60436100

CONTROL DATA CORPORATION

NOS VERSION 1
TEXT EDITOR
REFERENCE MANUAL

CONTROL DATA®
CYBER 170 SERIES
MODELS 172, 173, 174, 175
CYBER 70 SERIES
MODELS 72, 73, 74
6000 SERIES
COMPUTER SYSTEMS

SUMMARY OF AVAILABLE COMMANDS AND FORMATS

60436100

CONTROL DATA CORPORATION

NOS VERSION 1
TEXT EDITOR
REFERENCE MANUAL

CONTROL DATA®
CYBER 170 SERIES
MODELS 172, 173, 174, 175
CYBER 70 SERIES
MODELS 72, 73, 74
6000 SERIES
COMPUTER SYSTEMS

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released.
(9-30-74)	
B	Eliminates the EXTRACTS and REPLACES command forms, documents additions to the
(5-1-75)	system command EDIT, and corrects various technical errors.
C	Manual revised to reflect NOS 1.1. New features include an expanded EDIT command, termi-
(3-8-76)	nal interrupt capability, and batch usage of the Text Editor. Typographical corrections have
	been made and several descriptions have been clarified. This edition obsoletes all previous
	editions.

Publication No.
60436100

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 Lexington Avenue North
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

Printed in the United States of America

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	SFC †	REV
F. Cover		-
Summary		-
Title Page		-
ii		C
iii		C
v		C
vii		C
viii		C
1-1		C
1-2		C
2-1		C
2-2		C
2-3		C
2-4		C
2-5		C
2-6		C
2-7		C
2-8		C
2-9		C
2-10		C
3-1		C
3-2		C
3-3		A
3-4		C
3-5		C
3-6		C
3-7		C
3-8		C
3-9		C
3-10		C
3-11		C
3-12		C
3-13		C
3-14		C
3-15		C
3-16		C
3-17		C
3-18		C
3-19		C
3-20		C
3-21		C
3-22		C
3-23		C
3-24		C
3-25		C
3-26		C
3-27		C
3-28		C
3-29		C
3-30		C
3-31		C
3-32		C
3-33		C
3-34		C
3-35		C
A-1		C
A-2		C
A-3		C
B-1		C
B-2		C
B-3		C
B-4		A
B-5		C
B-6		C
B-7		C
B-8		A
B-9		C
C-1		C
C-2		C
C-3		C
D-1		C
D-2		C
Index-1		C
Index-2		C
Comment Sheet		C
Reply Env.		-
B. Cover		-

†SFC Software Feature Change

AA 5935

PREFACE

The Text Editor (also known as the EDIT program) is a part of the Network Operating System (hereinafter called NOS or the system) for CONTROL DATA® CYBER 170 Series, Model 172, 173, 174, and 175 Computer Systems, CDC CYBER 70 Series, Model 72, 73, and 74 Computer Systems, and CDC 6000 Series Computer Systems. Its purpose is to effect character-oriented data manipulations from a remote terminal.

This manual contains the information a user must have to use the Text Editor.

Section 1 introduces the Text Editor and summarizes its general capabilities.

Section 2 defines fundamental concepts and terminology inherent in Text Editor usage.

Section 3 contains descriptions of each of the EDIT commands, including pertinent examples of each general type of command. Certain commands are mutually related and these relationships are depicted in the examples.

The Text Editor user who needs basic information pertaining to system access and file management can refer to appendices A and B. For additional information about NOS software, refer to the current versions of the following manuals.

Control Data Publications	Publication No.
NOS Reference Manual Volume 1	60435400
NOS Reference Manual Volume 2	60445300
NOS Time-Sharing User's Reference Manual	60435500
NOS Terminal User's Instant Manual	60435800

This product is intended for use only as described in this document. Control Data Corporation cannot be responsible for the proper functioning of undescribed features or undefined parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

INTRODUCTION 1

GENERAL

The Text Editor (EDIT) performs data manipulations on a file specified in a time-sharing session or in a batch job. From a terminal, it is an interactive package; that is, the user enters a command, EDIT interprets the command and executes it, after which the user can enter another command to be executed, and so forth. This manual is presented from a time-sharing orientation. Batch usage is outlined in appendix D.

TEXT EDITOR CAPABILITY

Using Text Editor commands, the user can manipulate edit file data in the following ways (appropriate command words are shown in parentheses).

- Print a file, either in part or in entirety (LIST, FIND)
- Erase information from a file (DELETE, BLANK)
- Add information to a file (ADD, INSERTS)
- Replace information in a file with other information (CHANGE, RS)
- Move information to and from a temporary holding area for subsequent insertion (EXTRACT, CLEAR)
- Combine the contents of two files (MERGE)
- Obtain a count that reflects the number of times a specified combination of characters occurs in the edit file (NUMBER)
- Determine the line number of the file at which Text Editor is currently positioned (LINE)
- Direct Text Editor activity to a specific area of the edit file (SET, RESET, FIND)
- Control edit file and page format (LENGTH, WIDTH, ALIGN, DEFTAB, LISTAB, TAB)
- Terminate the editing session (END)

EDIT OPERATIONS

An edit command consists of a command word, followed optionally by a string specification and an n parameter.

The Text Editor command repertoire allows three basic types of operations.

1. Line mode operations are addressed to one or several entire lines of text in the edit file.

2. String mode operations are addressed to a sequence of characters, as indicated by a string specification that follows the command word. A string mode command word always ends with an S. A string mode command with an empty string specification has exactly the same effect as a line mode command.

3. Edit control commands are not addressed to specific lines or strings of text. In general, they perform such necessary functions as search-pointer control, format control, large-scale file manipulations, and exit from the Text Editor program.

CONVENTIONS

The symbol ⏎ is used throughout to denote the carriage return key (or its equivalent).

In the examples, the terminal printout has occasionally been expanded to accomodate the commentaries.

TEXT EDITING CONCEPTS 2

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

ENTERING TEXT EDITOR

After log in is completed, the user enters Text Editor by typing in the EDIT command. The full form of this command is:

EDIT, lfn_1, m, lfn_2, lfn_3

or

EDIT, FN=lfn_1, M=m, I=lfn_2, L=lfn_3

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

lfn_1	Name of the file to be edited. Default is the primary file.
m	Mode of file processing N Normal AS ASCII For a time-sharing session, default is whatever mode the terminal was in when Text Editor was entered. For a batch job, default is normal.
lfn_2	The file from which the edit commands are to be read. Default is input from the terminal.
lfn_3	The file on which the output is to be written. The default is output at the terminal.

The user will frequently want to use default versions of the EDIT command. Thus the entry

EDIT ⓒⓡ

calls Text Editor 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

EDIT, lfn ⓒⓡ

calls Text 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 set mode.

After the EDIT command is entered, the system replies:

```
  BEGIN TEXT EDITING.
?
```

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

The Text Editor may be called from any of the time-sharing subsystems. It can also be called from a local or remote batch job. Use of the Text Editor from a batch job is covered in appendix D.

CAUTION

Text Editor operates on a single record only. If it is entered with a multirecord file, all but the first record is lost (refer to the NOS Time-Sharing User's Reference Manual, section 3, File Sorting).

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

It is possible to enter Text Editor with an empty file and develop it during the edit session. Refer to Adding and Building Text in section 3.

EDIT FILE

The Text 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 Text Editor with the EDIT 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.

CAUTION

Editing a read-only file may cause unpredictable results.

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 EDIT 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).

EDIT COMMANDS (GENERAL FORMAT)

Each editing operation on the edit file is specified by an edit command. An edit command may consist of the following four elements.

1. Command word
2. String specification, consisting of zero, one, or two string fields
3. n parameter, consisting of a positive integer
4. Comment

Blanks have no significance in the Text Editor command language. They are permitted between the letters of the command word and between the components of the command.

The string specification and/or the n parameter are not used with certain commands, and commentary is optional with all commands.

The general form of an edit command is:

< cmwd> < strdef> < n> < comment>

where:

< cmwd>
: Any EDIT command word, or short form thereof, as listed in Command Words

< strdef>
: One of the following string definitions.

:< string> , < string>

:< string>

omitted

<string> consists of a nonzero number of alphanumeric characters, bounded on each end by a nonblank character (called a delimiter). In most commands the string identifies the part of the edit file being sought, although in several commands the string has a special purpose (for example, refer to MERGE command). The delimiters on each end must be the same character, must not be the character $ or blank, and cannot be used within the string.

< n>

One of the following.

;n

omitted

n is an integer or an asterisk. The integer is positive, except that a negative n is permitted if < cmwd> is SET. An asterisk implies an n value equal to the number of text lines or appearances of a string in the edit file from the search pointer to the end of the file.

< comment>

A comment, if included, consists of a dollar sign ($), followed by any sequence of characters ending with the CR . It has no effect on the operation of the command.

The entire command, including comment, must be on one line. Pressing the carriage return signifies the end of the command. Only one command is permitted on a single line.

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 the edit file as the basic unit of operation. In a string mode command, all operations are performed with a character string 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.

COMMAND WORDS

The command word determines the operation to be performed. The EDIT 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

ALIGN	(AL)	LISTAB	(LT)
CLEAR	(CL)	MERGE	(M)
DEFTAB	(DT)	RESET	(R)
END		SET	(S)
LENGTH		TAB	(T)
LINE	(LN)	WIDTH	(W)

STRINGS AND DELIMITERS

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

1. In the <strdef> field of a Text Editor command
2. 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.

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 prior to a carriage return), Text Editor interprets the character as the closing delimiter and the ENTER TEXT request is terminated. EDIT tests for the closing delimiter only after a carriage return.

Use of the delimiter character within the <strdef> field of a Text Editor command is not allowed and if used EDIT responds:

```
<cmwd>      SYNTAX ERROR.
```

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

Correct String Definition

```
/ABCDE/

/THE FORMAT OF/

BALWAYS IS B

? 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)
```

CAUTION

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

SEARCH STRING PARAMETER

The search string parameter of an EDIT command indicates to the Text 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 or an ellipsis.

SINGLE PHRASE SEARCH STRING

In a single phrase search string, the entire string of 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.

ELLIPSIS SEARCH STRING

An ellipsis search string specification consists of two delimited bracket strings, separated by a comma. 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/

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.

AS 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/

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/

identifies the underlined string properly.

SPECIAL STRING FIELDS

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 three 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
- Merge file name in a MERGE command

n PARAMETER

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

- 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 length of a file in lines or 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 applicable for the commands RESET, LINE, LISTAB, CLEAR, NUMBER(S), DEFTAB, and END. Negative values of n are allowed only in a SET 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.

DOCUMENTARY COMMENTS

To annotate the editing session (possibly for review purposes), append a dollar sign to any or all commands and follow the dollar sign with commentary information. The comment is ignored by the editor.

STRING BUFFER

The string buffer is a temporary storage area 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 may then 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:

```
$ CR
```

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.

ENTER TEXT REQUEST

The Text Editor issues an ENTER TEXT request in response 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 nonblank character entered in response 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 nonblank character except a dollar sign ($). If a blank or a dollar sign is entered as a delimiter from an interactive job, EDIT responds with:

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

For a local or remote batch job, EDIT 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 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).

PROCESSING TERMINAL INTERRUPTS

The time-sharing user may control his edit session through use of terminal interrupts. He exercises these interrupts under three circumstances.

- While output is being transmitted to his terminal. The transmission of output to a terminal is terminated on an ASCII Code terminal by pressing the BREAK, I, or S key; on a correspondence code terminal by pressing the ATTN key. One of the main uses of this type of interrupt is the termination of unwanted output from execution of a LIST command.

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

  Typing STOP after entering text in response to an ADD or CHANGE command terminates the command and 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.

- While the system is processing a command he has entered. If an edit command is in execution, the user may contingently terminate execution by typing STOP. The system gives the output status of the command in execution and then prints the enquiry

  ```
  CONTINUE COMMAND ?
  ```

  If the user types YES after the question mark, processing continues; if he types NO, processing terminates.

NOTE

Typing STOP after the execution of an edit command immediately terminates the edit session. Refer to Terminating Edit Session for additional information.

EDIT COMMANDS 3

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 the 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 similar commands.

ENTERING COMMANDS

All Text 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 command is typed and ⓒⓡ is pressed, the Text Editor either processes the command immediately or requests additional information. In general, each edit command operation is performed relative to the current position of the search pointer. Appendix C contains a summary of all Text Editor messages and requests. For batch origin jobs, commands are entered from an input file (refer to appendix D).

TEXT LISTING AND SEARCH POINTER CONTROL

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 string, or a set of contiguous lines.

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.

LINE MODE FORMATS (LIST OR L)

Command	Explanation
LIST	Prints the line of text specified by the search 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 current position of search pointer.
LIST:/string/;n	Prints the first n lines containing the string (n can equal *, in which case all lines in the edit file that contain the string are printed).
LIST:/string1/,/string2/	Prints the line or group of lines containing the ellipsis string1 through string2.
LIST:/string1/,/string2/;n	Prints the first n occurrences of lines or groups of lines containing the ellipsis string1 through string2.

STRING MODE FORMATS (LISTS OR LS)

Command	Explanation
LISTS	Same as LIST.
LISTS;n	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 occurrences of the string of characters specified by the ellipsis /string1/,/string2/.

SEARCH POINTER CONTROL (SET AND RESET)

EDIT initially locates the search pointer at the first line of the edit file. With the SET command, the search 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, regardless of its former position. Activity on the edit file always begins at the current search pointer setting.

SET COMMAND (SET OR S)

The following are the four forms of the SET command.

Command		Explanation
SET		Advances the search pointer one line relative to its current setting.
SET;n SET;-n	or	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/		Moves 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		Moves the search pointer forward from its current setting to the beginning of the line containing the nth occurrence of the search string; if there are less than n occurrences, 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 an n parameter is implied.

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

Using a search string without an n parameter moves the search pointer from its current setting forward to the line containing the first occurrence of the search string.

RESET COMMAND (RESET OR R)

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

```
RESET
```

Operand fields are not used with the RESET command.

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 Text 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.

LINE MODE FORMATS (FIND OR F)

Command	Explanation
FIND	Advances the search pointer one line and lists the line.
FIND;n	Advances the search pointer n lines and lists the line indicated by the new value of the search pointer.
FIND:/string/ FIND:/string/;n	Advances the search pointer to the nth line that contains at least one occurrence of /string/, and lists the line.
FIND:/string1/, /string2/ FIND:/string1/, /string2/;n	Advances the search pointer from its current position to the nth line that contains the beginning of the ellipsis search string. If the search string is multiline, all lines containing some part of the nth occurrence of /string1/, /string2/ are listed, and the search pointer is set to the line in which the nth occurrence begins.

STRING MODE FORMATS (FINDS OR FS)

Command	Explanation
FINDS	Same as FIND.
FINDS;n	Same as FIND;n.
FINDS:/string/ FINDS:/string/;n	Advances the search pointer to the line containing the nth occurrence of /string/ and lists the string.
FINDS:/string1/, /string2/ 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.

SEARCH POINTER CONTROL (SET AND RESET)

EDIT initially locates the search pointer at the first line of the edit file. With the SET command, the search 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, regardless of its former position. Activity on the edit file always begins at the current search pointer setting.

SET COMMAND (SET OR S)

The following are the four forms of the SET command.

Command		Explanation
SET		Advances the search pointer one line relative to its current setting.
SET;n SET;-n	or	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/		Moves 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		Moves the search pointer forward from its current setting to the beginning of the line containing the nth occurrence of the search string; if there are less than n occurrences, 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 an n parameter is implied.

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

Using a search string without an n parameter moves the search pointer from its current setting forward to the line containing the first occurrence of the search string.

RESET COMMAND (RESET OR R)

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

```
RESET
```

Operand fields are not used with the RESET command.

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 Text 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.

LINE MODE FORMATS (FIND OR F)

Command	Explanation
FIND	Advances the search pointer one line and lists the line.
FIND;n	Advances the search pointer n lines and lists the line indicated by the new value of the search pointer.
FIND:/string/ FIND:/string/;n	Advances the search pointer to the nth line that contains at least one occurrence of /string/, and lists the line.
FIND:/string1/, /string2/ FIND:/string1/, /string2/;n	Advances the search pointer from its current position to the nth line that contains the beginning of the ellipsis search string. If the search string is multiline, all lines containing some part of the nth occurrence of /string1/, /string2/ are listed, and the search pointer is set to the line in which the nth occurrence begins.

STRING MODE FORMATS (FINDS OR FS)

Command	Explanation
FINDS	Same as FIND.
FINDS;n	Same as FIND;n.
FINDS:/string/ FINDS:/string/;n	Advances the search pointer to the line containing the nth occurrence of /string/ and lists the string.
FINDS:/string1/, /string2/ 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.

LINE COMMAND (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.

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

Entry/Response	Commentary
EDIT	
BEGIN TEXT EDITING.	
? LIST;*	List to end of file.
00010 PROGRAM RANDNUM	
00020 MAINSEED = 5**13	
00030 NEXTSEED = 5**15	
00040 HUNDRETH = 1./100.	
00050 BIGNUMBR = 2.**48	
00060 FRACTION = HUNDRETH*BIGNUMBR	
00070 IFRAC = INT(FRACTION)	
00080 DO 20 I = 1,10	
00090 INCRMENT = 0	
00100 NEXTSEED = NEXTSEED*MAINSEED	
00110 10 INCRMENT = INCRMENT+1	
00120 IF((INCRMENT*IFRAC).LT.NEXTSEED) GO TO 10	
00130 NEWNUM = INCRMENT	
00140 20 CONTINUE	
00150 END	
-END OF FILE-	
? SET;8	Advance search pointer forward eight lines from current setting.
? L $ LIST LINE.	
00090 INCRMENT = 0	
? LINE	Determine position of search pointer relative to beginning of file.
FILE AT LINE NUMBER 9.	
? RESET	Reset search pointer to beginning.
? LIST:ZNEXTSEEDZ;2	List first two occurrences of lines containing string /NEXTSEED/.
00030 NEXTSEED = 5**15	
00100 NEXTSEED = NEXTSEED*MAINSEED	
? LISTS:*NEXTSEED*;3	List first three occurrences of string /NEXTSEED/.
NEXTSEED	
NEXTSEED NEXTSEED	

Entry/Response	Commentary
`? FIND;8`	Advance search pointer and list line.
`00090 INCREMENT = 0`	
`? L:/20/,/EN/;2`	List lines containing first two occurrences of ellipsis string /20/,/EN/.
`00120 IF((INCRMENT*IFRAC).LT.NEXTSEED) GO TO 10`	
`00140 20 CONTINUE`	
`00150 END`	
`? LISTS:/20/,/EN/;2`	List first two occurrences of ellipsis string /20 /,/EN/.
`  20 IF((INCRMEN`	
`  20 CONTINUE`	
`00150 EN`	
`? FIND:IHUNDREDTHI`	
` PHRASE NOT FOUND.`	
`? RESET`	Advance search pointer from current setting to line containing string /HUNDRETH/. Note that command is always relative to search pointer. Advance search pointer to line containing second occurrence of string /IFRAC/ and list string.
`? FIND:IHUNDRETHI`	
`00040 HUNDRETH = 1./100.`	
`? FS:/IFRAC/;2`	
`  IFRAC`	
`? LN`	
` FILE AT LINE NUMBER  12.`	
`? S;-5`	Move search pointer back five lines. List two contiguous lines.
`? LIST;2`	
`00070 IFRAC = INT(FRACTION)`	
`00080 DO 20 I = 1,10`	
`? RESET`	Reset search pointer to beginning of edit file. List first two occurrences of ellipsis string /13/,/40/.
`? LS:/13/,/40/;2`	
`  13`	
`00030 NEXTSEED = 5**15`	
`00040`	
`  130 NEWNUM = INCRMENT`	
`00140`	
`? LINE`	
` FILE AT LINE NUMBER  1.`	
`? FS:/13/,/40/;2`	Advance search pointer to line containing beginning of second occurrence of ellipsis string /13/,/40/ and list string.
`  130 NEWNUM = INCRMENT`	
`00140`	
`? LINE`	
` FILE AT LINE NUMBER  13.`	
`? S;-12`	Set search pointer back 12 lines.
`? LINE`	
` FILE AT LINE NUMBER  1.`	
`? S`	Advance search pointer one line.
`? LINE`	
` FILE AT LINE NUMBER  2.`	
`? FIND:/FRAC/,/ON/;3`	Advance search pointer to line containing beginning of third occurrence of ellipsis string /FRAC/,/ON/ and list lines.
`00120 IF((INCRMENT*IFRAC).LT.NEXTSEED) GO TO 10`	
`00130 NEWNUM = INCRMENT`	
`00140 20 CONTINUE`	
`? LN`	
` FILE AT LINE NUMBER  12.`	
`? SET:/END/;2`	Advance search pointer to second occurrence of string /END/.
`  1 OCCURANCES OF PHRASE FOUND.`	
`? L`	
`00150 END`	
`? END`	
` END TEXT EDITING`	

ADDING AND BUILDING TEXT

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

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 three 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 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 (CR) only. This causes the data entered in response to the most recent ENTER TEXT request to be added.

NOTE

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

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

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.

STRING MODE FORMATS (ADDS OR AS)

Command	Explanation
ADDS	Same as ADD.
ADDS;n	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.

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 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 of n occurrences of string1, beginning at the search pointer. Note that /string1/,/string2/ specification is not an ellipsis search string in this command.

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

Entry/Response	Commentary
`EDIT,FILE1`	Text Editor is called, creating empty working file FILE1.

`BEGIN TEXT EDITING.`
`? ADD`
`ENTER TEXT`
`? / THE ADD COMMAND CAN BE VERY`
`? USEFUL WHEN CREATING A TEXTUAL`
`? FILE. /`
`READY.`
	The file is built using the ADD command.

`? ADDS:+FILE. +`
`ENTER TEXT.`
`? /IN FACT, IT IS ONE`
`? OF THE FEW METHODS THAT CAN BE`
`? USED TO BUILD A DIRECT ACCESS`
`? FILE./`
`READY.`
	Text is added immediately after the first occurrence of the string /FILE. /.

`? LIST;*`
`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`
`FILE.`
`-END OF FILE-`
	List to end of file.

`?INSERTS:ACCESS/,/PERMANENT/`
`? FIND;4`
`USED TO BUILD A DIRECT ACCESS PERMANENT`
	The string / PERMANENT/ is inserted after the first occurrence of the string /ACCESS/.

`? AS:,FILE,`
`ENTER TEXT`
`? T,PROVIDING IT IS THE EDIT FILET`
`READY.`
	Text is added directly after the first occurrence of the string /FILE/.

`? A;*`
`ENTER TEXT`
`? = IT IS ALSO USEFUL WHEN ADDING`
`? TEXT TO A PREVIOUSLY EXISTING FILE.=`
`READY.`
	Text is added to the end of the file.

`? RESET`
`? L;*`
`THE ADD COMMAND CAN BE VERY`
`USEFUL WHEN CREATING A TEXTUAL`
`FILE. IN FACT, IT IS ONE`
`OF THE FEW METHODS THAT CAN`
`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-`
	Reset search pointer to beginning of file and list to end of file.

`? A:/EXISTING/`
`ENTER TEXT.`
`?/ LATER IT WILL BE DEMONSTRATED HOW TO`
`? USE THE ADD COMMAND TO REMOVE TEXT`
`? FROM THE STRING BUFFER./`
	Add text after the line containing the string /EXISTING/.

Entry/Response	Commentary
READY.	
? S;7	Advance search pointer seven lines.
? L;4	List four lines.
TEXT TO A PREVIOUSLY EXISTING FILE.	
LATER IT WILL BE DEMONSTRATED HOW TO	
USE THE ADD COMMAND TO REMOVE TEXT	
FROM THE STRING BUFFER.	
? R	
? S:/EDIT FILE/	
? ADDS	Same as ADD command.
ENTER TEXT	
? I IT IS ESPECIALLY USEFUL WHEN	
? ADDING TEXT IN THE BODY OF A FILE.I	
READY.	
? L;3	List three lines of text relative to current setting of search pointer.
FILE,PROVIDING IT IS THE EDIT FILE.	
IT IS ESPECIALLY USEFUL WHEN	
ADDING TEXT IN THE BODY OF A FILE.	
? IS:RTEXTUALR,I OR SOURCEI	Command is relative to position of search pointer.
PHRASE NOT FOUND.	
? RESET	
?IS:RTEXTUALR,I OF SOURCEI	Add string / OR SOURCE/ directly after first occurrence of string /TEXTUAL/.
? LIST;*	
THE ADD COMMAND CAN BE VERY	
USEFUL WHEN CREATING A TEXTUAL OR SOURCE	
FILE. IN FACT, IT IS ONE	
OF THE FEW METHODS THAT CAN BE	Listing of altered file.
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	

REMOVAL OF INFORMATION

Two types of operation are available for removing information from the edit file, DELETE and BLANK.

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.

LINE MODE FORMATS (DELETE OR D)

Command	Explanation
DELETE	Erases the line of the edit file specified by the search pointer.
DELETE;n	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 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/.

STRING MODE FORMATS (DELETES OR DS)

Command	Explanation
DELETES	Same as DELETE.
DELETES;n	Same as DELETE;n.
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/.

BLANK COMMAND

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

LINE MODE FORMATS (BLANK OR B)

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

STRING MODE FORMATS (BLANKS OR BS)

Command	Explanation
BLANKS	Same as BLANK.
BLANKS;n	Same as BLANK;n.
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/.

The following example illustrates the use of DELETE and BLANK commands.

Entry/Response	Commentary
EDIT	
BEGIN TEXT EDITING	
? L;*	List to end of file.
00010 PROGRAM RANDNUM	
00020 MAINSEED = 5**13	
00030 NEXTSEED = 5**15	
00040 HUNDRETH = 1./100.	
00050 BIGNUMBR = 2.**48	
00060 FRACTION = HUNDRETH*BIGNUMBR	
00070 IFRAC = INT(FRACTION)	
00080 DO 20 I = 1,10	
00090 INCRMENT = 0	
00100 NEXTSEED = NEXTSEED*MAINSEED	
00110 10 INCRMENT = INCRMENT+1	
00120 IF((INCRMENT*IFRAC).LY.NEXTSEED) GO TO 10	
00130 NEWNUM = INCRMENT	
00140 20 CONTINUE	
00150 END	
-END OF FILE-	
? DELETE:+BIGNUMBR+;*	Delete every line in file that contains the string /BIGNUMBR/.
2 OCCURENCES OF PHRASE FOUND.	
? S;3	Advance search pointer three lines.
? L;2	
00040 HUNDRETH = 1./100.	Lines 5 and 6 have been deleted.
00070 IFRAC = INT(FRACTION)	
? A	Add text after line relative to present position of search pointer (line 4).
ENTER TEXT.	
? /00050 BIG = 2.**48	
? 00060 FRACTION = HUNDRETH*BIG/	
READY.	
? BLANKS:XHUNDRETHX	Blank first occurrence of string /HUNDRETH/.
? L;2	
00040 = 1./100.	
00050 BIG = 2.**48	
? IS:/00040 /,/H100/	Insert string /H100/ immediately after string /00040 /.
? L	
00040 H100 = 1./100.	
? DS:/ /	Delete eight spaces denoted by string / /.
? L	
00040 H100 = 1./100.	
? BLANKS:)UNDRETH)	Blank first occurrence of string /UNDRETH/.
? LINE	
FILE AT LINE NUMBER 4.	
? F;2	Advance search pointer two lines and list line.
00060 FRACTION = H *BIG	
? ADDS:/H/	Add text after first occurrence of string /H/.
ENTER TEXT.	

Entry/Response	Commentary
? /100/ READY. ? L 00060 FRACTION = H100 *BIG ? DS:R R	Remove blanks.
? L 00060 FRACTION = H100*BIG ? R	Reset search pointer.
? F;13 00140 20 CONTINUE ? DS:/20/,/END/ ? L;*	Delete string specified by ellipsis /20/./END/.
00140 -END OF FILE- ? IS:*00140 *,*PRINT,NEWNUM* ? L	Note: string deleted, not line. Insert string /PRINT,NEWNUM/ immediately after string /00140 /.
00140 PRINT,NEWNUM ? A ENTER TEXT. ? /00150 20 CONTINUE ? 00160 END/	Add text after line indicated by search pointer.
READY. ? RESET ? LIST;*	Reset search pointer.
00010 PROGRAM RANDNUM 00020 MAINSEED = 5**13 00030 NEXTSED = 5**15 00040 H100 = 1./100. 00050 BIG = 2.**48 00060 FRACTION = H100*BIG 00070 IFRAC = INT(FRACTION) 00080 DO 20 I = 1,10 00090 INCRMENT = 0 00100 NEXTSEED = NEXTSEED*MAINSEED 00110 10 INCRMENT = INCRMENT+1 00120 IF((INCRMENT*IFRAC).LT.NEXTSEED) GO TO 10 00130 NEWNUM = INCRMENT 00140 PRINT,NEWNUM 00150 20 CONTINUE 00160 END -END OF FILE-	Altered file is listed to end of file.
? END END TEXT EDITING.	

SUBSTITUTION OF INFORMATION

The CHANGE 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.

CHANGE COMMAND

In effect, the CHANGE command combines a DELETE operation with 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 three ways.

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

2. 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 CR only. This causes the data entered in response to the most recent ENTER TEXT request to be used as the change information.

NOTE

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 CR only is entered in response to an ENTER TEXT request.

LINE MODE FORMATS (CHANGE OR C)

Command	Explanation
CHANGE	Replaces the line specified by the search pointer with the text that follows.
CHANGE;n	Replaces the first n lines of the edit file beginning at the search pointer.
CHANGE:/string/	Replaces the line containing the specified string; search for string begins at current position of search pointer.

Command	Explanation
CHANGE:/string/;n	Replaces the first n lines containing the string.
CHANGE:/string1/, /string2/	Replaces the line or group of lines containing ellipsis /string1/, /string2/.
CHANGE:/string1/, /string2/;n	Replaces the first n occurrences of the line or group of lines containing ellipsis /string1/, /string2/.

STRING MODE FORMATS (CHANGES OR CS)

Command	Explanation
CHANGES	Same as CHANGE.
CHANGES;n	Same as CHANGE;n.
CHANGES:/string/	Replaces the specified string; search for string begins at current position of search pointer.
CHANGES:/string/;n	Replaces the first n occurrences of the specified string.
CHANGES:/string1/, /string2/	Replaces the string of characters specified by the ellipsis /string1/, /string2/.
CHANGES:/string1/, /string2/;n	Replaces the first n occurrences of a string of characters specified by the ellipsis /string1/, /string2/.

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.

There are four valid formats.

Command	Explanation
RS:/string/	Equivalent to DELETES:/string/.
RS:/string/;n	Equivalent to DELETES:/string/;n.
RS:/string1/, /string2/	Replaces the first occurrence of string1 from the search pointer with string2.
RS:/string1/, /string2/;n	Each of n occurrences of string1 is replaced with string2, beginning at the search pointer.

The following is an example of CHANGE and RS commands.

Entry/Response	Commentary

```
EDIT

 BEGIN TEXT EDITING.
? L;*                                                    List to end of file.
00010 PROGRAM RANDNUM
00020 MAINSEED = 5**13
00030 NEXTSEED = 5**15
00040 H100 = 1./100.
00050 BIG = 2.**48
00060 FRACTION = H100*BIG
00070 IFRAC = INT(FRACTION)
00080 DO 20 I = 1,10
00090 INCRMENT = 0
00100 NEXTSEED = NEXTSEED*MAINSEED
00110 10 INCRMENT = INCRMENT+1
00120 IF((INCRMENT*IFRAC).LT.NEXTSEED) GO TO 10
00130 NEWNUM = INCRMENT
00140 PRINT,NEWNUM
00150 20 CONTINUE
00160 END
 -END OF FILE-
? CHANGE                                                 Change line indicated by current setting
 ENTER TEXT.                                             of search pointer.
? /00010 PROGRAM RANDOM (OUTPUT)/
 READY.
? RS:+NEXTSEED+,+NEXTSD+;*                               Replace each occurrence of the string
       4  OCCURANCES OF PHRASE FOUND.                    /NEXTSEED/ with /NEXTSD/.
? F;2
00030 NEXTSD = 5**15
? F;7
00100 NEXTSD = NEXTSD*MAINSEED
? CS:RSEEDR                                              Change first occurrence of string
 ENTER TEXT.                                             /SEED/ (relative to search pointer)
? /SD/                                                   with the string /SD/.
 READY.
? L
00100 NEXTSD = NEXTSD*MAINSD
? RESET
? CS:/SEED/,/15/                                         Change ellipsis string specified by
 ENTER TEXT.                                             /SEED/, /15/.
? 8SD = 5**13
? 00030 NEXTSD = 5**178
 READY.
? L;3
00010 PROGRAM RANDOM (OUTPUT)
00020 MAINSD = 5**13
00030 NEXTSD = 5**17
? RS:.INCREMENT.,.INC.;5                                 Replace first five occurrences of string
? RS:#FRACTION#,#FRAC#;2                                 /INCRMENT/ with /INC/ and two
                                                         occurrences of string /FRACTION/
                                                         with /FRAC/.
```

Entry/Response

Commentary

```
? LS:/INC/;5
    INC
       INC    INC
        INC
              INC
? CHANGE
 ENTER TEXT.
? M00005***THIS PROGRAM GENERATES
? 00006*    10 RANDOM NUMBERS BETWEEN
? 00007*    1 AND 100.
? 00010 PROGRAM RANDOM (OUTPUT)M
 READY.
? C:/NEWNUM/,/NEWNUM/
 ENTER TEXT.
? /00130 NRAN = INC
? 00140 PRINT 30, I, NRAN
? 00145 30 FORMAT(I2,2X,I4)/
 READY.
? L;*
00005***THIS PROGRAM GENERATES
00006*  10 RANDOM NUMBERS BETWEEN
00007*  1 AND 100.
00010 PROGRAM RANDOM (OUTPUT)
00020 MAINSD = 5**13
00030 NEXTSD = 5**17
00040 H100 = 1./100.
00050 BIG = 2.**48
00060 FRAC = H100*BIG
00070 IFRAC = INT(FRAC)
00080 DO 20 I = 1,10
00090 INC = 0
00100 NEXTSD = NEXTSD*MAINSD
00110 10 INC = INC+1
00120 IF((INC*IFRAC).LT.NEXTSD) GO TO 10
00130 NRAN = INC
00140 PRINT 30, I, NRAN
00145 30 FORMAT(I2,2X,I4)
00150 20 CONTINUE
00160 END
 -END OF FILE-
? END
 END TEST EDITING.
```

To add text to the beginning of the file, it is necessary to use the CHANGE command.

Change line(s) containing ellipsis search string /NEWNUM/,/NEWNUM/

LOADING THE STRING BUFFER

The EXTRACT command appends a copy of information from the edit file to the string buffer; it does not affect the contents of the edit file in any way. The CLEAR command restores the string buffer to an empty condition. Information is transferred from the string buffer to the edit file by the ADD or CHANGE commands.

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 equals *, all lines to the end of the edit file are copied.)
EXTRACT:/string/	Copies the first line containing the string; search for string begins at current position of search pointer.
EXTRACT:/string/;n	Copies the nth line containing the string.
EXTRACT:/string1/,/string2/	Copies the first line or group of lines containing ellipsis /string1/,/string2/.
EXTRACT:/string1/,/string2/;n	Copies the nth occurrence of the line or group of lines containing ellipsis /string1/,/string2/.

STRING MODE FORMATS (ES)

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

CLEAR STRING BUFFER (CLEAR OR CL)

The string buffer is not cleared automatically after an ADD or CHANGE command. It is the user's responsibility to clear the string buffer; if he does not 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.

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

Entry/Response	Commentary
`EDIT`	
`BEGIN TEXT EDITING.`	
`? LIST;*`	List to end of file.
`THE EXTRACT COMMAND CAN BE`	
`VERY USEFUL IN REARRANGING`	
`LINES OF TEXT.`	
`-END OF FILE-`	
`? S;2`	
`? EXTRACT`	The third line is copied into the string buffer.
`? R`	
`? ADD`	The contents of the string buffer are inserted into the file.
`ENTER TEXT.`	
`? $`	
`READY.`	
`? L;*`	
`THE EXTRACT COMMAND CAN BE`	Notice that the EXTRACT command does not delete the information that it copies into the buffer.
`LINES OF TEXT.`	
`VERY USEFUL IN REARRANGING`	
`LINES OF TEXT.`	
`-END OF FILE-`	
`? S:#LINES#;2`	Advance search pointer to line containing second occurrence of string /LINES/ and delete line.
`? DELETE`	
`-END OF FILE-`	
`? R`	
`? A`	
`ENTER TEXT.`	
`? )USED TO RESTRUCTURE INDIVIDUAL)`	Copy string /REARRANGING/ to string buffer.
`READY.`	
`? ES:+REARRANGING+`	
`? A;*`	Add contents of string buffer to end of file.
`ENTER TEXT.`	
`? $`	
`READY.`	Note that text from previous EXTRACTs remains in the string buffer and subsequent EXTRACTs causes text to be appended.
`? L;*`	
`THE EXTRACT COMMAND CAN BE`	
`USED TO RESTRUCTURE INDIVIDUAL`	
`LINES OF TEXT`	
`VERY USEFUL IN REARRANGING`	
`LINES OF TEXT.`	
`REARRANGING`	Clear string buffer.
`-END OF FILE-`	
`? CLEAR`	
`? S:/VERY/`	Copy string /RESTRUCTURE/ to string buffer; command is relative to position of search pointer.
`? D;*`	
`-END OF FILE-`	
`? ES:*RESTRUCTURE*`	Change string /INDIVIDUAL/ to contents of string buffer.
`-END OF FILE-`	
`? RESET`	
`? ES:*RESTRUCTURE*`	
`? CHANGES:KINDIVIDUALK`	List to end of file.
`ENTER TEXT.`	

Entry/Response	Commentary
? $ READY. ? L;* 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.	Delete first occurrence of string /RESTRUCTURE/.
? CLEAR ? ES:# STRING#,#BUFFER, # ? DS:# STRING#,#BUFFER# ? ADDS:.THAT THE. ENTER TEXT.	Clear string buffer. Note the difference between these two ellipsis strings. Contents of string buffer is inserted after string /THAT THE/.
? $? 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. IS NOT CLEARED AFTER AN ADD OR CHANGE $ COMMAND. TO REMOVE TEXT FROM THE STRING BUFFER, USE THE CLEAR COMMAND. -END OF FILE-	
? CLEAR ? D:/BUFFER,/,/E STRING/ ? 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-	Clear string buffer.
? END END TEXT EDITING. READY.	

EDIT FILE DIMENSIONING COMMANDS

The LENGTH and WIDTH commands are used to respecify the dimensions of the edit file. The ALIGN command removes extraneous blanks for printing purposes.

LENGTH COMMAND (LENGTH)

The LENGTH command limits the number of lines of the edit file on which other edit commands can operate and also resets the search pointer to the first line. Multiple truncations are allowed to a maximum of eight.

The following are valid forms of the command.

Command	Explanation
LENGTH;n	Truncates the edit file at line n. All text information beyond line n is saved in a scratch file SCR3. Information in SCR3 is not affected by editing commands.
LENGTH;*	Restores original processing boundaries of the edit file by appending the contents of scratch file SCR3 to the edit file. This version of the command is meaningful only if a LENGTH;n command has been given previously.

CAUTION

A truncated file must be restored (with the LENGTH;* command) prior to entering the END command or the information that was truncated is lost.

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 $6 \leq n \leq 150$. 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.

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 are valid forms of this format control command.

Command	Explanation
ALIGN	Removes excess blanks between words in the line of text specified by the search pointer.
ALIGN;n	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 string begins at current position of 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/.

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

Entry/Response	Commentary
EDIT	
BEGIN TEXT EDITING.	
? LIST;*	List to end of 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-	
? LENGTH;4	Truncate edit file at line 4.
? WIDTH;20	Set width indicator to 20.
? L;*	List to end of file.
THE LENGTH COMMAND IS VERY USEFUL WHEN IT IS DESIRABLE TO WORK ON ONLY A SMALL PORTION OF A LARGE FILE.	Notice that WIDTH command does not affect the file unless an ALIGN command is used.
-END OF FILE-	
? LENGTH;*	Restore truncated file.
? SET;4	
? ALIGN;*	Align to end of file from line 5 with width of 20.
? RESET	
? 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	
? L;4	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?	Align lines containing ellipsis string /VERY/,/WORK/.
? L;*	
THE LENGTH COMMAND IS VERY USEFUL WHEN IT IS DESIRABLE TO WORK ON ONLY A SMALL PORTION OF A LARGE FILE. THE WIDTH	

Entry/Response	Commentary

```
COMMAND IS EFFECTIVE
ONLY IF FOLLOWED BY
AN ALIGN COMMAND.
 -END OF FILE-
? W;62
? AL;*
? 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-
? BLANKS:#WORK ON#
? WIDTH;32
? 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-
? RS:*     ONLY*,*EDIT*
? ALIGN;4
? L;*
     THE LENGTH COMMAND IS VERY
USEFUL WHEN IT IS DESIRABLE TO
EDIT A SMALL PORTION OF A LARGE
FILE.
     THE WIDTH COMMAND IS
EFFECTIVE ONLY IF FOLLOWED BY AN
ALLIGN COMMAND.
 -END OF FILE-
? END
 END TEXT EDITING.
```

Set width to 62 and align entire file.

Blank string /WORK ON/.

Align four lines from search pointer.

TABULATION COMMANDS

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

DEFTAB COMMAND (DEFTAB OR DT)

The DEFTAB command defines a single tab character that is later 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.

TAB COMMAND (TAB OR T)

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

The following are valid forms of the command.

Command	Explanation
TAB	Clears existing tab stops.
TAB:/$t_1,t_2,\ldots,t_n$/	Each t_i is a column number, $t_i > 0$. A maximum of seven tab column numbers may be specified.

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. If the user forgets his tab specifications and enters tabbed text, he has to either begin over or use the RS or CS commands to replace each tab character in the text with the correct number of blanks. For a small file this is no great task but for a very large file this can be a formidable effort. Typical of this difficulty would be a user who has entered a long COMPASS program with tab characters in each line to establish the columnar formatting of the language and has forgotten to pre-establish the tab parameters. It is possible to preserve the coding that has been typed in and initiate proper tabbing. Essentially, the procedure is to extract the program, delete the remainder of the file, and then build an input file that contains the tab directives and the extracted program. The following example uses a patterned indirect access permanent file and four tabbed lines to demonstrate this procedure.

Entry/Response	Commentary

`GET,MATRIX`
 	
`READY.`
`EDIT, MATRIX`
	A copy of an indirect access permanent file called MATRIX is edited with the intention of adding four lines of tabulated text after the fourth line.

`BEGIN TEXT EDITING.`
`? L:*`
`1 ABCDE`
`2  BCDEF`
`3  CDEFG`
`4  DEFGH`
`5  EFGHI`
`6  FGHIJ`
`7  GHIJK`
`8  HIJKL`
`9  IJKLM`
`-END OF FILE-`	

`? F:/DEFGH/`
`4  DEFGH`
`? A`
`ENTER TEXT.`
`? /#THIS#IS#TAB$LINE#1.`
`? #THIS#IS#TAB#LINE#2.`
`? #THIS#IS#TAB#LINE#3.`
`? #THIS#IS#TAB#LINE#4./`
`READY.`
	The added text has the tab character # included in each line. However, the user has failed to enter tabulation specifications.

`? R`
`? L;*`
`1 ABCDE`
`2  BCDEF`
`3  CDEFG`
`4  DEFGH`
`#THIS#IS#TAB#LINE#1.`
`#THIS#IS#TAB#LINE#2.`
`#THIS#IS#TAB#LINE#3.`
`#THIS#IS#TAB#LINE#4.`
`5  EFGHI`
`6  FGHIJ`
`7  GHIJK`
`8  HIJKL`
`9  IJKLM`
`-END OF FILE-`
	A listing of the edited file reveals the oversight. The tab character is still there instead of the desired spacing.

`? F:/#/`
`#THIS#IS#TAB#LINE#1.`
`? EXTRACT;4`
	Position to the first line of inserted text and extract the four lines of new text. This is put in the string buffer.

`R`
`D;*`
`-END OF FILE-`
	The entire original file is deleted. Only the name MATRIX remains.

`? A`
`ENTER TEXT.`
`? +L;*`
`? F:/DEFGH/`
`? DEFTAB:/#/`
`? TAB:/6,18,22,40,45/`
	An input file of edit directives is built.

Entry/Response	Commentary
? ADD ? L;* ? END+ READY. ? L;*	The input file is checked with a listing.
L;* F:/DEFGH/ DEFTAB:/#/ TAB:/6,18,22,40,45/ ADD L;* END -END OF FILE- ? F:/ADD/ ADD ? ADD ENTER TEXT ? $ READY. ? L;*	The text in the string buffer is added to the input file.
ADD #THIS#IS#TAB#LINE#1. #THIS#IS#TAB#LINE#2. #THIS#IS#TAB#LINE#3. #THIS#IS#TAB#LINE#4. L;* END -END OF FILE- ? RS:/#/,/+#/ ? RS:/#4./,/#4.+/ ? R	The necessary delimiters are added.
? L;* L;* F:/DEFGH/ DEFTAB:/#/ TAB:/6,18,22,40,45/ ADD +#THIS#IS#TAB#LINE#1. #THIS#IS#TAB#LINE#2. #THIS#IS#TAB#LINE#3. #THIS#IS#TAB#LINE#4.+ L;* END -END OF FILE-	The input file is checked with a listing.
? END END TEXT EDITING. SRU 0.189 UNTS READY. RENAME,IN1=MATRIX	Exit is made from text editor in order to use control statements. The local copy of MATRIX is renamed so as not to conflict with the new copy that is obtained with a GET.
READY. GET,MATRIX	
READY. EDIT,MATRIX,,IN1	Text editing is reinitiated with the full EDIT command. The edit file is MATRIX, the mode is normal, and the input file is IN1.

Entry/Response	Commentary

```
 BEGIN TEXT EDITING.
1 ABCDE
2  BCDEF
3   CDEFG
4    DEFGH
5     EFGHI
6      FGHIJ
7       GHIJK
8        HIJKL
9         IJKLM
 -END OF FILE-
4    DEFGH
 ENTER TEXT.
 READY.
4    DEFGH
     THIS        IS  TAB             LINE 1.
     THIS        IS  TAB             LINE 2.
     THIS        IS  TAB             LINE 3.
     THIS        IS  TAB             LINE 4.
5     EFGHI
6      FGHIJ
7       GHIJK
8        HIJKL
9         IJKLM
 -END OF FILE-
 END TEXT EDITING.
READY.
LIST,F=MATRIX
```

A listing of the full edited file confirms the success.

```
1 ABCDE
2  BCDEF
3   CDEFG
4    DEFGH
     THIS        IS  TAB             LINE 1.
     THIS        IS  TAB             LINE 2.
     THIS        IS  TAB             LINE 3.
     THIS        IS  TAB             LINE 4.
5     EFGHI
6      FGHIJ
7       GHIJK
8        HIJKL
9         IJKLM
READY.
```

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 STOPS t_1 t_2 ... t_n

If the tab stops have been cleared (refer to TAB command), the system responds:

```
TAB STOPS NONE.
```

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

Entry/Response	Commentary
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 character with stops at 5, 10, and 20.
? TAB:/5,10,20/	
? 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	Clears previous tab character.
? TAB	Clears existing tab stops.
? LT	
TAB STOPS NONE.	
? END	
END TEXT EDITING.	

EXTERNAL FILE MERGE

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 are valid forms of the command.

Command	Explanation
MERGE:/lfn/ MERGE:/lfn/;n MERGE:/pfn/ MERGE:/pfn/;n	The contents of file lfn or pfn are inserted into the edit file. Merging takes place after the nth line of the edit file, relative to the search pointer.
MERGE:/lfn/, /string/ MERGE:/lfn/, /string/;n MERGE:/pfn/, /string/ MERGE:/pfn/, /string/;n	The contents of file lfn or pfn are inserted into the edit file. Merging takes place after the nth line that contains /string/ and only if n lines containing /string/ are found.

MERGE is the only Text Editor command that can reference a working file (lfn) or a permanent file (pfn). The file referenced cannot be the current edit file or any other reserved file name (refer to appendix C). If pfn is a direct access permanent file, it must be attached to the user's job before entering Text Editor (refer to the NOS Time-Sharing User's Reference Manual for information regarding direct access file usage).

NOTE

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

The following example illustrates the use of MERGE command.

Entry/Response	Commentary
`LNH,F=TXT2`	Before entering Text Editor, time-sharing commands are used to list working file TXT2, which is a copy of a permanent file.

`  THIS FILE IS NAMED TXT2`
`AND IS A COPY OF AN INDIRECT`
`ACCESS PERMANENT FILE OF THE`
`SAME NAME.`
`READY.`	

`NEW XYZ`	Working primary file XYZ created, releasing working file TXT2.
`READY.`	
`EDIT`	Enter Text Editor with empty primary file XYZ 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`
`USING THE TEXT EDITOR ADD`
`COMMAND.`
`  THIS FILE IS NAMED TXT2`
`AND IS A COPY OF AN INDIRECT`
`ACCESS PERMANENT FILE OF THE`
`SAME NAME.`
` -END OF FILE-`	

`? MERGE:/TXT2/,/INDIRECT/`	Merge permanent file TXT2 after first occurrence of string /INDIRECT/.

`? L;*`
`  THIS FILE IS BEING BUILT`
`USING THE TEXT EDITOR ADD`
`COMMAND.`
`  THIS FILE IS NAMED TXT2`
`AND IS A COPY OF AN INDIRECT`
`  THIS FILE IS NAMED TXT2`
`AND IS A COPY OF AN INDIRECT`
`ACCESS PERMANENT FILE OF THE`
`SAME NAME.`
`ACCESS PERMANENT FILE OF THE`
`SAME NAME.`
` -END OF FILE-`	

`? END` ` END TEXT EDITING.`	

STRING INCIDENCE COUNTING

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.

LINE MODE FORMATS (NUMBER OR N)

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

STRING MODE FORMATS (NUMBERS OR NS)

Command	Explanation
NUMBERS	Same as NUMBER
NUMBERS:/string/ NUMBERS:/string1/,/string2/	Returns a count of the number of occurrences of the specified string. Note that the string can be either single phrase or ellipsis.

The following example illustrates the use of NUMBER command.

Entry/Response	Commentary
BEGIN TEXT EDITING.	
? L;*	List to end of file.
00005***THIS PROGRAM GENERATES	
00006* 10 RANDOM NUMBERS BETWEEN	
00007* 1 AND 100.	
00010 PROGRAM RANDOM (OUTPUT)	
00020 MAINSD = 5**13	
00030 NEXTSD = 5**17	
00040 H100 = 1./100.	
00050 BIG = 2.**48	
00060 FRAC = H100*BIG	
00070 IFRAC = INT(FRAC)	
00080 DO 20 I = 1,10	
00090 INC = 0	
00100 NEXTSD = NEXTSD*MAINSD	
00110 10 INC = INC+1	
00120 IF((INC*IFRAC).LT.NEXTSD) GO TO 10	
00130 NRAN = INC	
00140 PRINT 30, I, NRAN	
00145 30 FORMAT(I2,2X,I4)	
00150 20 CONTINUE	
00160 END	
-END OF FILE-	
? NUMBER	Request line count from search pointer to end of file.
20 LINES TO EOF.	
? NUMBER:/NEXTSD/	Request count of number of lines containing string /NEXTSD/.
3 OCCURANCES OF PHRASE FOUND.	
? NUMBERS:/NEXTSD/	Request count of number of occurrences of string /NEXTSD/.
4 OCCURANCES OF PHRASE FOUND.	
? S;10	
? N	Request line count from search pointer to end of file.
10 LINES TO EOF.	
? NS:/INC/,/0/	Request count of occurrences of ellipsis string /INC/,/0/.
4 OCCURRENCES OF PHRASE FOUND.	
? LISTS:/INC/,/0/;*	Actual occurrences of ellipsis string /INC/,/0/ are listed.
INC = 0	
INC = INC+1	
0 INC*IFRAC).LT.NEXTSD) GO TO 10	
INC	
0	
? N:/INC/,/0/	Request count of number of lines containing ellipsis string /INC/,/0/.
3 OCCURRENCES OF PHRASE FOUND.	
? R	
? NS:/MAINSD/	Request count of occurrences of string /MAINSD/ in file.
2 OCCURENCES OF PHRASE FOUND.	
? NS:/PRINT/	Request count of occurrences of string /PRINT/ in file.
1 OCCURRENCES OF PHRASE FOUND.	
? END	
END TEXT EDITING.	

TERMINATING EDIT SESSION

END COMMAND (END)

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

The command format is:

```
END
```

The system responds

```
END TEXT EDITING.
```

It is necessary to terminate text editing whenever it is necessary or desirable to do a file operation (such as SAVE or REPLACE).

The user may 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 Text Editor control. In this case, the text file contents are unpredictable and all output files may be lost.

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.

SYSTEM ACCESS PROCEDURES A

LOG-IN SEQUENCE

1. Complete the dial-in procedure to connect the terminal to the NOS Time-Sharing System. Check to ensure that terminal switches (full/half duplex, even/odd parity, baud rate, etc.) are set to the correct position.

2. When the dial-in procedure is complete, it may be necessary to identify the type of terminal being used before proceeding.

Terminal Type	Identification
Correspondence code terminal/standard print	Press ATTN key
Correspondence code terminal/APL print	Type A and press ATTN key
ASCII code terminal/ standard print	Press (CR)
Memorex 1240 (ASCII code) terminal/APL print	Type A
Block transmission (ASCII code) terminal/standard print	Type B

3. When communication with NOS is established, the system types out three lines. The first line is in the format

   yy/mm/dd. hh.mm.ss.

   which gives the date and the time. The second line is the identifying header of the installation. In this manual, the examples use

   CDC MULTI-MODE OPERATING SYSTEM NOS

   The third line is a request for a family name

   FAMILY:

   The user types in the name of the family to which he is assigned and presses carriage return. If he is using the default family, he only presses carriage return.

4. There follows a request for the user number. The format is

   USER NUMBER:

   The user number is typed in on the same line and the carriage return depressed.

5. The system request the user's password as follows:

   PASSWORD
   ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠

The user types his password over the darkened area and presses carriage return. If he is not using a password, he only presses carriage return.

If the family name, user number, or password are not acceptable, the system responds

```
IMPROPER LOG IN, TRY AGAIN
FAMILY:
```

If the user is unsuccessful at logging in four times in succession, the system issues the message

```
ILLEGAL TERMINAL.
```

and disconnects the terminal.

6. If the family name, user number, and password are acceptable, the system prints the identifying number and type of the terminal on the next line. An example would be

```
TERMINAL:        45, TTY
```

7. The system follows this with either

```
RECOVER/CHARGE
```

or with

```
RECOVER/SYSTEM:
```

If RECOVER/CHARGE was issued, the user types in the word CHARGE and follows it with his charge number and project number. The system responds with

```
READY.
```

The user then enters the name of the subsystem to be used or any valid command.

If RECOVER/SYSTEM was issued, the user immediately enters the name of the subsystem to be used or any valid command.

8. If the subsystem entered above was BATCH, the system responds with

```
$RFL, 20000 (or a specified filed length)
/
```

The user then begins entering commands.

If the subsystem entered above was BASIC, FTN, or EXECUTE, the system responds with

```
OLD, NEW, OR LIB FILE:
```

The user response is one of the following:

OLD CR	This references a file that was previously saved as an indirect access permanent file.
NEW CR	This establishes a new primary working file.
LIBRARY CR	This references an indirect access permanent file that resides in the catalog of the special user number library.

The system responds:

FILE NAME:

The user then enters the one to seven character name of the file he is accessing or creating. He follows this with a carriage return.

If no errors are detected, the system responds:

READY.

The following is a typical log in:

```
yy/mm/ss. hh.mm.ss.
 CDC MULTI-MODE OPERATING SYSTEM.        NOS
FAMILY:
USER NUMBER: EFD250ø1
PASSWORD
■■■■■■■■
TERMINAL:      16,TTY
RECOVER/CHARGE: CHARGE,23,93N156

READY.
```

LOG-OFF SEQUENCE

When the user wishes to terminate the session, he logs off the system by entering the BYE or GOODBYE command. All current working files are then released and the system prints

```
(user number)   LOG  OFF.   hh.mm.ss.
(user number)   SRU          s.ss UNTS
```

and disconnects the terminal. The designation hh.mm.ss. is the time of log off, and s.ss is a measure of the system resources used from log in to log off.

SYSTEM FILE MANAGEMENT B

GENERAL

This appendix is included to provide the inexperienced user with enough information to perform basic file operations using the NOS Time-Sharing System. The outlined techniques are not necessarily the only ways to accomplish a given result, nor necessarily the most efficient. Direct access permanent files are not discussed at length since a user's first exposure to the Text Editor normally incorporates the use of working files and indirect access permanent files.

For more detailed information on NOS file handling, refer to the NOS Time-Sharing User's Reference Manual. Also, if operating under the batch subsystem, refer to the NOS Reference Manual, volume 1.

FILE TYPES

A clear understanding of files is vital to efficient use of the Text Editor. There are basically two types of files that the user can be associated with, working files and direct access files. It is important that the Text Editor user be aware of the type of file that is being edited and the effect of such editing to the file. There are primarily two cases to remember: either the file being edited is a copy of a file or it is the original file. In the latter instance, extreme care must be exercised since all changes affect the only existing copy of the file being edited.

WORKING FILES

Generally, a working file is either a new file, created by the user, or a copy of a file that already exists in the system (refer to Indirect Access Permanent Files). All working files are temporary in nature and can exist no longer than the user is logged into the system. Working files may be created, accessed, and released at the discretion of the user while he is logged into the system, but they are automatically released when he logs off the system.

Working files may also be referred to as local files. The parameter lfn on most time-sharing commands and permanent file commands signifies a local file name. Therefore, throughout this manual, the terms are synonymous.

PRIMARY FILES

The primary file is a working file that has special significance in certain system commands. In general, those time-sharing commands that have the optional lfn parameter reference the primary file if the parameter is omitted. There is at most one primary file active or available to the user at any given time.

INDIRECT ACCESS PERMANENT FILES

An indirect access file is a permanent file situated on mass storage. When accessed the system generates a copy of the indicated permanent file which becomes a working file. Alterations to the working copy have no effect on the original permanent file.

DIRECT ACCESS PERMANENT FILES

A direct access file is also a permanent file situated on mass storage. When accessed the file becomes linked directly to the user's job. No copy of the permanent file is created; hence, all editing and other alterations are performed directly on the permanent file itself. A direct access file cannot become a working file or primary file.

FILE HANDLING PROCEDURES

This section discusses briefly the file handling procedures and commands commonly used in conjunction with the Text Editor. Its purpose is to provide simple procedures that work; it does not reflect the full range of file handling capability.

CREATING A WORKING FILE

A working file is created either as a new file or as a working copy of an indirect access file. To create a new file, enter:

```
NEW (CR)
```

The system responds:

```
FILE NAME:
```

and a valid file name should be entered. The name should be different from all permanent file names currently stored under the user number if the file is to be made permanent at a later time. When the system accepts the file name, it responds:

```
READY.
```

The two steps can be combined into one by entering:

```
NEW,lfn (CR)
```

where lfn is the name of the file.

A working file created with a NEW, OLD, or LIBRARY command also becomes the primary file. To create a working file that is a copy of an indirect access permanent file, enter the OLD or LIBRARY (for a library file) command.

```
OLD,pfn (CR)
```

With this command, a copy of the indirect access permanent file pfn becomes the primary working file. To give the copy of pfn a different working file name lfn, use the command:

```
OLD,lfn=pfn (CR)
```

In both of the previous commands, LIBRARY can be substituted for OLD with the copy being made from a library file.

To retrieve a copy of an indirect access permanent file without affecting the current primary file or if no primary file is desired, use the command:

GET, pfn CR

The permanent file pfn is copied as a working file and the primary file is unaffected. To access the permanent file pfn under a different name, enter the command:

GET, lfn=pfn CR

where lfn is the new name of the working file copied from permanent file pfn.

In either a GET, LIBRARY, or OLD command, the choice of using the single file name (pfn) or the double file name specification (lfn=pfn) depends on what the user intends to do with the working file. If it is desired to store the working copy (with or without alterations) as a permanent file without disturbing the current version of file pfn, it is necessary to use an lfn=pfn specification or a RENAME command (discussed later) at some point in the processing of the file (although not necessarily when the file is first accessed).

A working file can also be created when entering Text Editor. The command:

EDIT, lfn CR

creates an empty working file named lfn if one does not already exist.

CREATING AN INDIRECT ACCESS PERMANENT FILE

To retain a copy of a working file on mass storage as an indirect access permanent file, enter the command:

SAVE CR

to store the primary file, or:

SAVE, lfn CR

to store the working file lfn (lfn can also be the name of the primary file). If the system responds:

lfn ALREADY PERMANENT

a permanent file named lfn already exists. To save the working file in this case, enter the command:

SAVE, lfn=pfn CR

where pfn is different from the name lfn. This causes the working file lfn to be saved permanently under the permanent file name pfn.

When the system responds:

READY.

the working file is established as a permanent file.

CAUTION

Newly created files should generally be saved prior to editing as a precautionary measure. Some Text Editor commands are powerful and can ruin a working file if the user makes a mistake. Also, it may be desirable to exit from the Text Editor at various stages of the edit procedure to save the edit file.

REPLACING A PERMANENT FILE

After a copy of an indirect access permanent file is obtained and alterations are performed, it is possible to replace the former version with the altered version. If the working copy is obtained with the command:

OLD, pfn ©R

the subsequent command:

REPLACE ©R

causes the permanent file pfn to be purged and replaced with the working copy, including alterations.

The REPLACE command without parameters acts as a SAVE if the primary file does not have the same name as a permanent file. The command:

REPLACE, lfn ©R

causes working file lfn to replace a permanent file of the same name if such a file exists; otherwise, it is equivalent to the command:

SAVE, lfn ©R

The command:

REPLACE, lfn=pfn ©R

causes working file lfn to replace the current contents of permanent file pfn.

It is important to use caution in the use of the REPLACE command or any other command that alters a permanent file. In a long session at the terminal, it is not uncommon for a user to forget what the primary file name is, and inadvertently specify a file replacement not intended. For example,

```
OLD, BLAB
    .
    .
    .
GET, GAB=BLAB
```

Two copies of the permanent file BLAB now exist as working files. If alternations are done on GAB, and the user enters the command:

REPLACE ⓒⓡ

the permanent file BLAB does not receive the changed version. Instead, the user should enter

REPLACE, GAB=BLAB. ⓒⓡ

BUILDING A FILE

Several methods of building a file are available. EDIT can be entered immediately and the file built using Text Editor commands, as explained in Adding and Building Text in section 3. Also, various time-sharing commands can be used. For information on the latter, refer to the following sections of the NOS Time-Sharing User's Reference Manual.

Section 3: File Sorting

Section 4: Terminal Control Commands (AUTO, NORMAL); Time-Sharing Job Commands (NOSORT, PACK, SORT, TEXT).

RENAMING A WORKING FILE

The command:

RENAME, lfn_1=lfn_2 ⓒⓡ

changes the name of working file lfn_2 to lfn_1. If another working file has the name lfn_1, that file is released.

Example:

```
OLD, TXTFILE
     .
     .
     .
RENAME, TXTFL1=TXTFILE
SAVE
```

The working file is saved as TXTFL1 without affecting the permanent file TXTFILE.

LISTING A FILE

Any working file can be listed with the command:

LIST, F=lfn ⓒⓡ

or

LNH, F=lfn ⓒⓡ (List with no header)

The F=lfn parameter may be omitted if the primary file is being listed.

If the file has line numbers, begin the listing at line n with the command:

LIST,n ⓒ
or
LNH,n ⓒ

The n parameter and F=lfn cannot be used in the same command. If the n parameter is used, the file must be sorted to obtain an accurate listing.

ANNOTATED SAMPLE TERMINAL SESSION

The following sample terminal session illustrates the use of NOS file handling commands described earlier in this appendix. Several other commands, such as TEXT and RUN, are used in the example; although not directly related to Text Editor usage, they tend to be used frequently in a typical terminal session.

Entry/Response	Commentary
yy/mm/dd. hh.mm.ss. CDC MULTI-MODE OPERATING SYSTEM NOS FAMILY: USER NUMBER: USER123 PASSWORD ■■■■■■■■ TERMINAL: 10,TTY RECOVER/SYSTEM: NULL OLD, NEW, OR LIB FILE: NEW FILE NAME: TXT1 READY.	User USER123 logs in, entering the NULL subsystem and creating a new file named TXT1.
TEXT ENTER TEXT MODE.	User enters text mode because data to be entered do not have line numbers.
THIS FILE IS BEING CREATED IN TEXT MODE. IT DOES NOT REQUIRE LINE NUMBERS.	
EXIT TEXT MODE.	To terminate text mode, press the interrupt or break key from an ASCII code terminal or the ATTN key from a correspondence code terminal.
PACK READY.	The PACK command compresses the file into one record.
LIST	The primary file is listed with header information.
yy/mm/dd. hh.mm.ss. PROGRAM TXT1 THIS FILE IS BEING CREATED IN TEXT MODE. IT DOES NOT REQUIRE LINE NUMBERS. READY.	
SAVE READY.	TXT1, the primary file, is stored as an indirect access permanent file.

Entry/Response	Commentary
`GET,FRAN2` `READY.`	A copy of permanent file FRAN2 is obtained as a working file.
`LNH,F=FRAN2`	The file is listed without header information. The F parameter is used to distinguish this file from the primary file, TXT1.

```
00005***THIS PROGRAM GENERATES
00006*  10 RANDOM NUMBERS BETWEEN
00007*  1 AND 100.
00010 PROGRAM RANDOM (OUTPUT)
00020 MAINSD = 5**13
00030 NEXTSD = 5**17
00040 H100 = 1./100.
00050 BIG = 2.**48
00060 FRAC = H100*BIG
00070 IFRAC = INT(FRAC)
00080 DO 20 I = 1,10
00090 INC = 0
00100 NEXTSD = NEXTSD*MAINSD
00110 10 INC = INC+1
00120 IF((INC*IFRAC).LT.NEXTSD) GO TO 10
00130 NRAN = INC
00140 PRINT 30, I, NRAN
00145 30 FORMAT(I2,2X,I4)
00150 20 CONTINUE
00160 END
READY.
```

`FTN` `RETURN.`	The FORTRAN subsystem is entered to execute the program.
`RUN,I=FRAN2`	The RUN command causes the program to be executed. The I parameter is necessary to execute a file other than the primary file.

```
 1    46
 2    45
 3    54
 4    88
 5    73
 6    94
 7    97
 8    21
 9    57
10    44
00746


SRU      1.110 UNTS.
RUN COMPLETE.
```

`NEW,TXT1`	A new empty primary file named TXT1 is created, releasing the previous primary file of the same name.

```
READY.
001 PROGRAM OUT(OUTPUT)
002 PRINT 10
003 10 FORMAT(* PROGRAM OUT*)
004 END
RNH
```

	Run program with no header.

Entry/Response	Commentary
`PROGRAM OUT`	TXT1 is a simple FORTRAN program that produces one line of output.
`SRU 0.857 UNTS`	
`RUN COMPLETE.`	
`SAVE` `TXT1 ALREADY PERMANENT.`	Attempt is made to save file TXT1, but a file of the same name already exists. Thus, the user stores the new file under the different name, OUT.
`SAVE,TXT1=OUT` `READY.`	
`OLD,TXT=TXT1` `READY.`	A copy of permanent file TXT1 becomes the primary file with the name TXT.
`TEXT` `ENTER TEXT MODE.`	An addition is made to primary file TXT in text mode. Permanent file TXT1 is unaffected.
`THIS SHOULD BE ADDED TO` `THE END OF THE FILE.`	
`EXIT TEXT MODE.`	
`PACK`	Compress into one record.
`READY.`	
`LNH`	Primary file is listed.

`THIS FILE IS BEING CREATED`
`IN TEXT MODE. IT DOES NOT`
`REQUIRE LINE NUMBERS.`
`THIS SHOULD BE ADDED TO`
`THE END OF THE FILE.`
`READY.`	

`RENAME,NEWTXT=TXT`	Rename file TXT to NEWTXT.
`CP 0.001 SECS.` `READY.`	
`REPLACE,NEWTXT=TXT1` `READY.`	File NEWTXT replaces the contents of permanent file TXT1. Finally, permanent file TXT1 is accessed and listed to reflect the changes.
`ØLD,TXT1` `READY.`	
`LNH`	

`THIS FILE IS BEING CREATED`
`IN TEXT MODE. IT DOES NOT`
`REQUIRE LINE NUMBERS.`
`THIS SHOULD BE ADDED TO`
`THE END OF THE FILE.`
`READY.`	

Entry/Response			Commentary
BYE			USER123 logs off.
USER123	**LOG OFF**	**17.47.Ø2.**	System supplies log off information, including system resources used.
USER123	**SRU**	**1.662 UNTS.**	

EDIT MESSAGES C

EDIT ERROR MESSAGES

These messages indicate a condition that prevents processing of the command.

CONTROL CARD ERROR.

More than two parameters were passed when calling the Text Editor or an illegal second parameter was supplied.

ILLEGAL COMMAND.

The command word is invalid.

ILLEGAL DELIMITER.

An invalid delimiter was entered in response to the ENTER TEXT request from a local or remote batch job.

ILLEGAL DELIMITER - REENTER TEXT.

An invalid delimiter was entered in response to the ENTER TEXT request from a time-sharing job.

ILLEGAL FILE NAME.

The file name passes with MERGE command is illegal.

IMPROPER TRUNCATION.

Length specified in LENGTH;n command is equal to or greater than previous length specified.

MERGE ERROR, SECONDARY FILE EMPTY.

One of these conditions exists:

1. The file to be merged with the edit file is empty.
2. The file to be merged does not exist.
3. The file to be merged is a direct access file that was not attached to user's job prior to entering Text Editor.

PHRASE NOT FOUND.

The search string specified in /string/ was not found in the edit file.

RESERVED FILE NAME.

The file name passed with MERGE command or when invoking Text Editor is reserved for use by Text Editor. Reserved file names are:

INPUT	SCR1	SCR4
OUTPUT	SCR2	SCR5
SCR	SCR3	name of current edit file

cmd SYNTAX ERROR.

String and/or n parameter is illegal with command cmd.

REQUESTS AND INFORMATIVE MESSAGES

These messages are issued in the course of normal edit operation.

BEGIN TEXT EDITING.

This message is issued when initialization of editor is complete and awaiting the first command.

CONTINUE COMMAND? (YES or NO)

This is an inquiry as to whether or not an interrupted command should continue to be processed.

DISREGARD PREVIOUS TEXT? (YES or NO)

This is an inquiry as to whether the text that has been entered in response to a text-entering command should be retained or discarded.

-END OF FILE-

The search pointer is currently set at end of file or end of file encountered during execution of a LIST command.

END TEXT EDITING.

This message is issued following execution of the END command, indicating a return to subsystem mode.

ENTER TEXT.

New or replacement text is required to process ADD (ADDS) or CHANGE (CHANGES) commands.

ENTER TEXT FILE NAME.

This message is issued when text file name is not passed with Text Editor call.

FILE AT LINE NUMBER m.

This message is a current search pointer value, issued by LINE command processor.

INTERRUPT AT LINE m.

This message informs the user of the current position of an interrupted command.

m LINES TO EOF.

This message is a line count message issued by NUMBER command processor.

m OCCURRENCES OF PHRASE FOUND.

End of file was encountered before number of iterations specified in n parameter were completed.

READY.

The response to an ENTER TEXT request is completed; that is, the last carriage return was preceded immediately by the closing delimiter.

TAB STOPS t_1 t_2 ... t_n

This message is a list of tab stops issued by LISTAB command processor.

BATCH USAGE OF TEXT EDITOR D

Text Editor commands can be used by a batch job if it includes the EDIT control statement in it's control statement record. Format of this control statement is:

EDIT(lfn_1, m, lfn_2, lfn_3)

or

EDIT(FN=lfn_1, M=m, I=lfn_2, L=lfn_3)

lfn_1	Name of the file to be edited.
m	Mode of file processing N Normal (default) AS ASCII
lfn_2	File from which the edit commands are to be read. The default is a record in the job deck (INPUT).
lfn_3	File on which the output is written. Default is OUTPUT which is routed to the printer.

Example:

A batch job contains a record listing six types of cable assemblies and the amounts on hand. The job calls on Text Editor to produce two listings of specific types. The deck is shown in figure D-1. The cable list is the second record in the INPUT file. This is copied to a local file and given the name PARTS.

The EDIT command references PARTS which is automatically rewound. The mode of file processing is normal. The missing parameter after the comma indicates the source default of INPUT. This means the editing commands are taken from the next record in the job deck. This is the one following the list of six cables.

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 which moves the text over one column leaving the first position of each line blank. This causes single spacing.

```
CABLES,T10.
USER,XYZ22.
CHARGE,5N2201,23J8.
COPYCR(,PARTS)
EDIT(PARTS,N,,TEMP)
REWIND(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
```

Printout from execution of the above 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 D-1. Batch Job Using Text Editor

INDEX

COMMENT SHEET

MANUAL TITLE CDC NOS Version 1 Text Editor Reference Manual

PUBLICATION NO. 60436100 REVISION C

FROM: NAME:

BUSINESS ADDRESS:

FOLD ALONG LINE

PRINTED IN USA

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD

FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINNESOTA 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A

CONTROL DATA CORPORATION