SESD009

1.0 INTRODUCTION_TO_EDT

EDT is an improved version of the standard NOS 1.3 text editor developed by R. Upton, with thanks to CDD division for ideas and EDT as a starting point.

EDT uses all EDIT commands as well as many new commands and extensions to existing commands such as: column searches, column manipulations, multiple entries on command lines and on the EDT control card.

This manual will describe all of the commands of EDT and all of their various options.

Please bring any errors, omissions or misunderstandings regarding this manual to the author's attention.

2.0 TEXT EDITING CONCEPTS

This section describes the fundamental concepts and terms associated with the Text Editor as a preparation for the discussion of the edit session. Included are such subjects as initiating Text Editor, general command syntax, string manipulation and column manipulation procedures.

2.1 INITIATING TEXT EDITOR

The user initiates the Text Editor with the EDT command or control statement. The format is:

```
EDT,lfn1,lfn2,lfn3,tabs.cmds

  or

EDT,FN=lfn1,I=lfn2,L=lfn3,T=tabs.cmds
```

The first format is order dependent; the second is order independent. The parameters have the following values:

lfn1	The name of the file to be edited. Default is the primary file.
lfn2	The file from which the edit commands are to be read. For a time-sharing session, default is input from the terminal. For a batch job, default is a record in the job deck (INPUT file). If lfn2 = 0 then the edit commands (cmds) are processed and the editor will end. If lfn2 = 1 then the edit commands (cmds) are processed and the editor will obtain the next command from the file INPUT.
lfn3	The file on which the output is to be written. For a time-sharing session, default is the terminal. For a batch job, default is the file OUTPUT.
tabs	This is the tabset to be used in EDT. See the TAB command for the valid mnenomics.

cmds This is a series of EDT commands which will be executed whenever lfn2 is 0 or 1. Otherwise, this is a comment field.

2.1.1 INTERACTIVE USAGE OF EDT

The time-sharing user frequently uses default versions of the EDT command. Thus, the entry

```
EDT
```

calls EDT and performs editing on the primary file with directives entered at the terminal. Output is printed at the terminal using the existing character set mode.

The default entry

```
EDT,lfn
```

calls the editor and performs editing on the local file lfn with directives entered at the terminal. Output is printed at the terminal using the existing character mode. After the EDT command is entered, the system replies:

```
 BEGIN TEXT EDITING.
?
```

This message indicates that the editor program is initiated and awaiting commands. The message will not appear if lfn2 = 0. The program is designed to process only the editor commands discussed in section 3 of this manual. Thus, the regular time-sharing commands are illegal until an exit is made from the editor. It may be necessary to enter and exit the editor several times during an editing session in order to use features not available under EDT control (refer to Terminating Editing Session at the end of section 3).

The text editor may be called from any of the time-sharing subsystems.

2.1.2 BATCH USAGE OF EDT

EDT can be used by a batch job if it includes the EDT control statement in its control statement record. Batch usage of EDT requires that the job deck be properly structured to ensure that the correct records are read from the job INPUT file or that they are available as local files. Refer to the NOS Reference Manual, volume 1 for a complete discussion of batch job structure.

For example, suppose a batch job contains a record listing six types of cable assemblies and the amounts on hand. The job calls on EDT to produce two listings of specific types. The deck is shown in figure 2.1. The cable list is the second record in the INPUT file. This is copied to a local file and given the name PARTS.

```
CABLES.
USER(usernam,passwor,fam)
CHARGE(chargeno,project)
COPYCR(,PARTS)
EDT(PARTS,,TEMP)
COPYSBF(TEMP,)
end-of-record
CABLE,4-WIRE,6-FOOT     ON-HAND 22
CABLE,4-WIRE,8-FOOT     ON-HAND 09
CABLE,6-WIRE,6-FOOT     ON-HAND 03
CABLE,6-WIRE,8-FOOT     ON-HAND 11
CABLE,8-WIRE,6-FOOT     ON-HAND 01
CABLE,8-WIRE,8-FOOT     ON-HAND 19
end-of-record
LIST:/6-FOOT/;*
LIST/8-WIRE/*
END
end-of-information
```

The input file can also appear as follows to get the same results.

```
CABLES.
USER(usernam,passwor)
CHARGE(chargeno,project)
COPYCR(,PARTS)
EDT(PARTS,0,TEMP)LIST:/6-FOOT/;*.LIST/8-WIRE/*
COPYSBF(TEMP,)
end-of-record
CABLE,4-WIRE,6-FOOT     ON-HAND 22
CABLE,4-WIRE,8-FOOT     ON-HAND 09
CABLE,6-WIRE,6-FOOT     ON-HAND 03
CABLE,6-WIRE,8-FOOT     ON-HAND 11
CABLE,8-WIRE,6-FOOT     ON-HAND 01
CABLE,8-WIRE,8-FOOT     ON-HAND 19
end-of-information
```

Printout from execution
of the preceeding job

```
BEGIN TEXT EDITING.

CABLE,4-WIRE,6-FOOT      ON-HAND 22
CABLE,6-WIRE,6=FOOT      ON-HAND 03
CABLE,8-WIRE,6-FOOT      ON-HAND 01
-END OF FILE-
CABLE,8-WIRE,6-FOOT      ON-HAND 01
CABLE,8-WIRE,8-FOOT      ON-HAND 19
-END OF FILE-
END TEXT EDITING.
```

Figure 2-1. Batch job using EDT

The EDT control statement references PARTS which is rewound by EDT. The mode of file processing is the current mode (normal). The missing parameter after the comma, in case 1, indicates the source default of INPUT. Therefore, the editing commands are taken from the next record in the job deck (following the list of six cables). In case two, the editing commands are taken from the EDT control card (after the (or .).

TEMP identifies a temporary file on which the results of editing are written. These results are not routed directly to the printer since, at this point, allowance has not been made for carriage control by the first character of each line.

The temporary file TEMP is copied to the OUTPUT file with a COPYSBF control statement, which moves the text over one column leaving the first position of each line blank. This causes single spacing.

2.1.3 ADDITIONAL CONSIDERATIONS

It is possible to enter EDT with an empty file and develop it during the edit session (refer to Adding and Building Text in section 3)

NOTES

EDT operates on a single record only. If it is entered with a multi-record file, all but the first record is lost (refer to the NOS

Time-Sharing Users Manual or the IAF Reference Manual, section 3, File Sorting).

EDT operates on files containing no more than 131,071 (377,777 octal) lines. Reference to lines beyond this gives unpredictable results.

Some EDT commands are powerful and can ruin a file if improperly used. Therefore, the user should have a copy of the file being edited. To create a copy of a direct access file of local file, refer to COPY control statements, NOS Reference Manual, volume 1. A working file can be saved prior to, or during editing.

2.2 EDIT_FILE

The editor operates on only one edit file at any given time. The edit file can be the primary file, a working file, or a direct access permanent file and is specified when entering EDT with the EDT command. All changes to the edit file are reflected in the original working file or direct access file. The edit file has a line limit of 150 characters. Lines longer than 150 characters are truncated.

NOTE

Editing a read-only file may cause unpredictable results.

2.3 SEARCH_POINTER

The search pointer is a place marker that indicates a particular line of the edit file. Unless command parameters indicate otherwise, the operation implied by the command word is performed on the line indicated by the search pointer. In any case, all action on a file begins relative to the search pointer.

The search pointer is set at the beginning of the edit file when EDT is initiated. The SET, FIND, RESET, and LENGTH commands are used to change its value, and are the only commands capable of doing so.

A command that operates on more than one line of the edit file always begins operation at the line indicated by the search pointer (or relative to that line).

2.4 EDT_COMMANDS_(GENERAL_FORMAT)

Each editing operation on an edit file is specified by an edit command. The following elements are possible in an edit command.

- Command word

  This is the manditory first element. It can be any one of the EDT commands of a short form thereof, as listed in the Commands Words section.

- String specification

  A string consists of a nonzero number of alphanumberic characters bounded on each end by a non-blank character (called a delimiter). In most commands, the string identifies the part of the file being sought, added to, or changed. (In the MERGE, REPLACE, SAVE and LOCAL commands, the string is a file name.) The delimiters on each end of the string must be the save character, must not be the character $ or blank, and cannot be used within the string. The command terminator (by default a .) cannot be used within the second string using the third form of the syntax, as well. (The / is arbitrarily used in the formats to designate the delimiting character.)

  A string specification must immediately follow either a) a colon, b) a comma or c) a delimiter. If two string specifications are included in a command, they may be separated by a comma and a delimiter or start immediately after the delimiter. NOTE in the third form a command terminator may NOT be used in the second string. The forms of a string specification are:

  omitted

  :/string/ or ,/string/ or /string/ (1)

  :/string/,/string/ or ,/string/,/string/ (2)

/string/string/ (3)

- Column specification

  String specifications can also be bounded by the columns. Column specifications will limit the string searches to the specific columns mentioned in the command. When the columns are specified they will override the current default limits set by the user.

  The column specification must have delimiters on each end. A semi-colon is optional. There must be two delimiters between the last string character and the first column descriptor. The forms of the column specification are:

  omitted

  ;/col/ or /col/

- Beginning column searches

  The column specifier, by default, locates the strings in which only the first character resides within the indicated columns. The B option causes EDT to locate a string which completely resides within the indicated columns. This feature has the following forms:

  omitted

  /b or ;b

- Delimited searches

  This option will allow any string searches to look for only strings which have a character that is not alphanumeric on each side of it. The beginning of line and end of line are treated as non-alphanumeric. The delimited string specifier has the following forms:

  omitted

  /d or ;d

- n parameter

  This indicates the number of times the particular edit operation is to be applied. n can be an integer or an asterisk. The integer must be positive for all commands except SET and FIND, which can use positive or negative values. An asterisk caused the operation to be repeated from the current position of the search pointer to the end of the file.

An n parameter specification may follow a semicolon. The forms of this specification are:

omitted

;n or n

;* or *

- Comment

  An optional comment can appear as the last element in an EDT command sequence. It is introduced by a $ and consists of any sequence of characters that can fit on the remainder of the line. Comment has no effect on the operation of the command.

Each command, including a possible comment, must be contained within a single line. Each element must appear in the sequence shown, although not every element need appear. Generally, columns specification cannot occur without a string specification (DC command is the exception). Delimited string specifier MUST have a string specifier to be of any use. Otherwise any element, except the command, may be left out. The square brackets ([,]) indicate the limits of each element. Pressing the carriage return initiates the operation of the command.

```
cmd [:/string1/ [,/string2/] ] [;/col/] [;d [b] ] [;n] [$comm]

    or

cmd [,/string1/ [,/string2/] ] [;/col/] [;d [b] ] [;n] [$comm]

     or

cmd [/string1/ [string2/] ] [/col/] [/d [b] ] [n] [$comment]
```

2.4.1 LINE MODE AND STRING MODE

Some edit commands have two modes of operation, line mode and string mode. In a line mode command, all operations are performed with a line of edit file as the basic unit of operation. The string may be a portion of a line or may extend over several lines.

NOTE
It is important not to confuse string mode with the search string used in both line mode and string mode edit commands. The search string specifies the point or area of the edit file to which the command operation is directed. The string mode refers to the nature of the command operation.

A string mode command with an empty search string specification has the same action as the corresponding line mode command.

2.5 COMMAND_WORD

The command word determines the operation to be performed. The EDT command words are listed with their corresponding short forms (if any) shown in parentheses.

Line_Command_Words		String_Command_Words	
ADD	(A)	ADDS	(AS)
BLANK	(B)	BLANKS	(BS)
CHANGE	(C)	CHANGES	(CS)
DELETE	(D)	DELETES	(DS)
EXTRACT	(E)	ES	
FIND	(F)	FINDS	(FS)
LIST	(L)	INSERTS	(IS)
NUMBER	(N)	LISTS	(LS)
		NUMBERS	(NS)
		RS	

Control_Command_Words

ADDC	(AC)	ECHO	
ASCII		EXd	

ALIGN	(AL)	LISTAB	(LT)
BREAK		LISTC	(LC)
CBd		LOCAL	
CHANGEC	(CC)	MERGE	(M)
CEOL		MERGEL	(ML)
CLEAR	(CL)	NORMAL	
COLUMN	(COL)	RESET	(R)
DEFTAB	(DT)	REPLACE	
DELETEC	(DC)	SAVE	
DEOL		SET	(S)
END		TAB	(T)
LENGTH		TERM	
LINE	(LN)	WIDTH	(W)

2.6 STRINGS AND DELIMITERS

A string is a sequence of alphanumberic characters that may include blanks and special characters. Strings are used in two ways.

- In the string specification of a EDT command
- In response to an ENTER TEXT request

The two ends of the string must be explicitly defined by a pair of matching characters called delimiters. A delimiter is any nonblank character except a dollar sign ($) and is chosen by the user.

In the third version of the string format, there are three additional restrictions in the second string. These are: (1) the first character cannot be a semi-colon, (2) a command terminator cannot be in the string, and (3) a comma cannot be the only character.

The delimiter character can be used within the string only in response to an ENTER TEXT request. If, however, such an embedded character (identical to the delimiter character) appears at the end of a line (for example, the last character entered proir to a carriage return), EDT interpets the character as the closing delimiter and the ENTER TEXT request is terminated. EDT tests for the closing delimiter only after a carriage return.

Use of the delimiter character within the string definition of an EDT command is not allowed and if used, EDT responds:

command SYNTAX ERROR.

(In this manual the character / is used to denote a delimiter in the presentation of command formats.)

Correct String Definition

/ABCDE/	
/THE FORMAT OF/	
BALWAYS IS B	See NOTE for warning of this delimiter
? INT(R*TAN(2*M))?	

Incorrect String Definition

/THIS STATEMENT WILL	(no closing delimiter)
(HOWEVER)	(different delimiter characters)
ANY COMMAND TERMINATED BY/	(unintended beginning delimiter)
$THIS LOOKS LIKE A COMMENT$	(illegal delimiter character)
/FIND/; IN THE PHRASE /	(a semi-colon is first character in second string)
/CHANGE/,/	(comma only character in second string)
/DELETE/THE ./	(period is default command terminator and is in second string)

NOTE

Improper or unintended string definitions are common errors, and because of the powerful nature of some EDT commands, are potentially destructive to a file.

In version 3 of the string format an alphabetic character cannot be a delimiter. Whenever the colon or comma separates the command and the string an alphabetic character can be a delimiter.

2.7 SEARCH_STRING_PARAMETER

The search string parameter of an EDT command indicates to the editor where the operation is to be performed. If no search string is given in a command, the operational location depends solely on the setting of the search pointer. If a search string is given, the operation specified is performed with respect to the first occurrence of the string after the beginning of the line indicated by the search pointer.

If the specified string does not occur after the beginning of the line indicated by the search pointer, the following message is printed.

```
PHRASE NOT FOUND.
```

The search string must be specified to identify uniquely the string being sought. If too small a string is given, the search may result in operating on an occurrence of the string that was not the intended target.

A search string is given in two forms, a single phrase of an ellipsis.

2.7.1 SINGLE PHRASE SEARCH STRING

In a single phrase search string, the entire string on consecutive characters is placed between a pair of delimiters. The string can include as many characters as required (subject to the requirement that the entire command be on a single line), and the search is satisfied only when an identical string is found within a single line of the edit file.

2.7.2 ELLIPSIS SEARCH STRING

An ellipsis string specification consists of two delimited bracket strings,in one of the two previously described formats. The search process attempts to locate a string of consecutive characters that begins with the first phrase and ends with the second phrase. The string implied by an ellipsis search string may appear in the file over more than one line.

Example:

The ellipsis search string

```
:/FORM/,/LONG/ or /FORM/LONG/
```

is satisfied by the string underlined.

THE ELLIPSIS IS A FORM_OF_SHORTHAND_FOR_LONG OR MULTILINE STRINGS.

One frequent source of error in using ellipsis search strings is a tendency to make the bracket strings too short. Consider the following text.

AN ANOTHER EXAMPLE, ASSUME THAT THE_TARGET_STRING_EXTENDS OVER_SEVERAL_LINES_LIKE_THIS_ONE.

If the underlined string is to be referenced, a command with the following string specification might be entered.

```
:/THE/,/ONE/ or /THE/ONE/
```

This does not reference the string desired, however, because the first occurrence of THE is in the word ANOTHER. The string specification

```
:/THE T/,/ONE/ or /THE T/ONE/
```

identifies the underlined string properly.

2.7.3 SPECIAL STRING FIELD

A special string has a format similar to that of a search string. Its interpretation depends on the command word with which it appears. The following are the six types of special string fields and the statements with which they are used.

- Tab stop sequence in a TAB command
- Tab character defined in a DEFTAB command
- A file name in a MERGE, LOCAL, REPLACE or SAVE command
- A column search sequence in a COL command
- A series of EDT commands in a CBx command
- The length of a truncated file in a LENGTH command

2.8 COLUMN_SEARCHES

This string is used to limit the string searches to certain columns within the edit file. The structure of the string is

```
;/COL/ (1)
```

/COL/ (2)

In format 2 there must be two delimiters in a row. The format of the COL parameter has four formats. These are:

COL1,COL2	Search from COL1 to COL2
COL	Search the one column COL only
,COL	Search from the current default starting column (by default set to 1 or set by COL command) to column COL.
COL,	Search from the column COL to the default end column search (by default end of line or set by COL command)

If the column search is used in conjunction with a string search, then the first character of the string must be within the columns indicated to be found. If the column search is used with an ellipsis string, then the string which is found is the first one in which the beginning and ending strings are within the columns specified. This string could be a completely different string than is desired.

2.9 BEGINNING_COLUMN_SEARCHES

By default, EDT will locate strings in which only the first character must be within the columns indicated. Sometimes it is useful th specify that the whole string must be within the indicated columns. The letter B is used to indicate that the whole string must be within the columns.

;b (1)

/b (2)

In format 2 there must be two delimiters in a row.
A delimiter is NOT used if 'b' follows a 'd'.

Example

if the edit file contains the following:

```
123456789
ABCDEFG
 ABCDEFG
  ABCDEFG
```

then the command:

```
B/ABCDEFG//2//B
```

will change the file to the following:

```
123456789
ABCDEFG

  ABCDEFG
```

Only the third line is blanked.

2.10 DELIMITED_STRING_SEARCHES

A delimited string is a string which is proceeded and suceeded by a character that is not a letter a number or a colon. When this option is used, it will ensure that all string manipulations will occur only on delimited strings. Beginning of line and end of line are also considered delimiters. The format of the option is:

```
;d (1)

/d (2)
```

In format 2 there must be two delimiters in a row.
A delimiter is NOT used if 'd' follows a 'b'.

Example

If the edit file is the following:

```
ABCDE
?ABC?
 ABC
```

then the command:

```
L/ABC//D*
```

will list the following

```
?ABC?
 ABC
```

but not the line ABCDE.

2.11 N_PARAMETER

The n parameter is an integer whose meaning depends on the context in which it appears; its use adds flexibility to EDT commands. The following are the possible interpetations.

- The number of lines on which a command is to be performed
- The number of strings on which a command is to be performed
- The number of lines the search pointer is to be moved forward or backward
- The maximum width of the lines in character columns
- The point in a file where new data is to be inserted

When omitted, n is assumed to equal 1 if applicable. the n parameter is not applicible for the commands RESET, LINE, LISTAB, CLEAR, NUMBER(S), DEFTAB, CBx, LENGTH, ASCII, BREAK, COL, ECHO, LOCAL, NORMAL, REPLACE, SAVE, TERM, and END. Negative values of n are allowed only in a SET or FIND command.

An asterisk (*) instead of a number in the n parameter indicates that the operation is performed at or until the end of the edit file. Refer to the description of the particular command of interest for specific details.

2.12 DOCUMENTARY_COMMENTS

To annotate the editing session (possibly for review purposes) append a dollar sign with commentary information. The comment is ignored by the editor.

2.13 COMMAND_LINE

A line of EDT input can consist of more than one command. Commands are separated by a terminator (by default a period).

EDT looks for a terminator outside of the string specifications. There is one exception, whenever the third format of the string definition (/string1/string2/) is used, the second string is scanned for a terminator. If this is a problem use formats one or two or see the next paragraph.

A local terminator can be used to get around the use of a terminator in a string of a command. The local terminator is in effect for one line only. It is specified with a non-alphabetic character as the first character in the line of input. EDT will then scan the line for this new terminator.

When the global terminator appears by itself on the command line, then the last line of input is repeated.

Examples

```
L/LIST/.F/STRING/
?L/LIST/?F/STRING/
.
```

2.14 STRING BUFFER

The string buffer is a temporary storage for information that is to be moved within the edit file.

Information is copied from the edit file into the string buffer using the EXTRACT command. This information can be kept as a local file when the edit session is ended by the LOCAL command or may be inserted elsewhere in the file, using the ADD or CHANGE command.

After the ADD or CHANGE command is entered, the system responds:

```
 ENTER TEXT.
?
```

IF the user responds by typing

```
 $
```

on the same line, the contents of the string buffer are inserted into the edit file at the point or points indicated by the ADD or CHANGE command.

The CLEAR command erases the contents of the string buffer. CLEAR is used whenever the contents of the string buffer is no longer needed. Until a CLEAR command is issued, repeated EXTRACT operations cause extracted strings to appear cumulatively in the string buffer, concatenated in the order of their extraction.

2.15 ENTER_TEXT_REQUEST

The text editor issues an ENTER TEXT request in reponse to an ADD command and in response to a CHANGE command.

After the ENTER TEXT request, type an opening delimiter, followed by the body of text to be entered, and then followed by a closing delimiter. The delimiters do not become part of the actual file.

The delimiter character is the first non-blank character entered in reponse to the ENTER TEXT request. The closing delimiter is the first recurrence of the delimiter character that is followed immediately by a carriage return. The delimiter character may occur in the actual text if it is not immediately followed by a carriage return.

The delimiter may be any non-blank character except a dollar sign ($). If a blank or a dollar sign is entered as a delimiter from an interactive job, EDT responds with:

```
 ILLEGAL DELIMITER - REENTER TEXT.
?
```

For a local or remote batch job, EDT issues the following error message to the user's dayfile

```
 ILLEGAL DELIMITER.
```

expecting the next statement in the INPUT file to be a new command.

For time-sharing origin jobs, the text editor types a question mark at the beginning of each line until the closing delimiter appears. The system then responds:

```
 READY.
?
```

The READY message indicates that the next line entered is treated as an edit command.

If a blank line is desired in the text, at least one space must be entered on a line and then followed with a carriage return. If the closing delimiter followed with a carriage return appears on a line by itself, a blank line is added to the text file. If this method is used by an IAF user, the closing delimiter cannot be the terminal control character or any other character recognized as a terminal defination command (refer to the IAF Reference Manual). If a carriage return alone is entered on a line, a final blank line is added to the text, and an exit from the enter text mode occurs (that is, a return to command mode). If two delimiters alone appear on the input line, then nothing happens to the text file and a return to command mode is issued.

2.16 PROCESSING TERMINAL INTERRUPTS

The time-sharing user may control his edit session with terminal interrupts. The BREAK command allows the user to turn off the interrupt control so that any interrupt the user issues will abort edit session. By default an interrupt will not abort the edit session. When the interrupt is turned on, there are three ways to use interrupts.

- While output is being transmitted to the terminal. The IAF user terminates the transmission of output to the terminal by entering the interruption or termination sequence (refer to the IAF Reference Manual). (In some manuals, these are also referred to as the user break 1 and user break 2 sequences, respectively) All other time-sharing users perform this interruption either by pressing the BREAK, I or S key (on an ASCII code terminal) or by pressing the ATTN key (on a correspondence code terminal). One of the main uses of this interrupt is the termination of unwanted output from execution of a LIST command.

- While the user is entering text in response to an ADD or CHANGE command.

  After entering text in response to and ADD or CHANGE command, the IAF user enters the interruption or termination sequence, and any other time-sharing user types STOP or presses the BREAK key (on an ASCII code terminal) or ATTN key (on a correspondence code terminal) at the beginning of a line to terminate the command. The user is given the choice of retaining or discarding the text just entered. The system does this by typing:

  ```
  DISREGARD PREVIOUS TEXT ?
  ```

  If the user types NO after the question mark the system responds with:

  ```
  READY.
  ```

```
?
```

In this case, the text entered is included in the edit file, and the system awaits a new edit command.

If the user types YES in response to the question, the system responds with:

```
 READY.
?
```

The text just entered is disregarded, and the system awaits a new edit command.

If the user attempts to interrupt or terminate the question, it is treated as an END command, and the editor terminates.

- While the system is processing a command. The IAF user enters the interruption or termination sequence, and all other time-sharing users type STOP or press the BREAK key (on an ASCII code terminal) or ATTN (on a correspondence code terminal) at the beginning of a line to terminate the execution of an edit command. The system gives the output status of the command in execution by printing:

  ```
  INTERRUPT AT LINE n.
  ```

  The output status is then followed by the enquiry:

  ```
  COMMAND CONTINUE?
  ```

  If the user types YES after the question mark, processing continues; if he types NO, the system prints a line indicating how far the command was processed, and processing terminates.

NOTE

Entering the termination sequence or typing STOP after the execution of an edit command immediately terminates the edit session (refer to TERMINATING EDIT SESSION).

3.0 EDT COMMANDS

This section describes the allowable formats for each text editor command and rules governing their use. The commands are grouped by general category of function; for example, the removal of information category includes DELETE and BLANK commands.

A group of contextual examples is included at the end of each category. These examples are designed to illustrate the effect of the various formats, and in particular, to clarify the differences between similiar commands.

Most commands will be illustrated using the longer syntax but the short syntax will also work.

All commands without an explicitly mentioned column specification may use the column and/or delimited string specifications when a search string is used.

3.1 ENTERING COMMANDS

All editor commands are entered at the time-sharing terminal or included in a batch job according to the general format described in section 2 of this manual. After an edit input line is typed and the carriage return is pressed, the editor either processes the commands immediately or requests additional information. In general, each edit command operation is performed relative to the current search pointer. Chapter 4 contains a summary of all EDT messages and requests.

3.2 TEXT LISTING AND SEARCH POINTER CONTROL

3.2.1 LIST COMMAND

The LIST command allows the operator to print all or selected portions of the edit file. the printout can include a string of characters, a single line, a set of lines each including a common character, or a set of contiguous lines. Execution of the LIST command does not change the position of the search pointer.

If an asterisk is specified in the n parameter or if the value of the n parameter extends beyond the end of the edit file, all remaining lines are printed, followed by

-END OF FILE-

If an ellipsis string is specified, a line mode command causes all lines to be printed that contain any portion of the ellipsis string. A string mode command prints only the string implied by the ellipsis.

3.2.1.1 Line Mode Formats (LIST or L)

Command	Explanation
LIST	Prints the line of text specified by the seach pointer.
LIST;n	prints n lines of contiguous text, beginning at the search pointer. (If n equals *, all lines to the end of the edit file are printed.)
LIST:/string/	Prints the line containing the specified string (the phrase must be contained in a single line). Search for string begins at the current position of the search pointer.
LIST:/string1/,/string2/	Prints the line or group of lines containing the

	ellipsis /string1/,/string2/.
LIST/string1/string2/n	Prints the first n occurrances of lines or group of lines containing the ellipsis /string1/string2/

3.2.1.2 String Mode Formats (LISTS of LS)

Command	Explanation
LISTS	Same as LIST
LISTSn	Same as LIST;n
LISTS/string/	Prints the specified string, if present in the edit file. Search for string begins at current position of search pointer.
LISTS,/string/;n	Prints the first n occurrences of the string.
LISTS:/string1/,/string2/	Prints the string of characters specified by the ellipsis /string1/,/string2/.
LISTS/string1/string2/n	Prints the first n occurrances of the string of characters specified by the ellipsis /string1/string2/.

3.2.2 SEARCH POINTER CONTROL (SET AND RESET)

EDT initially locates the search pointer at the first line of the edit file. With the SET command, the seach pointer can be moved to a particular line in the edit file without listing it. The RESET command sets the search pointer to the first line of the edit file, reqardless of its former position. Activity on the edit file always begins at the current search pointer setting.

3.2.2.1 SET Command (SET or S)

The following are the four forms of the SET command.

SET	Advances the search pointer one line relative to its current setting.
SET-	Sets the search pointer back one line from the current position.
SET;n or SET-n	Advances (or sets back) the search pointer n lines relative to its current setting. If the SET instruction results in a negative search pointer (the pointer being set back past the beginning of the file), the pointer is set to the first line. (If n equals * or extends beyond the end of the file, the pointer is set to the end of the edit file.)
SET/string/	Advances the search pointer to the line containing the string, relative to the current setting of the search pointer; if the current line contains the string the search pointer is not moved.
SET,/string/;n	Advances the search pointer from its current setting to the beginning of the nth line containing one or more occurrences of the search string; if there are less than n lines containing at least one occurrence, the search pointer is positioned at the last line containing the string.

The SET command requires locational information. If no search string is present, the use of the n parameter is implied.

Only single-phrase search strings are allowed. Ellipsis search strings are not allowed.

3.2.2.2 RESET Command (RESET or R)

The RESET command brings the serach pointer to the beginning of the edit file. Its format is:

```
RESET
```

Operational fields are not used with the RESET command.

3.2.3 FIND COMMAND

The FIND command scans the edit file, beginning at the line indicated by the search pointer. When a line (or string) is encountered that fulfills the combined requirements of the search string and/or the n parameter, the editor lists that line or string and sets the search pointer accordingly (as explained in the discussion of the FIND formats).

If the end of the edit file is reached before the nth occurrence is found, the search pointer is set to the line of the last string found.

3.2.3.1 Line Mode Formats (FIND or F)

Command	Explanation
FIND	Advances the search pointer one line and lists the line.
FIND;-	Sets back the search pointer one line and lists the line.
FINDn or FIND-n	Advances (or sets back) the search pointer n lines and lists the line indicated by the new value of the search pointer.
FIND:/string/	Advances the search pointer

	to the line that contains the string.
FIND,/string/n	Advances the search pointer to the nth line that contains at least one occurrence of the string and lists the line. n must be positive.
FIND/string1/string2/	Advances the search pointer from its current position to the first line that contains the beginning of the ellipsis search string. If the search string is multi-line, all line containing some part of /string1/string2/ are listed.
FIND:/string1/,/string2/;n	Advances the search pointer to the nth line which has the beginning of the ellipsis and lists the line(s) that has the ellipsis. n must be positive.

3.2.3.2 String Mode Formats (FINDS or FS)

Command	Explanation
FINDS	S ame as FIND.
FINDS;n	Same as FIND;n, but n must be positive.
FINDS/string/n	Advances the search pointer to the line containing the nth occurrence (if specified, otherwise the first) of /string/ and lists the string.
FINDS/string1/string2/;n	Advances the search pointer to the line containing the beginning of the nth

occurrence of /string1/string2/. The string is listed.

3.2.4 LINE COMMAND (LINE OF LN)

The LINE command causes a message to be printed that gives the current setting of the search pointer.
The format is:

```
LINE
```

The message is

```
FILE AT LINE NUMBER n.
```

where n indicates the line of the edit file to which the search pointer is currently pointing. If n is the last line of the file, the words

```
-END OF FILE-
```

are included in the message.

3.2.5 LISTC COMMAND (LISTC OR LC)

The LISTC command prints a line of numbers so that the user can see what columns the data is in. The format of the command is:

```
LISTC or LISTCn
```

where n is the number of columns to be listed (default is 79). The line appears as follows.

```
123456789.123456789.123456789.123456789.123456789.123456789.
```

3.2.6 EXAMPLE

The following example illustrates the use of LIST, SET, RESET, FIND, LINE and LISTC commands.

Entry/Response	Commentary
edt,a	

3.2.6 EXAMPLE

```
    BEGIN TEXT EDITING.
   ?  list;*                         List to end of file
        PROGRAM RANDNUM
        MAINSEED = 5**13
        NEXTSEED = 5**15
        HUNDRETH = 1./100.
        BIGNUMBR = 2.**48
        FRACTION = HUNDRETH=BIGNUMBR
        IFRAC = INT(FRACTION)
        DO 20 I = 1,10
        INCREMENT = 0
        NEXTSEED = NEXTSEED*MAINSEED
   10   INCREMENT = INCREMENT+1
        IF ((INCREMENT*IFRAC).LT.NEXTSEED) GO TO 10
        NEWNUM = INCREMENT
   20   CONTINUE
        END
    -END OF FILE-
   ?  set8                           Advance search      pointer
                                     forward   eight  lines  from
                                     current setting.
   ?  l     $list line
        INCREMENT = 0
   ?  line                           Determine position of search
                                     pointer     relative      to
                                     beginning of file.
    FILE AT LINE NUMBER       9.
   ?  reset                          Reset search    pointer   to
                                     beginning.
   ?  list,znextseedz2               List first  two  occurrences
                                     of lines  containing  string
                                     /nextseed/.
        NEXTSEED = 5**15
        NEXTSEED = NEXTSEED*MAINSEED
   ?  lists]nextseed];3              List first three occurrences
                                     of string /nextseed/.
        NEXTSEED
        NEXTSEED  NEXTSEED
   ?  find;8                         Advance search  pointer  and
                                     list line.
        INCREMENT = 0
   ?  l/20/c/;2                      List lines  containing first
                                     two occurrences of  ellipsis
                                     string /20/c/.
        DO 20 I =1,10
        INCREMENT = 0
   20   CONTINUE
   ?  lists,/20/,/c/2                List first  two  occurrences
                                     of ellipsis string /20/,/c/
```

```
            20 I = 1,10
         C
   20    C
   ?  find;-                         Move search pointer back one
                                     line and list the line.
         DO 20 I = 1,10
   ?  find,ihundrethi
    PHRASE NOT FOUND.
   ?  reset
   ?  find,ihundrethi                Advance search pointer from
                                     current setting to line
                                     containing string
                                     /hundreth/.
         HUNDRETH = 1./100.
   ?  fs/ifrac/2                     Advance search pointer to
                                     line containing second
                                     occurrence of string /ifrac/
                                     and list string.
                     IFRAC
   ?  ln
    FILE AT LINE NUMBER     12.
   ?  s-5                            Move search pointer back
                                     five lines.
   ?  list2                          List two contiguous lines.
         IFRAC = INT(FRACTION)
         DO 20 I = 1,10
   ?  reset
   ?  fs,/13/,/15/                   Advance search pointer to
                                     line containing ellipsis
                                     /13/,/15/ and list line.

                       13
         NEXTSEED = 5**15
   ?  lc30.f-1                       List thirty columns, backup
                                     one line and list the line.
   123456789.123456789.123456789.
         PROGRAM RANDNUM
   ?  end
    END TEXT EDITING.
   READY.
```

3.3 ADDING_AND_BUILDING_TEXT

The ADD, INSERTS, ADDB and ADDC commands cause new information to be included in the edit file at the place specified by the user.

3.3.1 ADD COMMAND

An ADD operation requires two sets of information, the location where the text is added (supplied in the command) and the actual new information to be inserted in the edit file (supplied by the user in response to the ENTER TEXT request).

After the command is entered, the system types:

```
 ENTER TEXT
?
```

Respond to this request in one of four ways.

1. Type the actual information to be added (including carriage returns and line numbers if required), bracketed with delimiters.

2. Type the dollar sign ($) character followed by a single space if you are an IAF user, since just the dollar sign could have a special meaning such as the cancel line character. All other users should type the dollar sign character with no delimiters or other characters. This causes the current contents of the string buffer to be added. (Information is placed in the string buffer by one or more EXTRACT statements.)

3. Type carriage return only. This causes the data entered in response to the most recent ENTER TEXT request to be added.

NOTE

(1) Whenever a MERGE command is issued, the data entered in response t the most recent ENTER TEXT request is lost. In this case, no data is added when the carriage return only is entered in response to an ENTER TEXT request.

(2) If the ENTER TEXT request is issued in a command buffer, a carriage return issued for the first request will cause all other ENTER TEXT requests to simulate a carriage return. This will happen until an input prompt is issued (a question mark).

4. Two delimiters and a carriage return only. This will end the ENTER TEXT request and leave the edit file unaltered.

Only single phrase search strings are allowed with this command. Ellipsis string specifications are illegal.

With no search string specification in force, the n parameter indicates where the insertion shall be made relative to the search pointer.

3.3.1.1 Line Mode Formats (ADD or A)

Command	Explanation
ADD	Inserts text after the line of the edit file specified by the search pointer.
ADD;n	Inserts text after the nth line (counting forward from the search pointer) of the edit file.
ADD/string/	Inserts text after the line containing the specified string; search for string begins at current position of search pointer.
ADD,/string/;n	Inserts text after each of the first n lines containing the specified string.

3.3.1.2 String Mode Formats (ADDS or AS)

Command	Explanation
ADDS	Same as ADD

ADDSn	Same as ADD;n.
ADDS,/string/	Inserts text immediately following the specified string; search for string begins at current position of search pointer.
ADDS:/string/n	Inserts text immediately following each of n occurrences of the specified string.

Line mode ADD commands cause the addition of text following the end of a particular line, whereas string mode ADD commands cause text to be added following a particular string of characters. A string mode command without a string specification is equivalent to a line mode command.

3.3.2 INSERTS COMMAND (INSERTS OR IS)

The INSERTS command is similar in purpose to the ADDS command, except that the text to be inserted is embedded within the command, thus speeding up the interaction.

The command has the following format.

```
INSERTS/string1/string2/;n
```

If the n parameter is omitted, 1 is assumed.

The character string denoted by string2 is inserted immediately after each n occurrence of string1, beginning at the search pointer. Note that /string1/string2/ specification is not an ellipsis search string in this command.

3.3.3 ADD BEFORE COMMAND (ADDB OF AB)

The ADDB command works indentical to the ADD command except that the text is added before the line specified. It also works only in the line mode.

3.3.3.1 ADDB line mode format

Command	Explantion
ADDB	Inserts text before the line of the edit file specified by the search pointer.
ADDBn	Inserts text before the nth line (counting forward from the search pointer) of the edit file.
ADDB,/string/	Inserts text before the line containing the specified string; search for the string begins at the current position of the search pointer.
ADDB/string/n	Inserts text before each of the first n lines containing the specified string.

3.3.4 ADD COLUMN COMMAND (ADDC OR AC)

The ADDC command adds the string starting in the column specified. It moves all existing characters to the end of the string, and thus extends the line. The command format is.

```
ADDC,/string/;/col/n
```

If the n parameter is not specified it is assumed to be 1.

The contents of string are added to the first n lines.

3.3.5 EXAMPLE

The following example illustrates the use of the ADD, INSERTS and ADDC commands.

Entry/Response	Commentary
edt,filel	EDT is called, creating empty working file 'filel'.
BEGIN TEXT EDITING.	
? add	The file is built using the

```
                                        ADD command.
    ENTER TEXT.
   ?  /  the add command can be very
   ?  useful when creating a textual
   ?  file.   /
    READY.
   ?  adds,+file.   +                   Text is added    immediately
                                        after  the  first occurrence
                                        of string /file.    /.
    ENTER TEXT.
   ?  /in fact, it is one
   ?  of the few methods that can be
   ?  used to build a direct access
   ?  file./
    READY.
   ?  list*                             List to end of file.
     THE ADD COMMAND CAN BE
   USEFUL WHEN CREATING A TEXTUAL
   FILE.  IN FACT, IT IS ONE
   OF THE FEW METHODS THAT CAN BE
   USED TO BUILD A DIRECT ACCESS
   FILE.
    -END OF FILE-
   ?  inserts/access/permanent/         The string   /permanent/  is
                                        inserted  after  the   first
                                        occurrence   of  the  string
                                        /access/.
   ?  find4
   USED TO BUILD A DIRECT ACCESS PERMANENT
   ?  as:,file,                         Text is added directly after
                                        the first occurrence of  the
                                        string /file/.
    ENTER TEXT.
   ?  t, providing it is the edit filet
    READY.
   ?  a*                                Text is  added to the end of
                                        file.
    ENTER TEXT.
   ?  = it is also useful when adding
   ?  text to a previously existing file.=
    READY.
   ?  reset
   ?  l;*                               List the whole file.]
     THE ADD COMMAND CAN BE VERY
   USEFUL WHEN CREATING A TEXTUAL
   FILE.  IN FACT, IT IS ONE
   OF THE FEW METHODS THAT CAN BE
   USED TO BUILD A DIRECT ACCESS PERMANENT
   FILE, PROVIDING IT IS THE EDIT FILE.
```

```
     IT IS ALSO USEFUL WHEN ADDING
   TEXT TO A PREVIOUSLY EXISTING FILE.
    -END OF FILE-
   ?  a/existing/                        Add text   after   the  line
                                         containing   the    string
                                         /existing/.
    ENTER TEXT.
   ?  /later it will be demonstrated how to
   ?  use the add command to remove text
   ?  string buffer./
    READY.
   ?  s8;                                Advance search pointer eight
                                         lines.
   ?  l;3
   LATER IT WILL BE DEMONSTRATED HOW TO
   USE THE ADD COMMAND TO REMOVE TEXT
   FROM THE STRING BUFFER.
   ?  addc,+  ++1+                       Add string /  /  to column 1
                                         of current line.
   ?  l
     LATER IT WILL BE DEMONSTRATED HOW TO
   ?  r
   ?  s&edit file&
   ?  adds                               Same as ADD command.
    ENTER TEXT.
   ?  !  it is especially useful when
   ?  adding text in the body of a file.!
    READY.
   ?  l3                                 List the next three lines.
   FILE, PROVIDING IT IS THE EDIT FILE.
     IT IS ESPECIALLY USEFUL WHEN
   ADDING TEXT IN THE BODY OF A FILE.
   ?  ls#textual#,# or source#           Add string /  or  source/
                                         directly  after  the  first
                                         occurrence    of    string
                                         /textual/.
    PHRASE NOT FOUND.
   ?  reset.ls,xtextualx or sourcex.l;*
     THE ADD COMMAND CAN BE VERY
   USEFUL WHEN CREATING A TEXTUAL OR SOURCE
   FILE.  IN FACT, IT ONE
   OF THE FEW METHODS THAT CAN BE
   USED TO BUILD A DIRECT ACCESS PERMANENT
   FILE, PROVIDING IT IS THE EDIT FILE.
     IT IS ESPECIALLY USEFUL WHEN
   ADDING TEXT IN THE BODY OF A FILE.
     IT IS ALSO USEFUL WHEN ADDING
   TEXT TO A PREVIOUSLY EXISTING FILE.
     LATER IT WILL BE DEMONSTRATED HOW TO
```

```
USE THE ADD COMMAND TO REMOVE TEXT
FROM THE STRING BUFFER.
 -END OF FILE-
?  end
 END TEXT EDITING.
READY.
```

3.4 REMOVAL_OF_INFORMATION

Four types of operation are available for removing information from the edit file, DELETE, BLANK, DELETEC and delete end of line.

3.4.1 DELETE COMMAND

A DELETE operation erases one or more occurrences of a particular string of characters or one or more lines containing a particular string of characters. The text is realigned, leaving no excess blanks. All operations begin at the current position of the search pointer.

3.4.1.1 Line_Mode_Formats_(DELETE_OR_D)

Command	Explanation
DELETE	Erases the line of the edit file specified by the search pointer.
DELETEn	Erases the first n lines of the edit file beginning at the search pointer.
DELETE,/string/	Erases the line containing the string.
DELETE:/string/n	Erases the first n lines containing the string.
DELETE/string1/,/string2/	Erases the first line or group of lines containing ellipsis /string1/,/string2/.
DELETE,/string1/string2/;n	Erases the first n occurrences of the line or group of lines containing ellipsis /string1/string2/.

3.4.1.2 String_Mode_Formats_(DELETES_or_DS)

Command	Explanation
DELETES	Same as DELETE.

DELETES;n	Same as DELETEn.
DELETES/string/	Erases the specified string.
DELETES,/string/;n	Erases the first n occurrences of the specified string.
DELETES/string1/,/string2/	Erases the string of characters specified by the ellipsis /string1/,/string2/.
DELETES:/string1/string2/;n	Erases the first n occurrences of the string of characters specified by the ellipsis /string1/string2/.

3.4.2 BLANK COMMAND

The BLANK command replaces the specified string, line or set of lines with blank characters. Unlike the DELETE, DELETEC and DEOL commands, BLANK does not relocate text. All operations begin at the current position of the search pointer.

3.4.2.1 Line_Mode_Format_(BLANK_or_B)

Command	Explanation
BLANK	Replaces with blanks the line of the edit file specified by the search pointer.
BLANKn	Replaces with blanks the first n lines of the edit file, beginning at the search pointer.
BLANK,/string/	Replaces the line containing the string with blanks.
BLANK:/string/;n	Replaces with blanks the first n lines containing the string.

BLANK/string1/,/string2/	Replaces with blanks the first line or group of lines containing ellipsis /string1/,/string2/.
BLANK/string1/string2/n	Replaces with blanks the first n occurrences of the line or group of lines containing ellipsis /string1/string2/.

3.4.2.2 String_Mode_Formats_(BLANKS_or_BS)

Command	Explanation
BLANKS	Same as BLANK.
BLANKS;n	Same as BLANKn.
BLANKS:/string/	Replaces with blanks the specified phrase.
BLANKS,/string/;n	Replaces with blanks the first n occurrences of the specified phrase.
BLANKS,/string1/,/string2/	Replaces with blanks the string defined by the ellipsis /string1/,/string2/.
BLANKS/string1/string2/;n	Replaces with blanks the first n occurrences of the string defined by the ellipsis /string1/string2/.

3.4.3 DELETE COLUMN COMMAND (DELETEC OR DC)

The DELETEC command removes all information in the columns specified. Like the DELETE command the edit file is realigned. The column specification must be present. The columns indicated are included in the deleted columns. Note, a semi-colon cannot appear before the column specifier in this one command. The format of the command follows.

```
DELETEC//col/;n
```

If the n parameter is not present, 1 is assumed.

The columns specified in the col specifier are deleted from the first n lines. The rest of the characters in these lines are packed together.

3.4.4 DELETE END OF LINE (DEOL)

The DEOL command will merge lines together. The end of line terminator is removed and the next line is abuted next to the last non-blank character of the current line. The format of the command is as follows.

```
DEOL;n
```

The first n (by default 1) end of lines are deleted. This causes the first n+1 lines to become one line. If the new line is more than 150 characters some of the character may be lost through truncation of the line.

3.4.5 EXAMPLE

The following is an example of DELETE, DELETEC, DEOL and BLANK commands.

```
Entry/Response                          Commentary

edt,a
 BEGIN TEXT EDITING.
?  l*                                   List to end of file.
      PROGRAM RANDNUM
      MAINSEED = 5**13
      NEXTSEED = 5**15
      HUNDRETH = 1./100.
      BIGNUMBR = 2.**48
      FRACTION = HUNDRETH*BIG
      IFRAC = INT(FRACTION)
      DO 20 I = 1,10
      INCREMENT = 0
      NEXTSEED = NEXTSEED*MAINSEED
10    INCREMENT = INCREMENT+1
      IF((INCREMENT*IFRAC) .LT.NEXTSEED) GO TO 10
      NEWNUM = INCREMENT
20    CONTINUE
      END
 -END OF FILE-
?  delete,+bigNUMBRnumbr+*              Delete every line in file
```

```
                                  that  contains  the   string
                                  /bignumbr/.
       2  OCCURRENCES OF PHRASE FOUND.
?  s;3                            Advance search pointer three
                                  lines.

?  l2
      HUNDRETH = 1./100.
      IFRAC = INT(FRACTION)
?  a
 ENTER TEXT.
?  /      big = 2**48
?       fraction = hundreth*big/
 READY.
?  blanks,xhundrethx              Blank first  occurrence  of
                                  string /hundreth/.

?  l;2
               = 1./100.
      BIG = 2.**48
?  ac,?h100?;?7?                  Insert string        /h100/
                                  starting in column seven.

?  l
      H100         = 1./100.
?  ds/        /                   Delete eight spaces  denoted
                                  by string /        /.

?  l
      H100 = 1./100.
?  blanks)undreth).line           Blank first  occurrence   of
                                  string  /undreth/  and  get
                                  current line number.

 FILE AT LINE NUMBER      4.
?  f2
      FRACTION = H       *BIG
?  adds/h/
 ENTER TEXT.
?  /100/
 READY.
?  lc31.l
123456789.123456789.123456789.1
      FRACTION = H100        *BIG
?  dc??22,29?                     Delete columns  22  to  29,
                                  inclusive.

?  l
      FRACTION = H100*BIG
?  r.f;13
20    CONTINUE
?  ds,/20/end/                    Delete string  specified  by
                                  ellipsis /20/,/end/.  Note:
                                  string deleted not line

?  l;*
```

```
 -END OF FILE-
?  ac,/ print,newnum//6/             Add string    starting    in
                                     column 6 (there is no column
                                     seven).

?  l.a
      PRINT,NEWNUM
 ENTER TEXT.
?  /20    cont                       A carriage return        was
                                     pressed by mistake.

?  inue
?  end/
 READY.
?  l4
      PRINT,NEWNUM
20    CONT
INUE
      END
?  s.deol                            Advance search  pointer  one
                                     line and remove the  end  of
                                     line

?  l
20    CONTINUE
?  end
 END TEXT EDITING.
READY.
```

3.5 SUBSTITUTION OF INFORMATION

The CHANGE, CHANGEC, CEOL and RS commands each cause a specified set of text information to replace text already present in the edit file. The length of the new information is independent of the length of the replaced text.

3.5.1 CHANGE COMMAND

In effect, the CHANGE command combines a DELETE operation and an ADD operation. A complete CHANGE operation requires two sets of information, a definition of the area to be changed (which is supplied in the CHANGE command) and the information that is to be inserted into that area (which is supplied by the user in response to the ENTER TEXT request).

After the command is entered, the system types:

```
 ENTER TEXT.
?
```

Respond to this request in one of four ways.

1. Type actual change information (including carriage return and line numbers, if required), bracketed with delimiters.

2. Type the dollar sign ($) character followed by a single space if you are an IAF user, since just the dollar sign could have a special meaning such as the cancel last character. All other users should type the dollar sign character with no delimiters or other characters. This causes the current contents of the string buffer to be used as the change information. (Information is placed in the string buffer by one or more EXTRACT statements.)

3. Type carriage return only. This causes the data entered in response to the most recent ENTER TEXT request to be used as the change information.

<u>NOTE</u>

Whenever a MERGE command is issued, the data entered in the most recent ENTER TEXT request is lost. In this case, no data is added if carriage return only is entered in response to an ENTER TEXT request.

4. Type two delimiters and a carriage return. This is a do nothing instruction and will terminate the ENTER TEXT request without altering the edit file.

3.5.1.1 Line Mode Formats (CHANGE or C)

Command	Explanation
CHANGE	Replace the line specified by the search pointer with the text that follows.
CHANGEn	Replace the first n lines of the edit file beginning at the search pointer with the text that follows.
CHANGE/string/	Replace the line containing the string with the text. The search for the string starts at the current position of the search pointer.
CHANGE:/string/;n	Replaces the first n lines containing the string with the following text.
CHANGE,/string1/string2/	Replaces the line or group of line containing ellipsis /string1/string2/ with the following text.
CHANGE/string1/,/string2/n	Replace the first n occurrences of the lines containing the ellipsis /string1/,/string2/ with the following text.

3.5.1.2 String Mode Format (CHANGES or CS)

Command	Explanation
CHANGES	Same as CHANGE
CHANGESn	Same as CHANGEn.

CHANGES/string/	Replace the specified string with the following text. The string search starts at the current search pointer.
CHANGES,/string/;n	Replace the first n occurrences of the string with the following text.
CHANGES:/string1/string2/	Replace the string of characters specified by the ellipsis /string1/string2/ with the following text.
CHANGES/string1/string2/n	Replaces the first n occurrences of the string of characters specified by the ellipsis /string1/string2/ with the following text.

3.5.2 RS COMMAND

The RS command is similar to the CHANGE command except that it performs only string replacements and the replacement text is embedded in the command, thus speeding the interaction. Also, the structure of the RS command does not allow ellipsis string specifications.

The format of the command is as follows.

```
RS/string1/string2/n
```

The first n (by default 1) occurrences of string1 will be replaced with string2.

3.5.3 CHANGEC COMMAND (CHANGEC OR CC)

The CHANGEC command will replace the indicated columns with the string. Unlike the CHANGE and RS commands, the CHANGEC command works on columns instead of lines. The columns will be changed irregardless of the contents. The format of the command is as follows.

```
CHANGEC/string/;/col/;n
```

The first n (by default 1) lines containing the columns indicated by 'col' will be replaced with 'string'.

3.5.4 CEOL COMMAND (CEOL)

The CEOL command is an easier way to replace the end of line with a string plus an end of line. The string will be placed before the end of line and will remove all spaces between the end of line and the last character. In this one command the column and delimited string specifiers cannot be used. The format of the command is as follows.

```
CEOL/string/;n
```

In the first n (by default 1) lines of the edit file the end of line will be replaced by the string plus and end of line.

3.5.5 EXAMPLE

The following is an example of CHANGE, CHANGEC, CEOL and RS commands.

```
Entry/Response                      Commentary

edt,a
 BEGIN TEXT EDITING.
?  l*                               List to end of file.
      PROGRAM RANDNUM
      MAINSEED = 5**13
      NEXTSEED = 5**15
      H100 = 1./100.
      BIG = 2.**48
      FRACTION = H100*BIG
      IFRAC = INT(FRACTION)
      DO 20 I = 1,10
      INCREMENT = 0
      NEXTSEED = NEXTSEED*MAINSEED
10    INCREMENT = INCREMENT+1
      IF((INCREMENT*IFRAC) .LT.NEXTSEED) GO TO 10
      NEWNUM = INCREMENT
      PRINT,NEWNUM
20    CONTINUE
      END
 -END OF FILE-
?  change                           Change line  indicated  by
                                    current  setting  of  search
                                    pointer.
 ENTER TEXT.
?  /      program random (output)/
 READY.
```

3.5.5 EXAMPLE

```
   ?  rs&nextseed&,&nextsd&*         Replace each occurrence of
                                     the string /nextseed/ with
                                     /nextsd/.
        4  OCCURRENCES OF PHRASE FOUND.
   ?  f2
        NEXTSD = 5**15
   ?  f;7
        NEXTSD = NEXTSD*MAINSEED
   ?  cs/seed/                       Change first occurrence of
                                     string /seed/ with string
                                     /sd/.
    ENTER TEXT.
   ?  ?sd?
    READY.
   ?  l
        NEXTSD = NEXTSD*MAINSD
   ?  /r/cs,#seed#15#/l;3            Reset the search pointer,
                                     change ellipsis string
                                     /seed/15/ and list three
                                     lines.
    ENTER TEXT.
   ?  8sd = 5**13
   ?       nextsd = 5**178
    READY.
        PROGRAM RANDOM (OUTPUT)
        MAINSD = 5**13
        NEXTSD = 5**17
   ?  rs,=increment=,=inc=5          Replace first five
                                     occurrences of string
                                     /increment/ with /inc/.
   ?  rs/fraction/,/frac/;2          Replace two occurrences of
                                     /fraction/ with /frac/.
   ?  ls/inc/;5
        INC
           INC   INC
            INC
                 INC
   ?  change
    ENTER TEXT.
   ?  mc      thus program generate
   ?  c      10 random numbers between
   ?  c      1 and 100.
   ?         program random (output)m
    READY.
   ?  lc44.l
123456789.123456789.123456789.123456789.1234
C      THUS PROGRAM GENERATE
   ?  cc,/i//9/                      Change column nine to string
                                     /i/
```

3.5.5 EXAMPLE

```
   ?  ceol,/s/                             Replace carriage return with
                                           string  /s/  and  carriage
                                           return.
   ?  l
   C      THIS PROGRAM GENERATES
   ?  c/newnum/newnum/                     Change line(s)   containing
                                           ellipsis              string
                                           /newnum/newnum/.
    ENTER TEXT.
   ?  .      nran = inc
   ?        print 30 i,nran
   ?  30    format (i2,2x,i4)
   ?  .
    READY.
   ?  l*
   C     THIS PROGRAM GENERATES
   C     RANDOM NUMBERS BETWEEN
   C     1 AND 100.
         PROGRAM RANDOM (OUTPUT)
         MAINSD = 5**13
         NEXTSD = 5**17
         H100 = 1./100.
         BIG = 2.**48
         FRAC = H100*BIG
         IFRAC = INT(FRAC)
         DO 20 I = 1,10
         INC = 0
         NEXTSD = NEXTSD*MAINSD
   10    INC = INC+1
         IF((INC*IFRAC).LT.NEXTSD) GO TO 10
         NRAN = INC
         PRINT 30, I, NRAN
   30    FORMAT (I2,2X,I4)
   20    CONTINUE
         END
    -END OF FILE-
   ?  end
    END TEXT EDITING.
   READY.
```

3.6 LOADING AND CLEARING THE STRING BUFFER

The string buffer (described in section 2) can be loaded with the EXTRACT command. Information is transfered from the string buffer to the edit file with the ADD and CHANGE commands. The buffer is not automatically cleared when information is transfered by either of these commands; it is cleared only with the CLEAR command. The string buffer may also be kept as a local file when the edit session is finished with the LOCAL command.

3.6.1 EXTRACT COMMAND

The EXTRACT command appends a copy of information from the edit file to the string buffer. This operation has no effect on the edit file.

3.6.1.1 Line Mode Formats (EXTRACT or E)

Command	Explanation
EXTRACT	Copies one line beginning at the search pointer.
EXTRACT;n	Copies n lines beginning at the search pointer. (If n equal *, all lines to the end of file are copied.)
EXTRACT/string/	Copies the first line containing the string; the search for the string starts at current position of the search pointer.
EXTRACT/string/n	Copies the first n lines containing the string.
EXTRACT,/string1/,/string2/	Copies the first line or group of lines containing the ellipsis /string1/,/string2/.
EXTRACT/string1/string2/n	Copies the first n occurrences of the line or group of lines containing the ellipsis /string1/string2/.

3.6.1.2 String Mode Formats (ES)

Commands	Explanation
ES	Same as EXTRACT.
ESn	Same as EXTRACTn.
ES/string/	Copies the string specified; search for string begins at current position of the search pointer.
ES,/string/;n	Copies the nth occurrence of the string.
ES:/string1/,/string2/	Copies the string of characters specified by the ellipsis /string1/,/string2/.
ES/string1/string2/n	Copies the nth occurrence of characters specified by the ellipsis /string1/string2/.

3.6.2 CLEAR COMMAND (CLEAR OR CL)

The CLEAR command clears the string buffer. It is the user's responsibility to clear this buffer and if he fails to do so, information from subsequent EXTRACT operations is appended to the information from previous EXTRACT operations. The format is:

```
 CLEAR
```

Operand fields are never used with this command.

3.6.3 LOCAL COMMAND (LOCAL)

The LOCAL command causes the string buffer to remain as a local file when the edit session is ended. Thus a section of the edit file may be extracted and kept local. Only the information in the string buffer at the end of the edit session will be kept. A MERGE command removes all previous information from the string buffer. The format of the command is:

```
LOCAL/lfn/
```

The string buffer will be a local file called lfn at the end of the edit session. Note: the last LOCAL command (if used more than once) will contain the lfn.

If the string buffer has not been used the command will be ignored and the following message issued.

```
STRING BUFFER HAS NOT BEEN USED.
```

If there is a problem in making the file local, the command will not be completed and the following message will be issued.

```
LOCAL FILE ERROR.
```

If the file name specified is a reserved or invalid file name, then the command will be ignored and the following message issued.

```
RESERVED FILE NAME.
```

3.6.4 EXAMPLE

The following example illustrates the use of EXTRACT, CLEAR and LOCAL commands.

```
Entry/Response                    Commentary

edt,a
 BEGIN TEXT EDITING.
?  list*
    THE EXTRACT COMMAND CAN BE
VERY USEFUL IN REARRANGING
LINES OF TEXT.
 -END OF FILE-
?  s2.extract                     The third line is copied
                                  into the string buffer.
?  r.a.l;*                        The contents of the string
                                  biffer are inserted into the
                                  file.
 ENTER TEXT.
?  $
 READY.
    THE EXTRACT COMMAND CAN BE
LINES OF TEXT.
VERY USEFUL IN REARRANGING
```

```
LINES OF TEXT.
 -END OF FILE-
?  s/lines/2.d.r                       Delete line      containing
                                       second     occurrenc     of
                                       /lines/.
?  a
 ENTER TEXT.
?  )used to restructure individual)
 READY.
?  es/rearranging/.a*.l;*              Copy string  /rearranging to
                                       string buffer.
 ENTER TEXT.
?  $
 READY.
     THE EXTRACT COMMAND CAN BE
USED TO RESTRUCTURE INDIVIDUAL
LNES OF TEXT.
VERY USEFUL IN REARRANGING
LINES OF TEXT.
REARRANGING
 -END OF FILE-
?  clear                               Clear string buffer.
?  s]very].d*
 -END OF FILE-
?  es/restructure/
 -END OF FILE-
?  r.es%restructure%
?  cs,kindividualk.l;*
 ENTER TEXT.
?  $
 READY.
    THE EXTRACT COMMAND CAN BE
USED TO RESTRUCTURE RESTRUCTURE
LINES OF TEXT.
 -END OF FILE-
?  ds]restructure/.a*
 ENTER TEXT.
?  (     remember that the string
?  buffer is not cleared after an
?  add or change $ command.
?       to remove text from the string
?  buffer, use the clear command.(
 READY.
?  clear                               Clear string buffer.
?  es# string#buffer, #
?  ds# string#buffer#                  Note difference     between
                                       these two ellipsis  strings.
?  as:<that the<.l*
 ENTER TEXT.
```

```
? $
 READY.
    THE EXTRACT COMMAND CAN BE
USED TO RESTRUCTURE
LINES OF TEXT.
    REMEMBER THAT THE STRING
BUFFER IS NOT CLEARED AFTER AN
ADD OR CHANGE $ COMMAND.
    TO REMOVE TEXT FROM THE STRING
BUFFER,
 IS NOT CLEARED AFTER AN
ADD OR CHANGE $ COMMAND.
    TO REMOVE TEXT FROM THE STRING
BUFFER, USE THE CLEAR COMMAND.
 -END OF FILE-
?  cl.d,zbuffer,z,ze stringz.l*
    THE EXTRACT COMMAND CAN BE
USED TO RESTRUCTURE
LINES OF TEXT.
    REMEMBER THAT THE STRING
BUFFER IS NOT CLEARED AFTER AN
ADD OR CHANGE $ COMMAND.
    TO REMOVE TEXT FROM THE STRING
BUFFER, USE THE CLEAR COMMAND.
-END OF FILE-
e3.local/abcd/.end
 END TEXT EDITING.
READY.
lnh,abcd
    THE EXTRACT COMMAND CAN BE
USED TO RESTRUCTURE
LINES OF TEXT.
```

3.7 EDIT FILE DIMENSIONING COMMANDS

The LENGTH, COLUMN and WIDTH commands are used to re-specify the dimensions of the edit file. The ALIGN command removes extraneous blanks for printing purposes.

3.7.1 LENGTH COMMAND (LENGTH)

The LENGTH command will shorten a file. This will have the effect of simulating to the editor that the file was never longer than the new shortened file. Thus the area of the file outside the truncated part cannot be affected by any command except for the LENGTH command. In a large file where only a small portion of the file is to be edited, this command will also speed up the interaction. The file can be shortened from both ends of the file. The format of the command is:

```
LENGTH:/line/
```

The file will be shortened if line is specified and restored to its full length if line is absent. The string line has the same format as the col specifier. The search pointer is placed at the first line of the shortened file and at the current line when the file is restored.

If 'line' equals /5,10/ , then the lines five through ten will be the only lines in the edit file.

If the file has been truncated and a second truncation is attempted (before the file has been restored), the command is ignored and the following message is issued.

```
FILE ALREADY TRUNCATED.
```

If the file has not been truncated (or has been restored) and a restore is attempted, the command is ignored and the following message issued.

```
FILE NOT TRUNCATED.
```

If the edit session is terminated by the END command and the file is truncated, the following message is issued before the END occurs.

```
WARNING: FILE HAS BEEN TRUNCATED.
DO YOU WANT TO END ?
```

If the session is to be ended with a shortened file enter

YES (or Y), otherwise enter NO (or N) and the END command is ignored.

NOTE

IF the editor is aborted with the ABORT or STOP command, the shortened file will remain shortened. The part of the file which preceeds the truncated file is in the file SCR3. The trailing end of the edit file is in the file SCR6.

If the file is not long enough for the truncation the following message will be issued and the command ignored.

```
FILE NOT LONG ENOUGH.
```

3.7.2 WIDTH COMMAND (WIDTH OR W)

The WIDTH command defines the maximum number of character columns that can be contained in a single line of the edit file when used with the ALIGN command. The command has no effect unless followed by an ALIGN command. The format is:

```
WIDTH;n
```

where n is the new line length, and 6 =< n =< 150. Note, however, that if n is larger than the size of the carriage, over-print may result on the right-hand end of a printed line.

Following a WIDTH command, the ALIGN command can be used to remove superfluous blanks and reformat in accordance with the changed right margin.

3.7.3 ALIGN COMMAND (ALIGN OR AL)

The ALIGN command eliminates extraneous blanks from the edit file, while retaining the structural integrity of words, sentences and paragraphs.

A word is defined as a set of characters between spaces. A sentence is defined as a group of words ending with a period (or question mark). The beginning of a paragraph is defined by an indented sentence.

The ALIGN command indents five spaces at the beginning of each paragraph, separates each word with one blank, and separates each sentence (group of words ending with a period or question mark) with two blanks. Blank lines are not removed as it is assumed that they serve a purpose in delimiting paragraphs and lines.

The following forms are valid forms of this command.

Command	Explanation
ALIGN	Removes excess blanks between words in the line of text specified by the search pointer.
ALIGNn	Removes excess blanks between words in n lines of text beginning at the search pointer. As many complete words as possible are placed in a line before starting another line.
ALIGN/string/	Removes blanks from the line of text containing the specified string; search for the string begins at current position of the search pointer.
ALIGN,/string/;n	Removes blanks from the first n lines containing the specified string.
ALIGN:/string1/string2/	Removes blanks from the lines of text specified by ellipsis /string1/string2/
ALIGN/string1/,/string2/;n	Removes blanks from the first n occurrences of the line or group of lines specified by ellipsis /string1/,/string2/.

3.7.4 COLUMN COMMAND (COLUMN OR COL)

The COLUMN command changes the default column limits. Whenever a command using strings is executed the default column limits are used unless the column limits are specified on the command. The default limits are 1 to 150. If the starting column is greater than the ending
column or a column is greater than 150 then a syntax error will occur. The command format is:

```
COLUMN/col/
```

The columns specified by col (same format as the column specifier) will become the default column limits.

3.7.5 EXAMPLE

The following section illustrates the use of LENGTH, WIDTH, ALIGN and COL commands.

<u>Entry/Response</u>	<u>Commentary</u>
edt,aa BEGIN TEXT EDITING. ? l* THE LENGTH COMMAND IS VERY USEFUL WHEN IT IS DESIRABLE TO WORK ON ONLY A SMALL PORTION OF A LARGE FILE. THE WIDTH COMMAND IS EFFECTIVE ONLY IF FOLLOWED BY AN ALIGN COMMAND. -END OF FILE-	
? length/1,4/	Truncate edit file to include lines one to four.
? width;20	Set width indicator to 20.
? l;* THE LENGTH COMMAND IS VERY USEFUL WHEN IT IS DESIRABLE TO WORK ON ONLY A SMALL PORTION OF A LARGE FILE. -END OF FILE-	
? length	Restore truncated file.
? s4.align*	Move the search pointer forward four lines and align to end of file with width of 20.

```
?  r.l*
     THE LENGTH COMMAND IS VERY
USEFUL WHEN IT IS DESIRABLE TO
WORK ON ONLY A SMALL PORTION
OF A LARGE FILE.
     THE WIDTH
COMMAND IS EFFECTIVE
ONLY IF FOLLOWED BY
AN ALIGN COMMAND.
 -END OF FILE-
?  align.l4                         Align line   specified   by
                                    search pointer.

     THE LENGTH
COMMAND IS VERY
USEFUL WHEN IT IS DESIRABLE TO
WORK ON ONLY A SMALL PORTION
?  al,/very/work/.l*                Align lines        containing
                                    ellipsis               string
                                    /very/work/.

     THE LENGTH
COMMAND IS VERY
USEFUL WHEN IT IS
DESIRABLE TO WORK ON
ONLY A SMALL
PORTION
OF A LARGE FILE.
     THE WIDTH
COMMAND IS EFFECTIVE
ONLY IF FOLLOWED BY
AN ALIGN COMMAND.
 -END OF FILE-
?  w;62.al*.l*                      Set width  to  62  and align
                                    entire file.
     THE LENGTH COMMAND IS VERY USEFUL WHEN IT IS DESIRABLE TO
WORK ON ONLY A SMALL PORTION OF A LARGE FILE.
     THE WIDTH COMMAND IS EFFECTIVE ONLY IF FOLLOWED BY AN
ALIGN COMMAND.
 -END OF FILE-
?  col/,40/                         Set column search to columns
                                    1 to 40.
?  l/desirable/                     Search for             string
                                    /desirable/.   It   is   not
                                    found  because  it  is   not
                                    contained  in  columns  1 to
                                    40.
?  PHRASE NOT FOUND.
?  col/30,70/
?  l]desirable]
     THE LENGTH COMMAND IS VERY USEFUL WHEN IT IS DESIRABLE TO
```

3.7.5 EXAMPLE

```
? blanks/work on/
? width32.al;*.l*
     THE LENGTH COMMAND IS VERY
USEFUL WHEN IT IS DESIRABLE TO
     ONLY A SMALL PORTION OF A
LARGE FILE.
     THE WIDTH COMMAND IS
EFFECTIVE ONLY IF FOLLOWED BY
AN ALIGN COMMAND.
 -END OF FILE-
? end
 END TEXT EDITING.
READY.
```

3.8 TABULATION COMMANDS

The commands DEFTAB, TAB and LISTAB allow the user to create structured text using tab settings.

3.8.1 DEFTAB COMMAND (DEFTAB OR DT)

The DEFTAB command defines a single tab character that is used (when responding to an ENTER TEXT request) to cause blank fill to the next tab stop. The tab character must not be present in the body of text that is to be created. Each typing of the tab character that occurs when entering text is ignored, except for purposes of tab control.

The following are valid forms of the command.

Command	Explanation
DEFTAB	clears previous tab character definition.
DEFTAB/tabchar/	Defines the character tabchar as a tab character.

3.8.2 TAB COMMAND (TAB OR T)

The TAB command sets tab stops at specified input columns. Default column numbers are 11, 18, 30, 40, 50.

The following forms of the command are valid.

Command	Explanation
TAB	Clears existing tab stops.
TAB,/t1,t2,...,tn/	Each ti is a column number, ti>0. A maximun of seven tab column numbers may be specified.
TAB/name/	The tab character and tab column numbers associated with the name in the following list are set.

List of Tab Mnemonics

Tab	Tab Stops						
CMPS	11	18	30	40			
ALGOL	7	10	13	16	19		
FTN	7						
SWL	3	5	7	9	11	13	15
COBOL	8	12	16	20	24		
CMPS86	10	20	41	50	60		
META	10	20	50				

The default tab character is ';' for all names except ALGOL's '$' and SWL's '?'.

Only one TAB command can be active at one time. Entering a TAB command negates the effect of any prior TAB command.

Since tabulation specification applies to input text, it must be made before the text is entered.

3.8.3 LISTAB COMMAND (LISTAB OR LT)

The LISTAB command causes a listing of the tab stops as specified in the most recent TAB command. The command format is:

```
LISTAB
```

The system responds:

```
TAB c STOPS t1,t2,...,tn
```

where c is the tab character and tn are the tab columns. If the tab has been cleared (refer to TAB command), the system responds:

```
TAB : STOPS NONE.
```

3.8.4 EXAMPLE

The following example illustrates the use of TAB, DEFTAB, and LISTAB commands.

Entry/Response **Commentary**

```
edt,a.
```

3.8.4 EXAMPLE

```
    BEGIN TEXT EDITING.
   ?  add
    ENTER TEXT.
   ?  /  the deftab and tab commands
   ?  are effective only if given prior
   ?  to an enter text request.  thus
   ?  the following will not be tabulated.
   ?  1#2#3#4
   ?  5#6#7#8
   ?  now define a tab character
   ?  and a set of tab stops./
    READY.
   ?  deftab:/#/                          Define the  character  #  as
                                          the tab
   ?  tab,/5,10,20/                       character with   stops    at
                                          5,10, and 20.
   ?  add*
    ENTER TEXT.
   ?  /a#b#c#d
   ?  e#f#g#h/
    READY.
   ?  list*
     THE DEFTAB AND TAB COMMANDS
   ARE EFFECTIVE ONLY IF GIVEN PRIOR
   TO AN ENTER TEXT REQUEST.  THUS,
   THE FOLLOWING WILL NOT BE TABULATED.
   1#2#3#4
   5#6#7#8
   NOW DEFINE A TAB CHARACTER
   AND A SET OF TAB STOPS.
   A   B    C         D
   E   F    G         H
    -END OF FILE-
   ?  listab
    TAB # STOPS     5   10   20
   ?  deftab
   ?  tab
   ?  lt
    TAB : STOPS NONE.
   ?  tab/cmps/
   ?  lt
    TAB ; STOPS   11   18   30   40
   ?  end
    END TEXT EDITING.
   READY.
```

3.9 EXTERNAL FILE COMMANDS

There are three file commands which will allow the user to stay in the editor and switch edit files. These commands are the MERGE, REPLACE and SAVE commands. The command MERGEL is a helper command for MERGE.

3.9.1 MERGEL COMMAND (MERGEL OR ML)

The MERGEL command is used to specify which lines of the merge file are to be added to the file. The command does not merge the files. It only sets up the parameters for the MERGE command. This command will do nothing unless it is followed by a MERGE command. By default the whole file will be merged. The format for the command is:

```
MERGEL/string/
```

The string is structured in the same format as the column specifier, except that is applies to lines.

3.9.2 MERGE COMMAND (MERGE OR M)

The MERGE command causes the contents of a specified file (working or permanent) to be merged into the edit file. The following formats of the command are valid.

```
MERGE/file/;n or MERGE,/file/,/string/;n
```

The file referenced (by the 'file' string) can be a local file or a permanent file, but it cannot be the current edit file or any other reserved file name (refer to appendix A). The merge file is placed after the nth (by default n = 1) line or after the nth line containing the 'string'.

NOTE

Whenever a MERGE command is issued, the data entered in reponse to the most recent ENTER TEXT request is lost. Therefore, a carriage return only in response to an ENTER TEXT request causes no data to be added (refer to the ADD and CHANGE command).

3.9.3 REPLACE COMMAND (REPLACE)

The REPLACE command will replace the edit file in the permanent file catalog of the current username. This command is the same as a REPLACE control card. The format of the command is:

```
REPLACE/pfn/
```

The edit file will be replaced in the permanent file catalog with the file name 'pfn'. If 'pfn' is not used, the edit file name will be used.

If the edit file is a direct file the command is aborted and the following message is issued.

```
ATTEMPTED REPLACE ON DIRECT FILE.
```

If the edit file is too big for an indirect file the following message is issued and the file is not replaced.

```
FILE TOO BIG.
```

Any other error encountered in replacing a file will abort the command and issue the following message.

```
PERMANENT FILE ERROR.
```

3.9.4 SAVE COMMAND (SAVE)

The SAVE command will save the edit file in the permanent file catalog of the current username. This command is identical to the SAVE control card. The format of the command is as follows.

```
SAVE/pfn/
```

The edit file is saved with the file name 'pfn'. If 'pfn' is not present then the edit file name is used. The file is saved in the default manner of the SAVE control card.

If a file of the same name exists, the command is aborted and the following message is generated.

```
FILE ALREADY PERMANENT.
```

If the file is too big for an indirect file the following message is generated.

    FILE TOO BIG.

Any other error condition will abort the command and generate the following message.

    PERMANENT FILE ERROR.

3.9.5 EXAMPLE

The following illustrates the use of the MERGE, SAVE and REPLACE commands.

```
Entry/Response                        Commentary

lnh,f=txt2                            Before    entering    EDT,
                                      time-sharing  commands  are
                                      used  to  list  working  fle
                                      txt2, which is a copy  of  a
                                      permanent file.
THIS FILE IS NAMED TXT2
AND IS A COPY OF AN PERMANENT
FILE OF THE SAME NAME.
READY.
new,a.                                Working   primary   file   a
                                      created,  releasing  working
                                      file txt2.
READY.
edt,a                                 Enter text editor with empty
                                      primary  file  a as the edit
                                      file

 BEGIN TEXT EDITING.
?  add
 ENTER TEXT.
?  /  this file is being built
?  using the text editor add
?  command./
 READY.
?  l;*
  THIS FILE IS BEING BUILT
USING THE TEXT EDITOR ADD
COMMAND.
 -END OF FILE-
?  merge/txt2/                        Merge permanent file txt2 to
                                      end of edit file.
?  l;*
  THIS FILE IS BEING BUILT
```

3.0 EDT COMMANDS
3.9.5 EXAMPLE

```
WITH THE TEXT EDITOR ADD
COMMAND.
  THIS FILE IS NAMED TXT2
AND IS A COPY OF PERMANENT
FILE OF THE SAME NAME.
 -END OF FILE-
?  merge,/txt2/permanent/
```

Merge permanent file txt2 after first occurrence of string /permanent/.

```
?  l*
  THIS FILE IS BEING BUILT
USING THE TEXT EDITOR ADD
COMMAND.
  THIS FILE IS NAMED TXT2
AND IS A COPY OF A PERMANENT
  THIS FILE IS NAMED TXT2
AND IS A COPY OF A PERMANENT
FILE OF THE SAME NAME.
FILE OF THE SAME NAME.
 -END OF FILE-
?  ml/2,3/
```

The next MERGE command will only add the second and third lines of the merge file.

```
?  m,/txt2/*.l*
  THIS FILE IS BEING BUILT
USING THE TEXT EDITOR ADD
COMMAND.
  THIS FILE IS NAMED TXT2
AND IS A COPY OF A PERMANENT
  THIS FILE IS NAMED TXT2
AND IS A COPY OF A PERMANENT
FILE OF THE SAME NAME.
FILE OF THE SAME NAME.
AND IS A COPY OF AN PERMANENT
FILE OF THE SAME NAME.
 -END OF FILE-
?  save/abc/
?  replace/abc/
?  end
 END TEXT EDITING.
READY.
```

3.10 STRING_INCIDENCE_COUNTING

3.10.1 NUMBER COMMAND

The NUMBER command provides a count of lines in a file or a count dependent on the presence of a specified string of characters. The count always begins relative to the search pointer.

3.10.1.1 Line_Mode_Formats_(NUMBER_or_N)

Command	Explanation
NUMBER	Returns a line count from current search pointer value to end-of-file.
NUMBER:/string/	Returns a count of the number of lines in the edit file that contain the string.
NUMBER,/string1/string2/	Returns a count of the number lines that each contain the ellipsis /string1/string2/.

3.10.1.2 String_Mode_Formats_(NUMBERS_or_NS)

Command	Explanation
NUMBERS	Same as NUMBER
NUMBERS,/string/	Returns a count of the number of occurrences of 'string' in the edit file.
NUMBERS/string1/,/string2/	Returns a count of the number of occurrences of the ellipsis /string1/,/string2/.

3.10.2 EXAMPLE

The following illustrates the use of the NUMBER command.

```
Entry/Response                        Commentary

edt,a

 BEGIN TEXT EDITING.
?  I*
*      THIS PROGRAM GENERATES
*      10 RANDOM NUMBERS BETWEEN
*      1 AND 100.
       PROGRAM RANDOM (OUTPUTO
       MAINSD = 5**13
       NEXTSD = 5**17
       H100 = 1./100.
       BIG = 2.**48
       FRAC = H100*BIG
       DO 20 I = 1,10
       INC = 0
       NEXTSD = NEXTSD*MAINSD
10     INC = INC+
       IF ((INC*FRAC).LT.NEXTSD) GO TO 10
       NRAN = INC
       PRINT 30, I, NRAN
30     FORMAT(I2,2X,I4)
20     CONTINUE
       END
 -END OF FILE-
?  number                             Request line   count   from
                                      search  pointer  to  end  of
                                      file.
        20 LINE TO EOF.
?  number/nextsd/                     Request count of  number  of
                                      lines    containing    string
                                      /nextsd/.
       3  OCCURRENCES OF PHRASE FOUND.
?  numbers/nextsd/                    Request count of  number  of
                                      occurrences     of     string
                                      /nextsd/.
       4  OCCURRENCES OF PHRASE FOUND.
?  s10
?  n                                  Request line    count   from
                                      search  pointer  to  end  of
                                      file.
       10 LINES TO EOF.
?  n/inc/,/0/                         Request line count of number
                                      of lines containing  elipsis
```

```
                                          /inc/,/0/.
          3  OCCURRENCES OF PHRASES FOUND.
   ?  end
    END TEXT EDITING.
   READY.
```

3.11 COMMAND BUFFERS

Command buffers are a means of executing a command or a series of commands a number of times without re-entering the commands. This is useful, for example, when proof reading a text file. The commands:

```
S15.L15
```

can be executed many times to read the file and stop to allow the user to read the current fifteen lines.

EDT has two types of command buffers. It has implicit and explicit command buffers. The implicit buffer cannot be executed more than once at a time, while the explicit buffer can be executed many times at once.

There is a restriction using these buffers. If an explicit command buffer is executed within an explicit command buffer, the rest of the first command buffer is lost and the editor will go into an infinite loop. A program interrupt will stop the execution of the buffer. An implicit command buffer cannot be called within an explicit command buffer. If an explicit command buffer is executed within an implicit command buffer (or within a line of input), the rest of the implicit buffer (or input) is executed after the command buffer (unless it is interrupted).

3.11.1 IMPLICIT COMMAND BUFFER

The implicit command buffer is defined as the last line of input read and executed by EDT. This command buffer can be executed by entering a global terminator and a carriage return. The buffer will be executed once for every time a terminator and carriage return is entered.

3.11.2 EXPLICIT COMMAND BUFFERS

Explicit command buffers are those that are set up by the CBd commands and executed by the EXd commands (d equals A, B, or C). There are three buffers (A, B, and C). Caution must be used in specifying the correct 'd' with the CBd and EXd command, otherwise the wrong buffer may be executed.

3.11.2.1 CBd COMMAND (CBA or CBB or CBC)

This command will save one of the three command buffers (depending on 'd') until the buffer is overwritten. A second call to any buffer will overwrite it. The command has the following format.

Command	Explanation
CBA or CBB or CBC	Display the contents of the command buffer.
CBd/string/	Place the contents of 'string' into the command buffer.

The command buffer is limited to 150 characters (in normal mode). If more than this limit is entered the command buffer will not be saved and the following message will be issued.

```
COMMAND BUFFER TOO LONG.
```

NOTE

'string' must be legal EDT commands. Any delimiters used for commands within 'string' cannot be the same delimiters that 'string' has surrounding it. The first character of 'string' cannot be a local command terminator. It must use the current local terminator.

3.11.2.2 EXd COMMAND (EXA or EXB or EXC)

The EXd command will execute the command buffer. The format of the command is as follows.

```
EXA;n or EXB;n or EXC;n
```

The command buffer desired will be executed n (by default one) times.

If the CBd command has not been used for buffer 'd' and the EXd command is issued the command will be aborted and the following message will be issued.

```
COMMAND BUFFER EMPTY.
```

3.11.3 EXAMPLE

The following example illustrates the use of the implicit command buffer, CBn and EXn commands.

```
Entry/Response                    Commentary

edt,aaa
 BEGIN TEXT EDITING.
?  l4.s4
    THE COMMAND BUFFER
WILL STORTEN THE NUMBER OF
ENTRIES THAT THE USER NEEDS
TO MAKE.
?  .                              The terminator will repeat
                                  the last input line
                                  (l4.s4).
ABC  ABC
      ABC   ABC
  ABC ABC ABC
 ABC       ABC
 -END OF FILE-
?  s-4
?  cba/rs?abc?def?.s/             Place string /rs?abc?def?.s/
                                  in command buffer a.
?  cba                            List contents of command
                                  buffer a.
 RS?ABC?DEF?.S
?  exa.l*                         Execute command buffer a
                                  once and list the file.
      ABC   ABC
  ABC ABC ABC
 ABC       ABC
 -END OF FILE-
?  exa3                           Execute command buffer a
                                  three times.
 -END OF FILE-
?  s-4.l*
DEF  ABC
      DEF   ABC
  DEF ABC ABC
 DEF       ABC
 -END OF FILE-
?  end
 END TEXT EDITING.
READY.
```

3.12 MISCELLANOUS COMMANDS

3.12.1 BREAK COMMAND (BREAK)

The default mode of EDT is to allow the editor to process program interrupts. Thus, the user cannot abort the editor with an interrupt. The break command allows the user to change the mode. The command format is:

```
BREAK,/mode/
```

where 'mode' can be ON (allows interrupts to abort the editor) or OFF (allows the editor to process interrupts).

3.12.2 ECHO COMMAND (ECHO)

The ECHO command allows the user to print out the commands that EDT is executing as it starts to execute them. This is useful when an alternate input file is used or in a batch mode edit. The command format is:

```
ECHO/mode/
```

where 'mode' is ON (print the commands) or OFF (do not print the commands).

3.12.3 ASCII COMMAND (ASCII)

The ASCII command changes the character mode from the present mode to ASCII mode. This will allow upper and lower case characters to be used. The character mode is returned to the initial mode when EDT is finished. The command format is:

```
ASCII
```

NOTE

If commands follow the ascii command on the same line the following three characters must not appear in the line (because of a change in the display code in ascii mode). The characters are ':', '^' and '@' signs.

3.12.4 NORMAL COMMAND (NORMAL)

The NORMAL command changes the character mode from the present mode to NORMAL mode. This will allow only upper case character to be used. The character mode is returned to the initial mode when EDT is finished. The command format is as follows.

```
NORMAL
```

<u>NOTE</u>

If there are commands following a NORMAL command, these commands will not be interpetable because they are in ASCII mode. This applies to the command buffers as well as commands strung together on the input line. The NORMAL command must be the last command before EDT requests input.

3.12.5 TERM COMMAND (TERM)

The TERM command redefines the global command terminator (by default a period). The is used by the implicit command buffer as well as the default terminator. The command format is:

```
TERM/char/
```

where 'char' is the new global terminator.

3.12.6 EXAMPLE

The following example illustrates the ASCII, BREAK, ECHO, NORMAL and TERM commands.

```
Entry/Response                         Commentary
edt,aaa
 BEGIN TEXT EDITING.
?  echo,/on/                           turn on the command echo.
?  l;*
L;*
     THESE COMMANDS DO NOT
AFFECT THE EDIT FILE, BUT
THEY TO ASSIST THE USER IN
USING EDT.
```

```
   ?  break]off].l;*                  Turn off           program
                                      interruption    (this    is
                                      default).
    BREAK]OFF]
    L;*
       THESE COMMANDS     <--- the break key is pressed.


   ?  echo/off/.break/on/             The prompt  comes  back  and
                                      EDT continues.
    ECHO/OFF/
   ?  l*
       THESE COMMANDS     <--- the break key is pressed.
   *TERMINATED*                       EDT  has stopped running and
                                      the operating system  is  in
                                      control of the system.
   get,aaa.                           Get the file.  Aborting  EDT
                                      may corrupt the edit file.

   READY.
   edt,aaa.
    BEGIN TEXT EDITING.
   ?  term/)/
   ?  l;*)f
       THESE COMMANDS DO NOT
   AFFECT THE EDIT FILE, BUT
   THEY DO ASSIST THE USER IN
   USING EDT.
   ?  .r.term?.?                      use the  local terminator to
                                      override     the      global
                                      terminator.
   ?  ascii                           Change the character mode to
                                      ASCII.
   ?  a*.l*
    ENTER TEXT.
   ?/     The ASCII command does affect input
   ?  inasmuch as lower case characters can
   ?  be used./
    READY.
       THESE COMMANDS DO NOT
   AFFECT THE EDIT FILE, BUT
   THEY DO ASSIST THE USER
   USING EDT.
       The ASCII command does affect input
   inasmuch as lower case characters can
   be used.
    -END OF FILE-
   ?  normal
   ?  a*.l*.ascii
    ENTER TEXT.
```

```
     the normal command changes all
characters to upper case.
 READY.
     THESE COMMANDS DO NOT
AFFECT THE EDIT FILE, BUT
THEY DO ASSIST THE USER IN
USING EDT.
     The ASCII command does affect input
inasmuch as lower case characters can
be used.
     THE NORMAL COMMAND CHANGES ALL
CHARACTERS TO UPPER CASE.
 -END OF FILE-
?  end
 END TEXT EDITING.
```

3.13 TERMINATING EDIT SESSION

3.13.1 END COMMAND (END)

The END command terminates text editing (that is, exits from EDT program control) and returns control to the subsystem control language. The command format is:

```
END
```

The system will respond (when the input file is not 0)

```
END TEXT EDITING.
```

3.13.2 STOP COMMAND (STOP)

The user can also end text editing by typing STOP after the execution of an edit command. This immediately terminates the edit session and the terminal is no longer under EDT control. In this case, the text file contents are unpredictable, and all output files can be lost. The error flag is not set.

This method of termination would be used in situations where the contents of files are to be examined but are not required after the edit session.

3.13.3 ABORT COMMAND (ABORT)

The ABORT command terminates the editor just as the STOP command does, and it also sets the error flag. Thus, if the user enters EDT from a procedure file, EDT may be ended by STOP or ABORT. STOP will allow the procedure file to carry on, while ABORT will cause the procedure file to abort.

A1.0 EDT MESSAGES

EDT supplies the following messages. There messages indicate a condition that prevents processing of a command or are issued in the course of normal edit operation.

MESSAGE	SIGNIFICANCE
ATTEMPTED REPLACE ON DIRECT FILE.	A REPLACE command was used on a direct access permanent file. ACTION - Make edit file a local file before editing.
BEGIN TEXT EDITING.	Informative command indicating the editor is ready to begin accepting commands. ACTION - None.
COMMAND BUFFER TOO LONG.	A series of commands of more than 150 characters was used in a CBd command. ACTION - re-enter the commands with less than 150 characters.
COMMAND CONTINUE?	EDT inquiry as to whether or not an interrupted command should continue to be processed. ACTION - Respond with YES (Y) or NO (N).
CONTROL CARD ERROR.	An illegal or invalid parameter was specified on the control card. ACTION - Retry using correct parameters.
DISREGARD PREVIOUS TEXT?	EDT inquiry as to whether or not the text that has been entered in response to a text-entering command should be retained or discarded.

	ACTION - Respond with YES (Y) or NO (N).
-END OF FILE-	Informatative message indicating that the text file is positioned at end of file or end of file was encountered during a LIST or FIND command. ACTION - None.
END TEXT EDITING.	Informative message indicating termination of EDT session. ACTION - None.
ENTER TEXT.	Requests entry of new or replacement text for ADD(S), or CHANGE(S) command. ACTION - Enter line(s) of text to be processed (enclosed in delimiters).
ENTER TEXT FILE NAME.	Text file name has not been passed with EDT call. ACTION - Enter text file name.
FILE ALREADY PERMANENT.	A SAVE command was attempted with a file name of an existing file. ACTION - Use a different file name or a REPLACE command.
FILE ALREADY TRUNCATED.	A LENGTH command was attempted on a file which has already been truncated by a LENGTH command. ACTION - Restore the file to its original length and then use the LENGTH command.
FILE AT LINE NUMBER n.	Text file is currently positioned at line number n. ACTION - None.
FILE NOT LONG ENOUGH.	A LENGTH command gave line

	numbers that were not in the file. ACTION - Retry the LENGTH command with correct line numbers.
FILE NOT TRUNCATED.	A LENGTH restore command was attempted on a file that has not been truncated. ACTION - Truncate file (with LENGTH command) before trying to restore the file.
FILE TOO BIG.	A SAVE or REPLACE was attempted on a file that was too big for the indirect file size. ACTION - Reduce the size of the file or enter EDT with a direct file attached in write mode or make file a direct file when finished EDT.
ILLEGAL COMMAND.	The command name entered is not a valid command. ACTION - Ensure that legal command is entered.
ILLEGAL DELIMITER.	An illegal delimiter was used in response to the ENTER TEXT request from a local or remote batch job. ACTION - Retry using correct delimiter.
ILLEGAL DELIMITER - REENTER TEXT.	An invalid delimiter was used in reponse to the ENTER TEXT request from a time-sharing job. ACTION - Retry using correct delimiter.
ILLEGAL FILE NAME.	The file name passed with the text editor MERGE SAVE or REPLACE command is illegal. ACTION - Specify legal file name.

INTERRUPT AT LINE n.	Informative message indicating the current position of an interrupted command. ACTION - None.
INTERRUPT DISABLED.	An informative message indicating that a program interrupt will not be processed by EDT, but will terminate EDT. ACTION - None.
INTERRUPT ENABLED.	An informative message indicating that a program interrupt will be processed by EDT. ACTION - None.
LINE TOO LONG.	A line lengthened by a DEOL, CEOL or AC command became longer than 150 characters. ACTION - Reduce line length to less than 150 characters.
LOCAL FILE ERROR.	The LOCAL command encountered problems attempting to make the string buffer local. ACTION - The string buffer cannot be made local by the NOS. Check for the reason outside of EDT.
MERGE ERROR, SECONDARY FILE EMPTY.	One of the following conditions exist. (1) The file to be merged with the edit file is empty. (2) The file to be merged does not exist. ACTION - Verify merge file.
PERMANENT FILE ERROR.	A NOS file error has occurred in a SAVE or REPLACE command. ACTION - Check outside of EDT for the reason for the aborted file request.

PHRASE NOT FOUND.	The specified search string was not found. ACTION - None.
READY.	Informative message indicating next command can be entered. ACTION - NONE.
RESERVED FILE NAME.	Operation attemped on a file name reserved by EDT. ACTION - Choose a non-reserved file name.
SEARCH COLUMNS ARE a b	Informative message indicating the current global search columns. ACTION - None.
STRING BUFFER HAS NOT BEEN USED.	A LOCAL command was attempted when the string buffer (used with EXTRACT) has not been used. ACTION - Use EXTRACT before LOCAL.
TAB : STOPS NONE.	No tab stops are currently established. ACTION - None.
TAB c STOPS t1 t2 ... tn	EDT tab character 'c' and stops issued in response to LISTAB command. ACTION - None.
WARNING: FILE HAS BEEN TRUNCATED. DO YOU WANT TO END ?	A warning message when the END command is issued to inform the user that the text file has been truncated with the LENGTH command. ACTION - Respond YES (Y) or NO (N).
n LINES TO EOF.	Informative message indicating number of lines in text file before end-of-file. ACTION - None.

A1.0 EDT MESSAGES

n LINES TO INTERRUPT.	Informative message indicating the number of lines that were processed before the user terminated a command. ACTION - None.
n OCCURRENCES OF PHRASE FOUND.	End-of-file was encountered before number of iterations specified in command were completed. ACTION - None.
command SYNTAX ERROR.	Improper syntax used with text editor command. ACTION - Retry using correct syntax.

Table of Contents