- string₁ is a unique character string or first unique character string of a range that determines where in the file the newstring will be inserted before the specified string.

string₂ is the ending unique character string of a range that determines where in the file the newstring will be inserted before the specified string.

Functional Details:

If the inserted string produces a line longer than the established line length, or the portion of text to the left of the truncate column is expanded, the following message is displayed if any non-blank characters are lost:

!NON-BLANK CHARACTERS LOST

Examples:

>BEFORE /DISC/FILE/2

Before the character string DISC, insert the character string FILE in line 2.

>BEFORE /DISC/FILE/2-/TAPE/

Before the character string DISC, insert the character string FILE from line 2 to the first occurrence of character string TAPE.

>BEFORE /DISC/FILE/2-

Before the character string DISC, insert the character string FILE from line 2 to end-of-file.

3.7 BOTTOM COMMAND

The BOTTOM command changes the current line pointer to the last line of the file.

Format:

BOTTOM

Examples:

```
>TY 1-  
  1  The BOTTOM command  
  2  changes the current line pointer  
  3  to the last line of the file.  
>TY 1  
  1  The BOTTOM command  
>BOTTOM  
  3  to the last line of the file.
```

3.8 CHANGE COMMAND

The CHANGE command replaces a specified character string with new data.

Format:

```
CHANGE /oldstring/ [col] , /newstring/  
[ { (line no  
  { /string/ [col] }  
  ) [ { (line no1  
    { /string1/ [col] }  
  ) - [ { (line non  
    { /string2/ [col] }  
  ) } ] ] ]
```

Parameters:

- / is a non-alphanumeric character delimiting a string. The comma, minus sign, blank, semicolon, and any character in the string itself cannot be used.
- oldstring is a unique character string to be replaced by the newstring.
- col is a decimal number from 1 to 256 indicating the starting column location of the character string to be changed.
- newstring is a unique character string which replaces the oldstring.
- line no₁ is a line number or first line number of a range to be searched for the oldstring.
- line no_n is the ending line number of a range to be searched for the oldstring.
- string₁ is a unique character string or first string of a range that determines where in the file the oldstring will be replaced by the newstring.

string₂ is the ending unique character string of the range that determines where in the file the oldstring will be replaced by the newstring.

col is a decimal number from 1 to 256 indicating the starting column location of string₁ and string₂.

Functional Details:

When CHANGE is used, all occurrences of the oldstring are replaced by the newstring.

If a tab character is contained within a string, it is treated the same as any character in the string.

If the inserted string causes the line length to exceed the default, or the portion of data to the left of the truncate column is expanded, the following message is displayed if non-blank characters are lost:

!NON-BLANK CHARACTERS LOST

Examples:

>CHANGE.READ,..INPUT.,3-

Change all occurrences of READ to INPUT, from line 3 through end-of-file.

>CHANGE?READ?,?INPUT?,1-20

Change all occurrences of READ to INPUT, from line 1 through line 20.

>CHANGE.READ,.. ,1-

Change all occurrences of READ to a space, from line 1 to end-of-file.

COLUMN

3.9 COLUMN COMMAND

The COLUMN command replaces data beginning at a specified column with a new character string.

Format:

$$\text{COLUMN col no, /string/} \left[\begin{array}{l} \left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \\ \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \end{array} \right]$$

Parameters:

- col no is a decimal number from 1 to 256 specifying the beginning column where data is to be replaced.
- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character that appears in the string itself cannot be used.
- string is a sequence of alphanumeric characters, specified by delimiters, that replaces the data beginning at the specified column.
- line no₁ is the line number or first line number of a range of lines where data will be replaced at the specified column.
- line no_n is the ending line number of a range where data will be replaced at the specified column.
- string₁ is a unique character string or the first string of a range that determines where data will be replaced in a file.
- string₂ is the ending unique character string of a range that determines where data will be replaced in a file.

col is a decimal number from 1 to 256 indicating the starting column location of string₁ and string₂.

Functional Details:

If the inserted string produces a line longer than the established line length, or the portion of text to the left of the truncate column is expanded, the following message is displayed if non-blank characters were lost:

!NON-BLANK CHARACTERS LOST

Examples:

>COL 16,/R3/

Change the data starting in column 16 to the character string R3 in the current line.

>COL 10,/LH/,1-20

Change the data starting in column 10 to the character string LH, in lines 1 through 20.

>COL 15,/A2B/,/A1B/-/A5B/

Change the data starting in column 15 to the character string A2B, from the line containing the character string A1B to the line containing the character string A5B.

>COL 5,/DISC/,-/FILE/

Change the data starting in column 5 to the character string DISC, from the current line to the line containing the character string FILE.

>COL 5,/DISC/,-

Change the data starting in column 5 to the character string DISC from the current line to the end-of-file.

DELETE

3.10 DELETE COMMAND

The DELETE command deletes data from a file.

Format:

$$\text{DELETE} \left[\left\{ \text{line no}_1 \right\} \right] \left[- \left[\left\{ \text{line no}_n \right\} \right] \right]$$
$$\left[\left\{ / \text{string}_1 / [\text{col}] \right\} \right] \left[\left\{ / \text{string}_2 / [\text{col}] \right\} \right]$$

Parameters:

line no ₁	is the line number or first line number of a range to be deleted.
line no _n	is the ending line number of a range of data to be deleted.
/	is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character appearing in the string itself cannot be used.
string ₁	is a unique character string or first unique character string of a range that determines where data will be deleted in a file.
string ₂	is the ending unique character string of a range that determines where data will be deleted in a file.
col	is a decimal number from 1 to 256 indicating the column location of string ₁ and string ₂ .

Functional Details:

| The line(s) specified by the operand(s) is(are) deleted from the
| file. If no operands are specified, the current line is deleted.
| After one or more lines are deleted, the new current line will be
| printed if the VERIFY option is enabled.

If all lines of data are deleted, the following message is displayed:

ALL LINES DELETED

The DELETE command cannot be re-executed by entering the AGAIN command.

Examples:

>DELETE 10-20

Delete lines 10 through 20.

>DELETE

Delete the current line.

>DELETE--

Delete all data from the current line to the end-of-file.

>DELETE 20-

Delete line 20 through the end-of-file.

>DELETE /LPR/--/ACC/

Delete all lines, from character string LPR to character string ACC.

The editor starts searching for the specified character string at the current line and continues, if necessary, to the end-of-file. Therefore, the current line must be numerically less than or equal to the line containing the character strings to be deleted. |

>DELETE /LPR/2-10

Delete all lines, from character string LPR beginning in column 2 through line 10.

DONE

3.11 DONE COMMAND

The DONE command saves the current file and ends the editing session.

Format:

DONE

Functional Details:

This command is equivalent to a SAVE* followed by an END command.

END

3.12 END COMMAND

The END command ends the editing session and returns control to the operating system.

Format:

END

Functional Details:

Edit goes to end-of-task with a code of 0 to indicate normal termination.

If data was not saved before the END command is entered, the following command is displayed:

REMINDER - SAVE YOUR CURRENT TEXT

The END command cannot be re-executed by entering the AGAIN command.

FIND

3.13 FIND COMMAND

The FIND command searches for all occurrences of a specified string in a given range of a file and displays them to the list device.

Format:

$$\text{FIND /string/[col]} \left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/[col]} \end{array} \right\} \left(\left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1\text{/[col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2\text{/[col]} \end{array} \right\} \right] \right) \right\} \right]$$

Parameters:

- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character appearing in the string itself cannot be used.
- string is a sequence of alphanumeric characters specified by delimiters, to be searched for.
- col is a decimal number from 1 to 256 indicating the starting column location of the specified string.
- line no₁ is the line or first line number of a range of lines to be searched for all occurrences of the specified character string.
- line no_n is the ending line number of a range to be searched for all occurrences of the specified character string.
- string₁ is a unique character string or first unique character string of a range that determines where in the file to search for the specified character string.
- string₂ is the ending unique character string of a range that determines where in the file to search for the specified character string.

col is a decimal number from 1 to 256 indicating the starting column location of string₁ and string₂.

Functional Details:

If a tab character is contained within a string, it is treated as just another character in that particular string.

Examples:

>FIND/SCH/,1-

Find every occurrence of the character string SCH, beginning in line 1 through end-of-file.

>FIND/SCH/1,5-7

Find every occurrence of the character string SCH, beginning in column 1 from lines 5 through 7.

>FIND /SCH/,/ACC/2-/CCA/2

Find every occurrence of the character string SCH, beginning at the location of string ACC in column 2 and ending at the location of string CCA in column 2.

>FIND /SCH/,-/CCA/2

Find every occurrence of character string SCH, beginning at the current line and ending at the location of string CCA in column 2.

```
-----  
|  GET  |  
-----
```

3.14 GET COMMAND

The GET command brings a file off disc and readies it for editing.

Format:

```
GET fd
```

Parameters:

fd is the file descriptor of the file being accessed.

Functional Details:

When a GET command is entered, a copy of the specified file is brought off disc. It is now opened for editing and data can be added, deleted, inserted, changed, etc. Not until a SAVE command is entered, does the updated version of the file replace the original version. If a GET command is entered before the current data is saved, a reminder to save your current text is displayed. The GET command is not re-executed by entering the AGAIN command.

If the record length of the fd is different from the current line length, the lines of data are expanded or decreased, accordingly, when they are output. When records are expanded, they are padded with blanks for ASCII mode. The pad character in hex mode is X'00'.

If OPTION LENGTH was issued prior to the GET command, and the physical length of the input file is larger than the specified line length, the following message is displayed:

```
FILE RECORD LENGTH=m, CURRENT LINE LENGTH=n
```

The variable m is the record length of the input file that was established when the GET command was issued. The variable n is the line length specified by the user via the OPTION command.

In a batch environment, the program terminates. In an interactive environment, the message is informative only.

When the editor is executed as a foreground task, only files with account number 0 can be accessed.

Example:

```
>GET CH1EDT.008
```

Get file CH1EDT.008 on the default user volume.

HELP

3.15 HELP COMMAND

The HELP command prints out information on how to use Edit.

Format:

HELP [{ (mnemonic) }
 *]

Parameters:

mnemonic is any valid Edit command mnemonic.
 Information will be displayed on how to use
 the command associated with the mnemonic
 specified.

* causes a list of all Edit commands to be
 displayed.

Functional Details:

The HELP file must be on the default system volume under account 0.

If parameters are omitted, Edit prints out information on how to use the HELP command.

Examples:

>HELP CH

 Displays to the user information on the CHANGE command.

>HELP INS

 Displays to the user information on the INSERT command.

3.16 INCLUDE COMMAND

The INCLUDE command copies a line or range of lines of data to a specified place in a file.

Format:

$$\text{INCLUDE } \left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/[col]} \end{array} \right\} \right], [\text{fd}]$$

$$\left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/[col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1\text{/[col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2\text{/[col]} \end{array} \right\} \right]$$

Parameters:

- line no is the line number after which data will be included. If this parameter is omitted, the data will be included immediately following the current line.
- string is a unique character string which determines where in the file data will be included.
- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character that appears in the string itself cannot be used.
- fd is the file descriptor of the file containing the lines to be included. If this parameter is omitted, the current file is the default.
- line no₁ is the line number or first line number of a range of lines to be inserted after the specified line number.
- line no_n is the ending line number number of a range of line numbers to be inserted after the specified line number.

string₁ is a unique character string or first unique character string of a range that determines the lines to be inserted after the specified line number.

string₂ is the ending unique character string of a range that determines the lines to be inserted after the specified line number.

col is a decimal number from 1 to 256 indicating the starting column location of a character string.

Functional Details:

If the specified range is null, and the fd is omitted, the current line is inserted. If the range is null, and fd is specified, the first line of the specified fd is inserted.

The lines are not deleted from their original place in the file. The inserted lines are assigned the line number they are inserted after, incremented by .01. Incrementing stops when the new line number equals the last existing line number. The last line number generated is then used for all subsequent lines inserted.

Examples:

>INCLUDE 2,,20-

Include after line 2, lines 20 through the last line of the current file.

>INCLUDE 2,TRY.EDT,1-

Include after line 2, all lines of a file named TRY.EDT.

>INCLUDE,,25-45

Include after the current line, lines 25-45 of the current file.

>INCLUDE,TEXT 100-200

Include after the current line, lines 100 through 200 of the file named TEXT.

>INCLUDE .END.7,TEXT,1-

Include, following the line containing END in column 7,
all lines of the file named TEXT.

>INCLUDE 2,TEXT,/BEGIN/2-/END/4

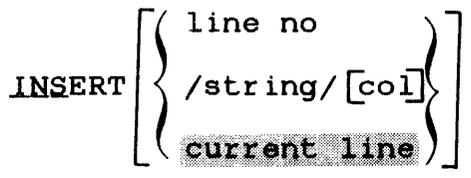
Include, after line 2 from the file named TEXT, all
lines from the line containing the character string
BEGIN in column 2 to the line containing character
string END in column 4.

INSERT

3.17 INSERT COMMAND

The INSERT command inserts one or more new lines of data into a file.

Format:



Parameters:

- line no is the line after which data is to be inserted. If line number is omitted, the data will be inserted after the current line.
- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character in the string itself cannot be used.
- string is a unique character string that determines where in the file data will be inserted.
- col is a decimal number from 1 to 256 indicating the starting column location of the specified character string.

Functional Details:

When the user enters the INSERT command interactively, the editor displays the next fractional line number and a prompt, to indicate that a line of data can be entered. Line numbers are displayed in increments of .01, following the line specified by the line parameter. Incrementing stops if the new line number equals the next existing line number. The last line number generated is then used for all subsequent lines inserted.

In a batch environment, input immediately follows the INSERT command. The data is terminated by a character string specified in the OPTION command.

Examples:

```
>TY 1-  
  1   This is an  
  2   example of  
  3   INSERT command works.  
>ins2  
  2.01>how the  
  2.02>  
>TY 1-  
  1   This is an  
  2   example of  
  2.01 how the  
  3   INSERT command works.
```

| LINE |
NUMBER

3.18 LINE NUMBER COMMAND

| Entering a line number by itself can display, replace, or insert a line of data.

Format:

n [data]

Parameters:

n is a line number.

data is the data to be inserted or replaced.

Functional Details:

| Entering a line number, followed by data, causes that data to be inserted if the line number does not exist. If the line number exists, the new data replaces the original data. If a line number is specified followed by a CR, that line of data is displayed. The command is not re-executed by entering the AGAIN command. In a batch environment, the terminator string specified in the OPTION command replaces the CR.

Examples:

| >2 Display line 2.

| >2 ABCD Replace data on line 2 with ABCD. Line 2 is now the current line.

| >2.01 EFGH Insert line 2.01 after line 2 with the data EFGH appearing on it.

| >0.01 EFGH Inserts a line containing character string EFGH on the line preceding the first line of a file. After the file is saved and renumbered, the line containing character string EFGH becomes the first line of the file.

3.19 MOVE COMMAND

The MOVE command moves a line or range of lines of data within a file.

Format:

$$\text{MOVE } \left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right]$$

$$\left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} - \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/[col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/[col]} \end{array} \right\} \right] \right]$$

Parameters:

- line no is the line number after which lines of data are to be moved.
- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character within the string itself cannot be used.
- string is a unique character string indicating where in the files data is to be moved.
- line no₁ is a line or first line number of a range to be moved after the specified line number.
- line no_n is the ending line number of the range to be moved after the specified line number.
- string₁ is a unique character string or first unique character string of a range that determines the lines to be moved.
- string₂ is a unique character string or ending string of a range that determines the lines to be moved.

col is a decimal number from 1 to 256 indicating the starting column location of a character string.

Functional Details:

After the lines are moved, the original lines of data are deleted. The moved lines are assigned the same line number as the line they are inserted after, incremented by .01.

Examples:

>MOVE 4, 2-3

After line 4, insert lines 2 through 3. Lines 2 through 3 are deleted from their original place in the file.

>MOVE 4, /DISC/-/FILE/

After line 4, insert the lines beginning with the line containing the character string DISC to the line containing the character string FILE. The lines identified by the specified character strings are deleted from their original place in the file.

>TY 1-
1 This is an
2 example of
3 MOVE command works.
4 how the

>MOVE 2,4

>TY 1-
1 This is an
2 example of
2.01 how the
3 MOVE command works.

3.20 NOFIND COMMAND

The NOFIND command searches for all lines in which a specified string does not appear.

Format:

```
NOFIND /string/[col]
      [ { (line no) }
        { (/string/[col]) }
        [ { (line no1) } ] - [ { (line non) } ]
        [ { (/string1/[col]) } ] - [ { (/string2/[col]) } ] ] ]
```

Parameters:

- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character appearing in the string itself cannot be used.
- string is the unique alphanumeric character string to be searched for.
- line no₁ is a line number or first line number of a range to be searched for the specified character string.
- line no_n is the ending line number of a range to be searched for the specified character string.
- string₁ is a unique character string or first unique character string of a range that determines the lines to be searched for the specified character string.
- string₂ is the ending unique character string of the range that determines the lines to be searched for the specified character string.
- col is a decimal number from 1 to 256 indicating the starting column location of a character string.

Functional Details:

If the string occurs in every line specified by the line number parameter, the following message is displayed:

?STRING OCCURS IN ALL LINES

Examples:

>NOFIND /DISC/,1-

Find all lines where character string DISC does not occur, from lines 1 to end-of-file.

>NOFIND /DISC/5, 1-/FILE/

Find all lines where character string DISC, beginning in column 5, does not occur, from line 1 to the line containing character string FILE.

>NOFIND /DISC/,/FILE/3-20

Find all lines where character string DISC does not occur, from the line containing character string FILE, beginning in column 3, through line 20.

NOTRUNCATE

3.21 NOTRUNCATE COMMAND

The NOTRUNCATE command disables truncation if the TRUNCATE command was previously specified.

Format:

NOTRUNCATE

Functional Details:

Truncation is not in effect unless the TRUNCATE command is specified. Truncation is disabled when the NOTRUNCATE command is entered and the following message is displayed:

TRUNCATE NOT IN EFFECT!

Parameters:

COMMAND= fd is a file containing a command or a sequence of commands to be executed when the fd is specified. There can be no more than 5 nested editor command files.

BLOCK= n is a decimal number from 1 to 255 which sets the data block size for allocated files. The default is 5.

MODE= specifying **HEXADECIMAL**, sets the character format to hexadecimal. Specifying **ASCII**, sets the character format to ASCII. The default is ASCII.

LENGTH= n sets the line length for data input and output. This value cannot exceed 256. The default value is 80 bytes or the record length of the file.

LIST fd assigns the list device to the specified fd. The default fd is the fd assigned in the **START** command.

LOG fd assigns the log device to the specified fd. The default is the fd assigned in the **START** command.

NOLOG cancels the log option.

TAB= tab char is a non-alphanumeric character used to set tabs for data formatting. The tab char is followed by a decimal number from 1 to 256 specifying the columns where tabs are to be set.

NOTAB clears current tab settings.

TERMINATOR= is a 1- to 4-character alphabetic string used to terminate data input. Terminator can also equal a carriage return.

SCREEN= n is a decimal number from 1 through 23 that sets the number of lines per screen to be displayed when the **SCREEN** command is entered. The default is 23 lines.

NOVERIFY cancels the verify option. Located, changed or altered lines are not output to the list device.

VERIFY causes data that was located, changed or altered to be output to the list device. If not specified, Verify is in effect.

INPLACE=	ON causes all I/O operations that change a line of text to be performed directly to the disc file. If text currently exists in the edit buffer when this parameter is specified, the ON option is rejected.
	OFF causes all I/O operations that change a line of text to be performed on the text currently in the edit buffer. If neither ON nor OFF is specified, the default is OFF.
BIOC	enables BIOC editing and tabulation features described for the ALTER and OPTION TAB commands. This option is effective only if the terminal has been sysgened with the BIOC terminal driver.
NOBIOC	disables BIOC functions for terminals which do not use the BIOC driver. If neither BIOC nor NOBIOC is specified, the default is NOBIOC.
CONTINUE	causes the editor to continue execution in batch mode when some of the commands in the command file cannot be executed (e.g., RANGE NOT FOUND, STRING NOT FOUND). If this parameter is omitted, the default is EXIT.
EXIT	causes the editor to terminate execution when in batch mode and any one of the commands cannot be executed. If neither CONTINUE nor EXIT are specified, the default value is EXIT. When an unrecoverable error occurs during batch processing (I/O error), execution of the current job is halted regardless of the option setting.
WINDOW	allows the user to specify a range of characters on each line that will be affected by the Edit commands. col ₁ /col ₂ are the start and end column settings that can range from 1 to the maximum line length. The window size cannot exceed 68 characters and col ₁ must be less than col ₂ . col ₁ ,N is the start column setting and N is the size of the window. (e.g., col ₂ =col ₁ +N)
ENFORCE	only the characters within the range specified are affected by the Edit commands.
WARNING	the entire line is affected by the Edit commands and a warning is issued if the characters outside the window are affected.
NOWINDOW	resets the window.

Functional Details:

The OPTION command can be entered any time during an edit session while in command mode. Parameters can be entered on the same line, in any order, separated by commas. The editor processes them sequentially from left to right. If conflicting parameters are entered in the same statement, the last one specified is in effect. The editor displays current option settings if the command is entered with no parameters.

When the command option is specified, if an end-of-file condition occurs when reading the command file, the following message is displayed:

```
!END OF COMMAND FILE
```

There can be no more than 5 nested editor command files.

The block option sets the data block size for all subsequent file allocations. The block size is a decimal number from 1 to 255; the index block size is always 1. The value is used only when a file is allocated. At that time, if the block size specified exceeds the maximum system block size, allocation fails, and the following message is displayed:

```
!!FILE ERR, LUn, SIZE ERROR, fd
```

In the mode option, if ASCII is specified, all characters are interpreted in ASCII. If HEXADECEIMAL is specified, all characters are interpreted as hexadecimal digits. Hex and ASCII modes are recognized for all Edit operations. When the editor is in hex mode, all data must be entered in hexadecimal. Valid hex characters are 0-9 and A-F. Special characters such as the carriage return and tab character have no effects. The pad character in hexadecimal mode is X'00'. The corresponding pad character in ASCII is ' '. The following is an example of hex editing:

```
*EDIT
>O MO=HEX
>AP
1  >01239FFACEDC88ACDCBB789EF55
2  >FFFDDDDDEEEEE88992136553CDACCCEDF
3  >
>CH /DDDD/,/0000/,2
2  FFFF0000EEEE88992136553CDACCCEDF000000000000000000000000
    00000000
```

ASCII-EBCDIC conversion tables are presented in Appendix C.

When the tab option is in effect, data is expanded before it is stored. It remains in expanded form. The tab option entered with no arguments displays the current tab character and settings. Up to 20 tabulation points can be set. Once tabs are set, new tabs can still be inserted before the next highest setting. Column specification can be omitted to avoid changing both column and tab.

When TAB=FORTRAN is specified, the editor sets tabs in columns 7 and 73. Control I is the tab character.

When TAB=CAL is specified, the editor sets tabs in columns 10, 16, 36, and 73. Control I is the tab character.

| If the terminal is sysgened (with full BIOC support) with the BIOC driver and the BIOC functions are enabled (by OPTION BIOC), and if the tab character is Control I, then the tab positions will be "down-loaded" to the driver by the editor. The TAB key then causes the cursor to move to the appropriate position on the screen. The tabs will be expanded by the driver rather than the editor.

NOTE

A tab character will always cause the text to be tabulated to the next tab stop to the right of the current cursor position. For example, if option TAB=CAL is selected and the user tabs to column 10 and types six characters (cursor is now in column 16), the next tab will cause the cursor to jump to column 36.

| When the BIOC option is disabled and text has been inserted past a tab stop, a tab character will cause the text to be tabulated one space instead of to the next tab stop. For example, if TAB=CAL is selected and the user tabs to column 10 and types six characters (cursor is now in column 16), the next tab will cause the cursor to move one space, to column 17.

The terminator option is generally used in batch environment to terminate data input. The terminator characters are positioned in columns 1-4 of the input line and are followed by blanks or a carriage return. In batch environment, when specifying a line number command to delete a line, the alternate terminator can directly follow the line number. TERMINATOR=CR specifies that a carriage return is the only valid terminator.

Any command that would cause the file to be expanded or contracted is disallowed while OPTION INPLACE is in effect. These commands are:

- APPEND
- DELETE
- INCLUDE
- INSERT
- REPLACE
- SAVE

If any of these commands are issued when OPTION INPLACE is in effect, the following message is displayed:

!INPLACE IN EFFECT - COMMAND DISALLOWED

When the INPLACE option is not enabled, any commands that would cause a file to be expanded or contracted are allowed.

OPTION BIOC enables special handling of the ALTER command and TABS. If the terminal has not been sysgened as a BIOC driver, this option has no effect. This option should be disabled when running the editor from the system console.

The WINDOW option is operational only in ASCII mode. However, the mode is always reset to ASCII when the window is set and the window is reset when the mode is set to HEX.

The WINDOW option and the TRUNCATE command cannot be set at the same time.

Certain Edit commands are affected by the WINDOW option and now perform their functions as follows:

SCREEN	displays characters between col ₁ and col ₂ .
TYPE	displays characters between col ₁ and col ₂ . When ENFORCE is set and a string is specified, each line is searched between col ₁ and col ₂ . When WARNING is set, the entire line is searched and if the string is found outside the window, a warning is issued. Window settings are ignored when columns are specified by the TYPE command.

FIND when ENFORCE is set, each line is searched between col₁ and col₂. When WARNING is set, the entire line is searched and if the string is found outside the window, a warning is issued. Window settings are ignored when columns are specified by the FIND command.

CHANGE when ENFORCE is specified, only characters between col₁ and col₂ are affected. When a portion of the line between col₁ and col₂ is expanded or contracted, data outside of the window is not affected. Blanks are inserted to fill in the space when a string is substituted with a shorter string. When a string inside the window is substituted with a longer one, characters moved past col₂ are lost. When WARNING is specified, the entire line is affected and a warning is issued if changes are made outside of the window. Columns specified in the CHANGE command cause the window settings to be ignored.

SUBSTITUTE the same as the CHANGE command but only the first occurrence of the string is replaced.

ALTER allows the user to modify text within the range specified by the WINDOW option. When a portion of the line between column settings is expanded or contracted, data outside of the window is not affected. Blanks are inserted to fill in the space when a string is substituted with a shorter string. When characters are inserted, all characters that are moved past col₂ are lost.

APPEND the text typed in is actually moved to start at col₁ of the window. The beginning of the line is filled with blanks and the number of characters entered should not exceed the size of the window.

**AFTER
BEFORE** when ENFORCE is specified, only a portion of the line between col₁ and col₂ is searched for a user-specified string. When the new string is inserted, the characters moved past col₂ are lost. When WARNING is specified the entire line is searched and a warning is issued if changes are made outside of the window.

RULER the portion of the ruler within the window range is displayed to a list device.

Examples:

>OPTION LENGTH=130

Sets the line length for data entry and output to 130 bytes per line.

>OPTION BLOCK=10

Sets the data block size to 10 for all subsequent file allocations.

>OPTION

Displays all values for current options.

>OPTION LI=PR:

Assigns the printer as the current list device.

>OPTION MODE

Displays the current mode.

>OPTION MODE=H

Sets the current mode to hexadecimal.

>OPTION TAB=\$,5,10

Assigns the \$ as the tab character and the tab stops at columns 5 and 10.

>OPTION TAB

Verifies the tab character and settings.

>OPTION TAB=\$,15,TAB

Adds a third tab setting to those already in effect and displays the tab options.

>OPTION NOTAB

Clears the current tab settings.

>OPTION LOG=fd

Logs all commands and messages to the file or device specified by fd.

>OPTION NOLOG

Cancel the Log option. Commands and messages are not logged.

>OPTION TERMINATOR=END

Defines END, in columns 1-4, as the alternate terminator.

>OPTION VERIFY

Displays the line modified by the CHANGE command to the log device.

>OPTION NOVERIFY

Cancels the Verify option.

>OPTION INPLACE=ON

All changes are made directly to the disc file.

>OPTION INPLACE=OFF

All changes are made directly to the edit buffer.

PAUSE

3.23 PAUSE COMMAND

The PAUSE command temporarily returns control to the operating system.

Format:

PAUSE

Functional Details:

To return to the editor, enter the CONTINUE command. The PAUSE command does not delete or change any data in the file, and the current line is unchanged.

3.24 REPLACE COMMAND

The REPLACE command replaces existing lines of data with new information.

Format:

$$\text{REPLACE} \left[\left\{ \text{line no}_1 \right\} \left[\left\{ \text{/string}_1 / [\text{col}] \right\} \right] \right] \left[- \left[\left\{ \text{line no}_n \right\} \left[\left\{ \text{/string}_2 / [\text{col}] \right\} \right] \right] \right]$$

Parameters:

- line no₁ is the line number or first line number of a range to be replaced.
- line no_n is the ending line number of a range to be replaced.
- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character appearing in the string itself cannot be used.
- string₁ is a unique character string or first unique character string of a range that determines where in the file data is to be replaced.
- string₂ is the ending unique character string that determines where in the file data is to be replaced.
- col is a decimal number from 1 to 255 indicating the column location of string₁ and string₂.

Functional Details:

In an interactive environment, the editor responds to a REPLACE command with a line number as the prompt. The first number is the line specified in the command. Succeeding lines are displayed in increments of .01.

In a batch environment, input immediately follows the REPLACE command. The file is terminated by a specified string in the OPTION command.

Examples:

>REPLACE 4

Allows the user to re-enter new data on line 4.

>TY 1-

1 This is an
2 example of
3 how the
4 MOVE command works.

>REPLACE

4 >REPLACE command works
4.01>

>TY 1-

1 This is an
2 example of
3 how the
4 REPLACE command works.

3.25 RULER COMMAND

The RULER command displays a ruler to the list device.

Format:

RULER

Functional Details:

You can enter the RULER command any time during command mode of an editing session.

The ruler is output in hex, if OPTION MODE=HEX is specified in the OPTION command.

Examples:

```
>RULER
      1      2      3      4      5      6      |
12345678901234567890123456789012345678901234567890123456789012345678 |
>OPTION MODE=H
>RULER
0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F202122
```

NOTE |

The actual line displays 69 characters |
when the prompt is enabled and 70 |
characters when the prompt is disabled. |

SAVE

3.26 SAVE COMMAND

The SAVE command outputs data to a file or device.

Format:

$$\text{SAVE } \left\{ \begin{array}{l} \text{fd} \\ * \end{array} \right\} \left[\left\{ \begin{array}{l} \text{line no} \\ / \text{string} / [\text{col}] \end{array} \right\} \left[\left[\left\{ \begin{array}{l} \text{line no}_1 \\ / \text{string}_1 / [\text{col}] \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ / \text{string}_2 / [\text{col}] \end{array} \right\} \right] \right] \right]$$

Parameters:

- fd is the file descriptor of the allocated file to which the data is to be saved.
- * saves new data to the current input file.
- line no₁ is the line number or first line number of a range to be saved.
- line no_n is the ending line number of a range to be saved.
- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character in the string itself cannot be used.
- string₁ is a unique character string or first unique character string of a range that determines the data to be saved.
- string₂ is the ending unique character string of a range that determines the data to be saved.
- col is a decimal number from 1 to 256 indicating the starting column location of string₁ and string₂.

Functional Details:

If fd specifies a file that does not currently exist, an indexed file is allocated with the specified fd. If fd is the name of a file that currently exists, or is the name of the current input file, Edit displays the following message:

```
DELETE AND REALLOCATE voln:filename/P?
```

If the user indicates that the file should be deleted by entering YES, Edit deletes the file. If the file deleted was indexed, nonbuffered indexed or extendable contiguous, a file with the specified fd and attributes equal to the attributes of the file being edited is allocated. If a contiguous file was deleted, an indexed file with the specified fd is allocated.

Current data is output to the allocated file.

If the user indicates that the file should not be deleted by entering NO, Edit displays the following message:

```
FILE NOT SAVED
```

The editor did not execute the SAVE and outputs a command prompt.

If a previously saved file is saved to an asterisk, or if fd is the name of the file being edited, the current data overwrites the current input file. The editor displays the following message:

```
WORKFILE=voln:filename/P  
RENUMBERED INPUT FILE AVAILABLE,voln:filename/P
```

The data is renumbered and is available for further editing.

When the fd or range parameter is omitted, the entire file is saved. To specify the lines of data to be saved, any combination of range arguments is permitted. Edit searches the file for the specified lines or character strings. After a match is made, the actual function of the command is performed.

Data is saved beginning with the first line or string specified, continuing through the last line or string specified.

To save only the current line, explicitly specify the current line number or a unique string in that line.

Examples:

>SAVE NEW.EDT

Save all the lines to a file named NEW.EDT on the default volume.

>SAVE NEW.EDT,1-10

Save lines 1 through 10 to a file named NEW.EDT on the default volume.

>SAVE M67B:NEW.EDT

Save all the lines to a file named NEW.EDT on volume M67B.

>SAVE*

Save the current file to the same filename.

>SAVE M67B:NEW.EDT,/ACC/2-/CCA/60

Save all data from the line identified by the first occurrence of character string ACC in column 2, to the line identified by character string CCA in column 60 to the file NEW.EDT on volume M67B.

3.27 SCREEN COMMAND

The SCREEN command displays a full screen of data starting, ending, or centered at the current line.

Format:

SCREEN [{ E }
 { C }
 { S }]

Parameters:

- E displays a full screen of data ending at the current line. If this parameter is omitted, the default is S.
- C displays a full screen of data centered at the current line. If this parameter is omitted, the default is S.
- S displays a full screen of data starting at the current line.

Functional Details:

If the SCREEN parameter was specified via the OPTION command, Edit displays only the number of lines specified by n.

```
-----  
|   SEND   |  
|   STOP   |  
-----
```

3.28 SEND STOP COMMAND

The SEND STOP command allows the user to halt execution of the current command.

Format:

SEND STOP

Functional Details:

The command is only valid during interactive use of the editor.

The SEND STOP command is recognized when Edit is executing as a foreground task, is the current task, or is a terminal task under an MTM system.

Enter SEND STOP in response to an operating system prompt. If Edit is currently outputting to the command device, depress the Break key to obtain the prompt. SEND STOP causes Edit to stop processing the command issued prior to SEND STOP. Edit is now ready to accept the next command.

3.29 SUBSTITUTE COMMAND

The SUBSTITUTE command replaces a specified string of data with a new string of data. In a line of data only the first occurrence of the string is replaced using this command.

Format:

$$\text{SUBSTITUTE /oldstring/newstring} \left[\left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 / [\text{col}] \end{array} \right\} \right] \right. \\ \left. \left[- \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 / [\text{col}] \end{array} \right\} \right] \right] \right]$$

Parameters:

- / is a non-alphanumeric character delimiting a string. The comma, minus sign, blank, semicolon, and any character in the string itself cannot be used.
- oldstring is a unique character string to be replaced by the newstring.
- col is a decimal number from 1 to 256 indicating the starting column location of the character string to be changed.
- newstring is a unique character string which replaces the oldstring.
- line no₁ is a line number or first line number of a range to be searched for the oldstring.
- line no_n is the ending line number of a range to be searched for the oldstring.
- string₁ is a unique character string or first string of a range that determines where in the file the oldstring will be replaced by the newstring.
- string₂ is the ending unique character string of the range that determines where in the file the oldstring will be replaced by the newstring.

col is a decimal number from 1 to 256 indicating the starting column location of string₁ and string₂.

Examples:

>SU /SUB/READ

Change the first occurrence of the character string SUB to the character string READ in the current line.

>SU .READ..1-

Change the first occurrence of the character string READ to the null string from line 1 to end-of-file.

>SU /SUB/READ//DISC/--/FILE/

Change the first occurrence of character string SUB to character string READ from the line containing character string DISC to the line containing character string FILE.

>TY 1-

1 A*B=B*A
2 B*A=A*B

>SU /A/C/1-2

1 C*B=B*A
2 B*C=A*B

>AGAIN

1 C*B=B*C
2 B*C=C*B

3.30 TOP COMMAND

The TOP command changes the current line pointer to the first line of the file.

Format:

TOP

Examples:

```
>TY 1-  
1 The TOP command  
2 changes the current line pointer  
3 to the first line of the file.  
>TOP  
1 The TOP command
```

TRUNCATE

3.31 TRUNCATE COMMAND

The TRUNCATE command truncates a line of data at a specified column.

Format:

```
TRUNCATE [col no]
```

Parameters:

col no is the column number after which data is truncated.

Functional Details:

| The specified column number becomes the point at which characters
| are lost when data to the left of this column is expanded (e.g.,
| by substituting a longer string via the SUBSTITUTE command).
| Data to the right of the truncate column does not move when data
| to the left of the column is expanded. If any non-blank
| characters are lost (because they overflowed the truncate
| column), the following message is displayed:

```
!NON-BLANK CHARACTERS LOST
```

When blank characters are truncated, no message is displayed.

| If data to the right of the truncate column is expanded,
| characters are lost off the end of the line, the same as if
| truncate was not in effect.

If the col no parameter is omitted and truncation is in effect, the value of the current column is displayed. If truncation has been disabled with a NOTRUNCATE command, the following message is displayed:

```
TRUNCATE NOT IN EFFECT!
```

3.32 TYPE COMMAND

The TYPE command displays lines of data to the list device.

Format:

$$\text{TYPE} \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 / [\text{col}] \end{array} \right\} \right] \left[- \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 / [\text{col}] \end{array} \right\} \right] \right]$$

Parameters:

- line no₁ is the line or first line number of a range of lines to be displayed to the list device.
- line no_n is the ending line number of the range to be displayed to the list device.
- / is a non-alphanumeric character delimiting a string. The comma, blank, minus sign, semicolon, and any character in the string itself cannot be used.
- string₁ is a unique character string or first unique character string of a range that determines what data is to be displayed.
- string₂ is the ending unique character string of a range that determines what data is to be displayed.
- col is a decimal number from 1 to 256 indicating the starting column of string₁ and string₂.

Functional Details:

When a string parameter is specified, Edit begins searching from the line following the current line.

If the tab option is in effect, information is displayed appropriately.

Examples:

>TYPE 10-20

Display lines 10 through 20 of the current file.

>TYPE 8

Display line 8 of the current file.

>TYPE 20-

Display lines 20 through the last line of the current file.

>TYPE

Display the current line.

>TYPE -

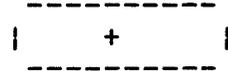
Display all lines from the current line through the last line of the current file.

>TYPE/DISC/-/FILE/

Type all lines from the first occurrence of character string /DISC/ to the first occurrence of character string /FILE/.

>TYPE/DISC/3-/FILE/60

Type all lines from the first occurrence of character string /DISC/ in column 3, to the first occurrence of the character string /FILE/ in column 60.



3.33 PLUS SIGN COMMAND

The plus sign (+) displays an increment of the current line of a file.

Format:

+ [n]

Parameters:

n is the increment of the current line to be displayed. If this parameter is omitted, the default is the current line no+1.

Functional Details:

If the line number parameter is greater than the number of lines in the file, then the last line of data is displayed along with the following message:

LAST LINE

Examples:

>+ 2 Display the second line following the current line.

>+ Display the line following the current line.

-

3.34 MINUS SIGN COMMAND

The minus sign (-) displays a line preceding the current line of a file.

Format:

- [n]

Parameters:

n is the line preceding the current line to be displayed. If this parameter is omitted, the default is the current line number-1.

Functional Details:

If the line number parameter is greater than the number of preceding lines in the text, then the first line is output, along with the following message:

FIRST LINE

Examples:

>- Display the line before the current line.
>-4 Display the fourth line preceding the current line.

CHAPTER 4 SAMPLE EDIT SESSION

The following Edit session was performed in an MTM environment that supports accounting.

```
>AP
1  >The standard includes such tigh-performance features
2  >as capacity for up to 3 of memory, 124 general-
3  >purpose registers, mjltillevel interrupt structure,
4  >memory relocation and protection hardware along with a
5  >8 megabytes/second direct memory access bandwidth, dynamic
6  >comprehensive instruction set.
7  >a host of performance instruction sets.
8  >To complement the 3220's inherent power, Perkin-Elmer offers
9  >
>BEFORE /includes/processor /1
1  The standard processor includes such tigh-performance features
>AFTER /3/ megabytes/2
2  as capacity for up to 3 megabytes of memory, 124 general-
>MOVE 6,8
>TY 5-7
5  8 megabytes/second direct memory access bandwidth, dynamic
6  comprehensive instruction set.
6.01 To complement the 3220's inherent power, Perkin-Elmer offers
7  a host of performance instruction sets.
>SA ch4edt.008
WORK FILE = M300:CH4EDT.000/P
RENUMBERED INPUT FILE AVAILABLE, M300:CH4EDT.008/P
>GET CH4EDT.008
>TY 1-
1  The standard processor includes such tigh-performance features
2  as capacity for up to 3 megabytes of memory, 124 general-
3  purpose registers, multilevel interrupt structure,
4  memory relocation and protection hardware along with a
5  8 megabytes/second direct memory access bandwidth, dynamic
6  comprehensive instruction set.
7  To complement the 3220's inherent power, Perkin-Elmer offers
8  a host of performance instruction sets.

>O TA=/,29
>O TA
THE TAB CHARACTER IS /
29
>AP
9  >Cache Memory/Writable Control Store
10 >Memory Error Logger/Floating Point Processor
11 >Battery Backup System/Data Handling Instructions
12 >
```



```

4      8 megabytes/second direct memory access bandwidth, dynamic
5      memory relocation and protection hardware along with a
6      comprehensive instruction set.
>COL 28,/o/,9-11
9      Cache Memory          oWritable Control Store
10     Memory Error Logger   oFloating Point Processor
11     Battery Backup System oData Handling Instructions
>CH /o/,//,9-11
9      Cache Memory          Writable Control Store
10     Memory Error Logger   Floating Point Processor
11     Battery Backup System  Data Handling Instructions
>SA *
WORK FILE = M300:CH4EDT.000/P
RENUMBERED INPUT FILE AVAILABLE, M300:CH4EDT.008/P
>G ch4edt.008
>TY 1-
1      The standard processor includes such high-performance features
2      as capacity for up to 4 megabytes of memory, 128 general-
3      purpose registers, multilevel interrupt structure,
4      8 megabytes/second direct memory access bandwidth, dynamic
5      memory relocation and protection hardware along with a
6      comprehensive instruction set.
7      To complement the 3220's inherent power, Perkin-Elmer offers
8      a host of performance instruction sets.
9      Cache Memory          Writable Control Store
10     Memory Error Logger   High Performance Floating Point
11     Battery Backup System  High Speed Data Handling Instructions
>END
MAUREEN -END OF TASK CODE= 0   CPUTIME=7.313/6.777

```


APPENDIX A
COMMAND SUMMARY

AFTER /string/newstring $\left[\left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 / [\text{col}] \end{array} \right\} \right] \right]$
 $\left[- \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 / [\text{col}] \end{array} \right\} \right] \right]$

AGAIN

ALTER [line no₁] [-[line no_n]] $\left[\left\{ \begin{array}{l} \backslash \\ @ \\ \% \\ \%/\text{string}/ \\ x \end{array} \right\} \right]$

APPEND

BEFORE /string/newstring $\left[\left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 / [\text{col}] \end{array} \right\} \right] \right]$
 $\left[- \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 / [\text{col}] \end{array} \right\} \right] \right]$

BOTTOM

CHANGE /oldstring/ [col] ,/newstring/

$$\left[\left\{ \left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right\} \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \right]$$

COLUMN col no,/string/

$$\left[\left\{ \left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right\} \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \right]$$

DELETE $\left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] \left[- \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \right]$

DONE

END

FIND /string/ [col]

$$\left[\left\{ \left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right\} \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \right]$$

GET fd

HELP $\left[\left\{ \begin{array}{l} \text{mnemonic} \\ * \end{array} \right\} \right]$

INCLUDE $\left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right], [\text{fd}]$

$\left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right. \\ \left. \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \right\} \right]$

INSERT $\left[\begin{array}{l} \text{line no} \\ \text{/string/ [col]} \\ \text{current line} \end{array} \right]$

MOVE $\left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right]$

$\left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right. \\ \left. \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \right\} \right]$

NOFIND /string/ [col]

$\left[\left\{ \begin{array}{l} \text{line no} \\ \text{/string/ [col]} \end{array} \right\} \right. \\ \left. \left[\left\{ \begin{array}{l} \text{line no}_1 \\ \text{/string}_1 \text{/ [col]} \end{array} \right\} \right] - \left[\left\{ \begin{array}{l} \text{line no}_n \\ \text{/string}_2 \text{/ [col]} \end{array} \right\} \right] \right\} \right]$

NOTRUNCATE

OPTION [COMMAND [=fd]] [,BLOCK [= {n}]] [,MODE [= {HEXADECIMAL}]]
 [,LENGTH [= {n}]] [,LIST [= { fd }]]
 [,LOG [= { fd }]] [,NOLOG]
 [,TAB [= { tab char, col₁ [, col₂, ..., col_n] }]] [,NOTAB]
 [,TERMINATOR [= { string }]] [,SCREEN [= {n}]]
 [,VERIFY] [,INPLACE [= {ON}]] [,BIOC]
 [,EXIT] [,NOBIOC]
 [,CONTINUE] [,WINDOW [= { col₁/col₂ }]] [,ENFORCE]
 [,EXIT] [,NOWINDOW] [,WARNING]]

PAUSE

REPLACE [{ line no₁ }] [- [{ line no_n }]]
 [/string₁/[col]] [/string₂/[col]]

RULER

SAVE { fd } [{ line no }] [/string/[col]]
 * [{ line no₁ }] [- [{ line no_n }]]
 [/string₁/[col]] [/string₂/[col]]]

SCREEN $\left\{ \begin{array}{c} E \\ C \\ S \end{array} \right\}$

SEND STOP

SUBSTITUTE /oldstring/newstring $\left[\left[\left\{ \begin{array}{c} \text{line no}_1 \\ /string_1/[col] \end{array} \right\} \right] \right]$
 $\left[- \left[\left\{ \begin{array}{c} \text{line no}_n \\ /string_2/[col] \end{array} \right\} \right] \right]$

TQP

TRUNCATE [col no]

TYPE $\left[\left\{ \begin{array}{c} \text{line no}_1 \\ /string_1/[col] \end{array} \right\} \right] \left[- \left[\left\{ \begin{array}{c} \text{line no}_n \\ /string_2/[col] \end{array} \right\} \right] \right]$

+ [n]

- [n]

n [data]

APPENDIX B
MESSAGES

!!FILE ERR,LUN,SIZE ERROR,fd

The specified block size exceeds the system maximum data block size.

!!I/O ERR, LUn, code, fd

n is the logical unit (lu) number.

code is the error code. See Table B-1.

fd is the file descriptor.

TABLE B-1 SVCL ERROR CODES

ERROR CODE	MEANING
X'81'	ILLEGAL OR UNASSIGNED LU
X'82'	PARITY OR RECOVERABLE ERROR
X'84'	UNRECOVERABLE ERROR
X'88'	END OF FILE
X'90'	END OF MEDIUM
X'A0'	DEVICE UNAVAILABLE
X'CO'	ILLEGAL FUNCTION

!!FILE ERR, LUn, code, fd

n is the lu number.

code is the error code. See Table B-2.

fd is the file descriptor.

TABLE B-2 SVC7 ERROR CODES

ERROR CODE	MEANING
01	ILLEGAL FUNCTION
02	LU ERROR
03	VOLUME ERROR
04	NAME ERROR
05	SIZE ERROR
06	PROTECT ERROR
07	PRIVILEGE ERROR
08	BUFFER ERROR
09	ASSIGNMENT ERROR
0A	TYPE ERROR
0B	FILE DESCRIPTOR ERROR
0C	TGD ASSIGNMENT ERROR
0D	ACCOUNT ERROR

ALL LINES DELETED

All lines were deleted from a file using the DELETE command.

?COL # WITHIN STRING OPERAND INVALID HERE

A column was specified where it is not allowed in the command format.

?COLUMN INCORRECT

A column was specified where it is not allowed in the command format.

?COMMA MISSING

A comma was missing in the command syntax.

?COMMAND NOT RECOGNIZED

The command mnemonic was entered incorrectly.

!COMMAND LEVELS NESTED TOO DEEP

Command level exceeds that supported by program.

DELETE AND REALLOCATE VOLN:FILENAME /P?

The file specified in the SAVE command already exists or is the name of a current input file. Enter YES if the file is to be deleted and reallocated. Enter NO if the file is not to be deleted and reallocated.

!END OF COMMAND FILE

An EOF condition occurred while reading a command file.

END OF TASK CODE=0

Normal completion of a job.

END OF TASK CODE=1

Abnormal job termination as a result of invalid START arguments.

END OF TASK CODE=2

Execution was terminated as a result of an error encountered in batch processing.

?FILE NOT FOUND

The file was not found on the specified volume or the default volume.

FILE NOT SAVED

NO was entered in response to the DELETE AND REALLOCATE message.

?FILENAME INCORRECT

The filename was entered incorrectly.

!FIRST LINE

The line number specified exceeds top-of-file.

HELP FILE FORMAT ERROR - ABORTED

The Help file format is incorrect.

HELP FILE NOT FOUND

The Help file was not found in the system volume under the system account.

!INVALID BLOCK SIZE

The specified block size is not in the range allowed.

?INVALID COLUMN SPECIFIER

The specified column number is invalid.

?INVALID HEX CHARACTER

The specified character string contains an invalid hex character.

!INVALID LIST DEVICE

The fd specified in the list command cannot be assigned. The original assignment is in effect.

!INVALID SEND MESSAGE

An invalid parameter was entered during a send sequence.

!INVALID TRUNCATE VALUE

An invalid value was specified for the truncate column.

!INVALID OPERAND

The parameter specified in the SCREEN command is not one of 'S', 'E', or 'C'.

!FILE RECORD LENGTH=m CURRENT LINE LENGTH=n

The record length of the file specified when a GET command was issued, exceeds the length specified in the OPTION LENGTH command.

!LAST LINE

The number specified results in going to end-of-file.

!LENGTH OPTION NOT ACCEPTED

The length option was specified after the GET command.

!LINE NOT FOUND

The line or string after which data is to be inserted could not be found.

?LINE OPERAND INVALID

The specified line number is invalid.

!NON-BLANK CHARACTERS LOST

The replacement character string exceeds the default number of characters per line. Non-blank characters are truncated.

!NO TEXT

An Edit command was entered before a file was brought off disc via the GET command, or there was no data in the allocated file.

?NUMBER INVALID

The line parameter is not a numeric character.

?OCCURS IN ALL LINES

When executing the NOFIND command, the specified character string to be searched for was found in all lines.

!OPTION ERROR FOLLOWS XXXX

The characters appearing in place of xxxx are the last four non-blank characters recognized as valid. An OPTION parameter was not valid.

!RANGE 1 NOT FOUND

The lower limit of the range could not be found.

!RANGE 2 NOT FOUND

The upper limit of the range could not be found.

?RANGE INVALID

The specified range was entered incorrectly.

!REMINDER - SAVE YOUR CURRENT TEXT

An END or another GET command was entered before entering a SAVE of current data.

?REPEAT DISALLOWED

The DELETE, END, GET commands, and a line of data entered with a line number are not re-executed by entering a carriage return.

!SCREEN SIZE INVALID

The number of lines specified for screen size is invalid.

?STRING INVALID

The specified character string was entered incorrectly.

!STRING NOT FOUND

Character string not found within the specified range.

!TAB ILLEGAL

An illegal tab setting was specified, or an illegal tab character was used.

!TABS TABLE FULL

There are more than 20 tab settings.

TEXT IN BUFFER, CANNOT SWITCH TO INPLACE

Text currently exists in the buffer. ON option rejected.

!TRUNCATE NOT IN EFFECT

The NOTRUNCATE command disabled truncate.

!TRUNCATE VALUE INVALID

There is an invalid value specified for the truncate column.

?UNABLE TO LIST FULL SCREEN

There is insufficient data in the file to list a full screen as requested in the SCREEN command.

APPENDIX C
ASCII - EBCDIC CONVERSION TABLES

CHARACTER	CARD PUNCHES	ASCII	EBCDIC
		HEXADECIMAL	
Uppercase Letters			
A	12-1	41	C1
B	12-2	42	C2
C	12-3	43	C3
D	12-4	44	C4
E	12-5	45	C5
F	12-6	46	C6
G	12-7	47	C7
H	12-8	48	C8
I	12-9	49	C9
J	11-1	4A	D1
K	11-2	4B	D2
L	11-3	4C	D3
M	11-4	4D	D4
N	11-5	4E	D5
O	11-6	4F	D6
P	11-7	50	D7
Q	11-8	51	D8
R	11-9	52	D9
S	0-2	53	E2
T	0-3	54	E3
U	0-4	55	E4
V	0-5	56	E5
W	0-6	57	E6
X	0-7	58	E7
Y	0-8	59	E8
Z	0-9	5A	E9
Lowercase Letters			
a	12-0-1	61	81
b	12-0-2	62	82
c	12-0-3	63	83
d	12-0-4	64	84
e	12-0-5	65	85
f	12-0-6	66	86
g	12-0-7	67	87
h	12-0-8	68	88
i	12-0-9	69	89
j	12-11-1	6A	91
k	12-11-2	6B	92
l	12-11-3	6C	93

CHARACTER	CARD PUNCHES	ASCII	EBCDIC	
		HEXADECIMAL		
Lowercase Letters				
m	12-11-4	6D	94	
n	12-11-5	6E	95	
o	12-11-6	6F	96	
p	12-11-7	70	97	
q	12-11-8	71	98	
r	12-11-9	72	99	
s	11-0-2	73	A2	
t	11-0-3	74	A3	
u	11-0-4	75	A4	
v	11-0-5	76	A5	
w	11-0-6	77	A6	
x	11-0-7	78	A7	
y	11-0-8	79	A8	
z	11-0-9	7A	A9	
Numerals				
0	0	30	F0	
1	1	31	F1	
2	2	32	F2	
3	3	33	F3	
4	4	34	F4	
5	5	35	F5	
6	6	36	F6	
7	7	37	F7	
8	8	38	F8	
9	9	39	F9	
Symbols				
Exclamation point	!	11-8-2	5D	5A
Quotation mark, dieresis	"	8-7	22	7F
Number sign, pound sign	#	8-3	23	7B
Dollar sign	\$	11-8-3	24	5B
Percent sign	%	0-8-4	25	6C
Ampersand	&	12	26	50
Apostrophe, acute accent	'	8-5	27	7D
Opening parenthesis	(12-8-5	28	4D
Closing parenthesis)	11-8-5	29	5D
Asterisk	*	11-8-4	2A	5C
Plus sign	+	12-8-6	2B	4E
Comma, cedilla	,	0-8-3	2C	6B
Minus sign, hyphen	-	11	2D	60
Period, decimal point	.	12-8-3	2E	4B
Slash, virgule, solidus	/	0-1	2F	61
Colon	:	8-2	3A	7A
Semicolon	;	11-8-6	3B	5E

CHARACTER	CARD PUNCHES	ASCII	EBCDIC
		HEXADECIMAL	
Symbols			
Less than	< 12-8-4	3C	4C
Equal sign	= 8-6	3D	7E
Greater than	> 0-8-6	3E	6E
Question mark	? 0-8-7	3F	6F
Commercial at symbol	@ 8-4	40	7C
Opening bracket	[12-8-2	5B	4A
Closing bracket] 12-8-7	21	4F
Reverse slash	\ 0-8-2	5C	E0
Circumflex	^ 11-8-7	5E	5F
Underline	_ 0-8-5	5F	6D
Grave accent	` 8-1	60	79
Opening brace	{ 12-0	7B	C0
Closing brace	} 11-0	7D	D0
Vertical line	12-11	7C	6A
Overline, tilde	~ 11-0-1	7E	A1
Nonprintable Characters			
ACK (acknowledge)	0-9-8-6	06	2E
BEL (bell)	0-9-8-7	07	2F
BS (backspace)	11-9-6	08	16
CAN (cancel)	11-9-8	18	18
CR (carriage return)	12-9-8-5	0D	0D
DC1 (device control 1)	11-9-1	11	11
DC2 (device control 2)	11-9-2	12	12
DC3 (device control 3)	11-9-3	13	13
DC4 (device control 4)	9-8-4	14	3C
DEL (delete)	12-9-7	7F	07
DLE (data link escape)	12-11-9-8-1	10	10
EM (end of medium)	11-9-8-1	19	19
ENQ (enquiry)	0-9-8-5	05	2D
EOT (end of transmission)	9-7	04	37
ESC (escape)	0-9-7	1B	27
ETB (end of transmission block)	0-9-6	17	26
ETX (end of text)	12-9-3	03	03
FF (form feed)	12-9-8-4	0C	0C
FS (file separator)	11-9-8-4	1C	1C
GS (group separator)	11-9-8-5	1D	1D
HT (horizontal tabulation)	12-9-5	09	05
LF (line feed)	0-9-5	0A	25
NAK (negative acknowledge)	9-8-5	15	3D
NUL (null)	12-0-9-8-1	00	00
RS (record separator)	11-9-8-6	1E	1E
SI (shift in)	12-9-8-7	0F	0F
SO (shift out)	12-9-8-6	0E	0E
SOH (start of heading)	12-9-1	01	01
SP (space)		20	40
STX (start of text)	12-9-2	02	02

CHARACTER	CARD PUNCHES	ASCII	EBCDIC
		HEXADECIMAL	
Nonprintable Characters			
SUB (substitute)	9-8-7	1A	3F
SYN (synchronous idle)	9-2	16	32
US (unit separator)	11-9-8-7	1F	1F
VT (vertical tabulation)	12-9-8-3	0B	0B

APPENDIX D
BIDIRECTIONAL INPUT/OUTPUT CONTROL (BIOC)
FUNCTIONS APPLICABLE TO OS/32 EDIT

Bidirectional input/output control (BIOC) is a standard OS/32 terminal driver. This appendix contains information about BIOC function control codes applicable to users of the OS/32 EDIT program. Listed are function control codes, standard control characters generated by the use of the codes, and the functions performed.

On terminals that do not generate standard control characters for any of the function keys, it is necessary to determine which key will produce the required control characters in order to invoke a desired function.

When a combination of control and ASCII keys cannot be accepted, BIOC rejects that combination and responds with a bell code. An example of this is the use of CTRL-H if attempted at the first character position in a line.

ASCII control characters for BIOC are not echoed (displayed to the console) to prevent confusion between BIOC functions and terminal functions.

For a complete list of BIOC functions available, see the Multi-Terminal Monitor (MTM) Reference Manual.

CTRL-B (STX) Backspace (Nondestructive)

This code causes the cursor to backspace one character each time the code is used. To be effective, CTRL-B cannot be entered at the first character position on a line. When the cursor has been backspaced to the desired character position, the line may be changed by typing the desired characters. All other characters backspaced over can be restored and the cursor brought back to the end of the line in one of two ways:

- CTRL-F moves the cursor forward one character at a time.
- CTRL-Z "zooms" the cursor immediately to the end of the line.

CTRL-C (ETX) Capture the Last Line Entered

Entering this code causes the last line entered (maximum of 80 characters) to be displayed on the console. By using CTRL-C repeatedly, character strings can be concatenated. If an insert or delete function is performed, the CTRL-C code is rejected and a bell sounds to remind you that the buffer has now been overwritten.

CTRL-F (ACK) Forward Space and Restore

This code is used to restore a line that has been backspaced over by the CTRL-B, CTRL-W, or CTRL-] code. After the cursor has been moved to the desired position and the correction has been made, CTRL-F moves the cursor forward one character position at a time until it reaches the end of the line. CTRL-F is rejected if there are no characters to be restored.

CTRL-H (BS) Backspace (Destructive)

This code is used to delete a character or characters. Unlike CTRL-B, however, any character(s) backspaced over by using the CTRL-H code cannot be restored by using the CTRL-F or CTRL-Z codes and must be retyped. If they are not retyped, blank spaces appear in those character positions. CTRL-H is rejected if attempted at the first character position in a line. On most terminals, the CTRL-H code can be generated by the "backspace" key.

CTRL-L (FF) Set Page Pause Line Count

To set the CRT screen display for a specific number of lines, the CTRL-L code is entered, followed by depressing the control key again with another ASCII character. The numeric value of the ASCII character sets the number of lines to be displayed. To select a count for a 24-line CRT, enter the sequence: CTRL-L, CTRL-X (CTRL-X has a decimal value of 24).

The following table shows the proper combinations for line displays ranging from 1 to 24.

TABLE D-1 LINE DISPLAY
COMBINATIONS

SEQUENCE	NUMBER OF LINES
CTRL-L CTRL-A	1
CTRL-L CTRL-B	2
CTRL-L CTRL-C	3
CTRL-L CTRL-D	4
CTRL-L CTRL-E	5
CTRL-L CTRL-F	6
CTRL-L CTRL-G	7
CTRL-L CTRL-H	8
CTRL-L CTRL-I	9
CTRL-L CTRL-J	10
CTRL-L CTRL-K	11
CTRL-L CTRL-L	12
CTRL-L CTRL-M	13
CTRL-L CTRL-N	14
CTRL-L CTRL-O	15
CTRL-L CTRL-P	16
CTRL-L CTRL-Q	17
CTRL-L CTRL-R	18
CTRL-L CTRL-S	19
CTRL-L CTRL-T	20
CTRL-L CTRL-U	21
CTRL-L CTRL-V	22
CTRL-L CTRL-W	23
CTRL-L CTRL-X	24

Each display of the requested number of lines is terminated with a bell sound. At this point the user may continue to the next page by entering a carriage return (CR). This causes the same number of lines to appear; each CR, in fact, produces that number of lines until the page pause line count is changed. To change the count, terminate write by entering ESC or Break, and enter a different sequence for the desired new line count (e.g., CTRL-L CTRL-O = 15 lines, etc.).

To cancel the page pause mode, use the sequence CTRL-L CTRL-@. If the page pause mode is not terminated within 5 minutes, BIOC automatically continues output to prevent the terminal from being permanently tied up.

CTRL-M (CR) Terminate Read

This function is a carriage return (CR). Entering CTRL-M indicates to BIOC that read should be terminated. If CTRL-M is entered at a location other than the end of the line, BIOC performs a zoom to the end of the line (EOL) before storing the CR and terminating the read request.

CTRL-R (DC 2) Reprint Entered Line

When this code is entered, the current cursor location within the line determines the number of characters that are reprinted on the next line. All characters to the left of the cursor, including blank spaces, are reprinted.

The CTRL-R function is especially useful for hardcopy terminals where corrections are made over the existing typed lines. To view a "clean" line after all corrections have been made, CTRL-R is used.

CTRL-W (ETB) Word Backspace (Nondestructive)

CTRL-W causes the cursor to be backspaced (nondestructively) to the nearest nonalphabetic character. Thus, CTRL-W allows the cursor to backspace over one complete word, rather than one character, as with CTRL-B. Words backspaced over may be restored by the use of CTRL-F or CTRL-Z. CTRL-W is rejected if attempted at the beginning of a line.

CTRL-X (CAN) Cancel Current Input Line

All characters previously entered on the current line are deleted upon use of the code. Characters may not be restored with the CTRL-F or CTRL-Z functions. If no characters are on the line, CTRL-X is rejected.

CTRL-Z (SUB) "Zoom" to Furthest End of Line

CTRL-Z can be used to restore a line that has been backspaced over by CTRL-B, CTRL-W, or CTRL-]. CTRL-Z causes the cursor to "zoom" to the end of the line, but is rejected if there are no characters to be restored.

CTRL-] (GS) Backward Character Search (Nondestructive)

This code serves to locate a specific character on the current line. For example, to find the character \$, enter CTRL-]\$. BIOC backspaces until the first \$ is found. To find any additional dollar signs on the same line, the code must be entered again for each time the \$ symbol appears. Characters backspaced over may be restored by using CTRL-F or CTRL-Z. CTRL-] will be rejected if attempted at the beginning of the line.

CTRL-^ (RS) Toggle Between Insert-on and Insert-off

Each CTRL-^ toggles from insert-on to insert-off or from insert-off to insert-on. When the insert mode is selected, characters typed are inserted to the left of the character currently at the cursor position. The insert mode may be selected only when the echo state is in effect and the cursor is positioned at a location other than end of the line. Insert mode is terminated by another CTRL-^ or by any command that moves the cursor to the end of line (e.g., CTRL-Z). The CTRL-C function is not valid while in the insert mode. All other functions are valid if the cursor is not in motion. All data entered while the cursor is in motion is ignored until the cursor has stopped.

CTRL-_ (US) Delete Character

Each CTRL-_ deletes the character currently at the cursor position. The delete code is valid only when the cursor is positioned at a location other than end of line. Characters entered while the cursor is in motion are ignored.

INDEX

AFTER command, 3-2
AGAIN command, 3-4
ALTER command, 3-5
APPEND command, 3-10
ASCII and hexadecimal editing, 1-4
ASCII-EBCDIC conversion tables, C-1
ASCII mode, 1-4

Background task, 1-3
Batch processing, 1-3
BEFORE command, 3-11
BIOC function applicable to Edit, D-1
Block size, 3-39, 3-41
BOTTOM command, 3-13
Break key, 3-52
Building Edit, 2-1

Carriage return, 1-2, 1-8
CHANGE command, 3-14
Character strings, 1-5
COLUMN command, 3-16
Column display, 1-9
Command
 files, 3-39, 3-41
 mode, 1-1, 1-2, 3-41
 prompt, 1-2, 3-49
Commands
 AFTER, 3-2
 AGAIN, 3-4
 ALTER, 3-5
 APPEND, 3-10
 BEFORE, 3-11
 BOTTOM, 3-13
 CHANGE, 3-14
 COLUMN, 3-16
 DELETE, 3-18
 DONE, 3-20
 END, 3-21
 FIND, 3-22
 GET, 3-24
 HELP, 3-26
 INCLUDE, 3-27
 INSERT, 3-30
 LINE NUMBER, 3-32
 MINUS SIGN, 3-60
 MOVE, 3-33
 NOFIND, 3-35
 NOTRUNCATE, 3-37
 OPTION, 3-38
 PAUSE, 3-44
 PLUS SIGN, 3-59
 REPLACE, 3-45
 RULER, 3-47
 SAVE, 3-48
 SCREEN, 3-51
 SEND STOP, 3-52
 SUBSTITUTE, 3-53
 TOP, 3-55
 TRUNCATE, 3-56
 TYPE, 3-57
 Command summary, A-1
 Control characters, 1-2
 Control I, 3-41
 CTRL H, 1-2
 CTRL X, 1-2
 Control options, 3-38
 Current line, 1-6

DELETE command, 3-18
Delimiters, 1-6
Disk devices, 1-1
DONE command, 3-20

Edit command syntax, 1-9
Edit features, 1-4
 BIOC-related, 3-6
Edit load module, 2-2
Edit requirements, 1-1
Edit session, 4-1
END command, 3-21

File
 class, 1-4
 descriptors, 1-3
 group, 1-4
 private, 1-4
 system, 1-4
Filename, 1-4
FIND command, 3-22
Formatting data, 1-8

GET command, 3-24
Group file, 1-4

HELP command, 3-26
Hexadecimal editing, 1-4, 3-41

Hexadecimal, 1-4, 3-38, 3-40
Identifying data, 1-4
INCLUDE command, 3-27
Input mode, 1-1, 1-2
INSERT command, 3-30
Interactive
 command mode, 1-1
 input mode, 1-1
Interactive, 3-5, 3-45

Line
 length, 1-8, 3-3, 3-15, 3-47
 number, 1-4
 per screen, 3-39
LINK NUMBER command, 3-32
Loading Edit, 2-1, 2-2
 from an MTM terminal, 2-1
 from the system console, 2-2

Messages, 1-11, B-1
MINUS SIGN command, 3-60

Mode
 ASCII, 1-4
 command, 1-1, 1-2
 hexadecimal, 1-4
 input, 1-1, 1-2
MOVE command, 3-33

NOFIND command, 3-35
NOTRUNCATE command, 3-37
Null line, 1-2

Operating system prompt, 3-52
OPTION command, 3-38
 BIOC, 3-40, 3-42
 BLOCK, 3-39
 COMMAND, 3-39
 CONTINUE, 3-40
 ENFORCE, 3-40
 EXIT, 3-40
 INPLACE, 3-40
 LENGTH, 3-39
 LIST, 3-39
 LOG, 3-39
 MODE, 3-39
 NOBIOC, 3-40
 NOLOG, 3-39
 NOTAB, 3-39
 NOVERIFY, 3-39
 NOWINDOW, 3-40
 SCREEN, 3-39
 TAB, 3-39

OPTION command (Continued)
 TERMINATOR, 3-39
 VERIFY, 3-18, 3-39
 WARNING, 3-40
 WINDOW, 3-40

PAUSE command, 3-44
PLUS SIGN command, 3-59
Private file, 1-4
Prompts, 1-2
 command, 1-2
 disabled, under MTM, 2-2
 greater than sign, 1-2
 line, 1-2
 operating system, 3-52

Ranges, 1-6
REPLACE command, 3-45
RULER command, 3-47

Sample Edit session, 4-1
SAVE command, 3-48
SCREEN command, 3-51
SEND STOP command, 3-52
Starting Edit, 2-3
Statement syntax conventions, 1-10
SUBSTITUTE command, 3-53, 3-56
System environments, 1-1
System file, 1-4

Tab
 character, 3-39
 settings, 1-9, 3-39
Tabulation points, 3-42
Temporary file support, 1-1
Terminator, 3-39, 3-42
Text display, 1-8
TOP command, 3-55
TRUNCATE command, 3-56
TYPE command, 3-57

Verification, 1-9

WINDOW, use of, 3-40

PUBLICATION COMMENT FORM

We try to make our publications easy to understand and free of errors. Our users are an integral source of information for improving future revisions. Please use this postage paid form to send us comments, corrections, suggestions, ect.

1. Publication number _____

2. Title of publication _____

3. Describe, providing page numbers, any technical errors you found. Attach additional sheet if necessary.

4. Was the publication easy to understand? If not, why?

5. Were illustrations adequate? _____

6. What additions or deletions would you suggest? _____

7. Other comments: _____

From _____ Date _____

Position/Title _____

Company _____

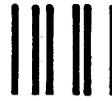
Address _____

STAPLE

STAPLE

FOLD

FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 22 OCEANPORT, N.J.

POSTAGE WILL BE PAID BY ADDRESSEE

PERKIN-ELMER

Data Systems Group
106 Apple Street
Tinton Falls, NJ 07724

**ATTN:
TECHNICAL SYSTEMS PUBLICATIONS DEPT.**

FOLD

FOLD

STAPLE

STAPLE