60436100

CONTROL DATA CORPORATION

TEXT EDITOR REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS 1

SUMMARY OF AVAILABLE COMMANDS AND FORMATS

Command	Page
ADD(S)	3-8
ADD(S);n	3-8
ADD(S):/string/	3-8
ADD(S):/string/;n	3-8
ALIGN	3-23
ALIGN;n	3-23
ALIGN:/string/	3-23
ALIGN:/string/;n	3-23
ALIGN:/string1/, /string2/	3-23
ALIGN:/string1/, /string2/;n	3-23
BLANK(S)	3-12
BLANK(S);n	3-12
BLANK(S):/string/	3-12
BLANK(S):/string/;n	3-12
BLANK(S):/string1/, /string2/	3-12
BLANK(S):/string1/, /string2/;n	3-12
CHANGE(S)	3-15
CHANGE(S);n	3-15
CHANGE(S):/string/	3-15
CHANGE(S):/string/;n	3-16
CHANGE(S):/string1/, /string2/	3-16
CHANGE(S):/string1/, /string2/;n	3-16
CLEAR	3-19
DEFTAB	3-26
DEFTAB:/tabchar/	3-26
DELETE(S)	3-11
DELETE(S);n	3-11
DELETE(S):/string/	3-11
DELETE(S):/string/;n	3-11
DELETE(S):/string1/, /string2/	3-11
DELETE(S):/string1/, /string2/;n	3-11
EDIT, lfn_1, m, lfn_2, lfn_3.	2-1
EDIT, FN=lfn_1, M=m, I=lfn_2, L=lfn_3.	2-1
END	3-35
ES	3-19
ES;n	3-19
ES:/string/	3-19
ES:/string/;n	3-19
ES:/string1/, /string2/	3-19
ES:/string1/, /string2/;n	3-19
EXTRACT	3-19
EXTRACT;n	3-19
EXTRACT:/string/	3-19
EXTRACT:/string/;n	3-19
EXTRACT:/string1/, /string2/	3-19
EXTRACT:/string1/, /string2/;n	3-19
FIND(S)†	3-5
FIND(S);n	3-5
FIND(S):/string/	3-5
FIND(S):/string/;n	3-5
FIND(S):/string1/, /string2/	3-5
FIND(S):/string1/, /string2/;n	3-5
INSERTS:/string1/, /string2/;n	3-8
LENGTH;n	3-22
LENGTH;*	3-22
LINE	3-6
LIST(S)	3-2
LIST(S);n	3-2
LIST(S):/string/	3-2
LIST(S):/string/;n	3-2
LIST(S):/string1/, /string2/	3-2
LIST(S):/string1/, /string2/;n	3-2
LISTAB	3-29
MERGE:/lfn/	3-31
MERGE:/lfn/;n	3-31
MERGE:/pfn/	3-31
MERGE:/pfn/;n	3-31
MERGE:/lfn/, /string/	3-31
MERGE:/lfn/, /string/;n	3-31
MERGE:/pfn/, /string/	3-31
MERGE:/pfn/, /string/;n	3-31
NUMBER(S)	3-33
NUMBER(S):/string/	3-33
NUMBER(S):/string1/, /string2/	3-33
RESET†	3-3
RS:/string/	3-16
RS:/string/;n	3-16
RS:/string1/, /string2/	3-16
RS:/string1/, /string2/;n	3-16
SET†	3-3
SET;n	3-3
SET;-n	3-3
SET:/string/	3-3
SET:/string/;n	3-3
STOP	3-35
TAB	3-26
TAB:/$t_1, \ldots, t_n$/	3-26
WIDTH;n	3-22

†The position of the search pointer can change when using this command.

Manual Title CDC Text Editor Reference Manual   Pub. No. 60436100   Rev. G

As part of Control Data's continuing quality improvement program, we invite you to complete this questionnaire so that you may have a more direct influence on the manuals you use.

Please rate this manual for each general and individual category on a scale of 1 through 5 as follows:

1 - Excellent   2 - Good   3 - Fair   4 - Poor   5 - Unacceptable

I.	Writing Quality	____
A.	Technical accuracy	____
B.	Completeness	____
C.	Audience defined properly	____
D.	Readability	____
E.	Understandability	____
F.	Organization	____
II.	Examples	____
A.	Quantity	____
B.	Placement	____
C.	Applicability	____
D.	Quality	____
E.	Instructiveness	____
III.	Format	____
A.	Type size	____
B.	Page density	____
C.	Art work	____
D.	Legibility	____
E.	Printing/Reproduction	____
IV.	Miscellaneous	____
A.	Index	____
B.	Glossary	____

V. Please provide a yes or no answer regarding manuals in general:

A. I prefer that a manual on a software product be as comprehensive as possible; physical size is of little importance. ____

B. I prefer that information on a software product be covered in several small manuals, each covering a certain aspect of the product. Smaller manuals with limited subject matter are easier to work with. ____

C. I am interested primarily in reference manuals designed for ease of locating specific information. ____

D. I am interested primarily in user guides designed to teach the user about a product or certain capabilities of a product. ____

VI. We recognize that we have a wide variety of users. Please identify your primary area of interest or activity:

A. Student ____
B. Applications programmer ____
C. Systems programmer ____
D. How many years programming experience do you have? ____
E. What languages
1. Algol ____
2. Basic ____
3. Cobol ____
4. Compass ____
5. Fortran ____
6. PL/I ____
7. Other ____

F. Have you ever worked on non-CDC equipment? ____

1. If yes, approximately what percent of your experience is on non-CDC equipment? ____

2. How do you rate CDC manuals against other similar manuals using the 1-5 ratings. (Example: XYZ Corp. 2 means XYZ manuals are good as compared to CDC manuals.)

Burroughs ____
DEC ____
Hewlett-Packard ____
Honeywell ____
IBM ____
NCR ____
Univac ____
Other ______________ ____

General Comments ________________________________

____

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

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

FOLD ON DOTTED LINES AND STAPLE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

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

60436100

CONTROL DATA CORPORATION

TEXT EDITOR REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS 1

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released. This manual reflects NOS 1.0 at PSR level 404.
(09-30-74)	
B	Manual revised to reflect NOS 1.0 at PSR level 404. Eliminates the EXTRACTS and REPLACES
(05-01-75)	command forms, documents additions to the system command EDIT, and corrects various technical
	errors.
C	Manual revised to reflect NOS 1.1 at PSR level 419. New features include an expanded EDIT com-
(03-08-76)	mand, terminal 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.
D	Manual revised to reflect NOS 1.2 at PSR level 439. Explanation of general format of edit com-
(12-06-76)	mands has been rewritten and expanded for clarification, several descriptions have been clarified,
	typographical corrections have been made, and all examples have been redone in a format that dis-
	tinguishes user input from system response. This edition obsoletes all previous editions.
E	Revised to update the manual to NOS 1.2 at PSR level 452, to reformat error messages, and to
(07-15-77)	make typographical and technical corrections. Support of CDC CYBER 170 Series, Model 171 is
	also included.
F	Revised to update the manual to NOS 1.3 at PSR level 472 and to make typographical and technical
(03-20-78)	corrections. Support of the Interactive Facility (IAF) is included. Appendixes A and B have been
	deleted and appendix D has been modified and is included in section 2. This edition obsoletes all
	previous editions.
G	Revised to update the manual to NOS 1.4 at PSR level 498 and to make typographical and technical
(05-30-79)	corrections. A glossary is added. 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 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

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

PREFACE

The Text Editor (also known as the EDIT program) is a part of the Network Operating System (hereafter called NOS or the system) for CDC® CYBER 170 Series Computer Systems; CDC® CYBER 70 Series, Models 71, 72, 73, and 74 Computer Systems; and CDC® 6000 Series Computer Systems. By using EDIT, the user is able to make many changes to a file while at an interactive terminal or in a batch job. These changes might include replacing characters, deleting lines within a file, or adding to the file.

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.

RELATED PUBLICATIONS

The Text Editor user should be familiar with general file structures and NOS time-sharing procedures. For additional information about NOS software, refer to the current versions of the following manuals.

Control Data Publication	Publication Number
NOS Version 1 Reference Manual, Volume 1	60435400
NOS Version 1 Reference Manual, Volume 2	60445300
NOS Version 1 Time-Sharing User's Reference Manual	60435500
NOS Version 1 Terminal User's Instant	60435800
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Network Terminal User's Instant	60455270

DISCLAIMER

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

INTRODUCTION 1

GENERAL

The Text Editor (EDIT) performs data manipulations on a file. EDIT can be used from jobs of time-sharing origin or of batch origin. When at a time-sharing terminal, the user can interact with the Text Editor program or can use the EDIT command to specify a file that contains all the EDIT commands and input data to be used while in the Text Editor. From a batch job, the user supplies all commands and input data through a job deck or specifies a local or permanent file for EDIT to process. This manual is presented from a time-sharing orientation. Any references in this manual to time-sharing, unless otherwise stated, also apply to the Interactive Facility (IAF).

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).
- Remove 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).
- Incorporate the contents of a file into another file (MERGE).
- Obtain a count that reflects the number of times a specified combination of characters or lines 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, STOP).

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.

- Line mode operations are addressed to one or several entire lines of text in the edit file.
- 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.
- 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 $\textcircled{CR}$ is used throughout to denote the carriage return key (or its equivalent).

In the examples, the terminal printout is occasionally expanded to accommodate the commentaries. All user input is lowercase, and Text Editor output is uppercase.

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 initiating Text Editor, general command syntax, and string manipulation procedures.

INITIATING TEXT EDITOR

The user initiates the Text Editor with the EDIT command or control statement. The format 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 If omitted, normal mode is assumed. If a terminal is in ASCII mode in a time-sharing session, the system adds an AS to all EDIT commands when the EDIT statement is entered with no parameters or as EDIT, lfn_1 (order dependent format only). This does not apply to EDIT statements in procedure files or EDIT statements with more than one positional parameter.
lfn_2	The file from which the edit commands are to be read. For a time-sharing session, default is input from the terminal. For a batch job, default is a record in the job deck (INPUT file).
lfn_3	The file on which the output is to be written. The default is OUTPUT.

INTERACTIVE USAGE OF TEXT EDITOR

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

```
EDIT (CR)
```

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 (CR)
```

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.

BATCH USAGE OF TEXT EDITOR

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

For example, suppose a batch job contains a record listing six types of cable assemblies and the amounts on hand. The job calls on Text Editor to produce two listings of specific types. The deck is shown in figure 2-1. The cable list is the second record in the INPUT file. This is copied to a local file and given the name PARTS.

```
CABLES.
USER(XYZ22,ABC11)
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 preceding job

BEGIN TEXT EDITING.

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

Figure 2-1. Batch Job Using Text Editor

The EDIT control statement references PARTS which is rewound by Text Editor. The mode of file processing is normal. The missing parameter after the comma indicates the source default of INPUT. Therefore, the editing commands are taken from the next record in the job deck (following the list of six cables).

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

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

ADDITIONAL CONSIDERATIONS

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).

NOTES

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 or the IAF Reference Manual).

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

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

To change the contents of the edit file, the user must assign the file in write mode. For interactive jobs, EDIT sends the following message when the edit file is not in write mode.

```
EDIT FILE NOT IN WRITE MODE.
DO YOU WISH TO CONTINUE?
```

The user must respond YES (CR) or NO (CR). If the user enters NO, EDIT terminates immediately, without attempting to write on the edit file. If the user enters YES, editing is allowed (useful for just interrogating a file), but EDIT eventually aborts when EDIT attempts to write on the edit file. For noninteractive jobs, EDIT allows editing on a file that is not in write mode but aborts when EDIT attempts to write on the edit file.

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.

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. 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 an edit file is specified by an edit command. The following elements are possible in an edit command.

- Command word

  This is the mandatory first element. It can be any one of the EDIT commands or a short form thereof, as listed in the Command Words section.

- String specification

  A 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 file being sought, added to, or changed. (In the MERGE command, the string is a file name.) The delimiters on each end of the string must be the same character, must not be the character $ or blank, and cannot be used within the string. (The / is arbitrarily used in the formats to designate the delimiting character.)

  A string specification must immediately follow a colon. If two string specifications are included in a command, they must be separated by a comma. The forms of a string specification are:

  ```
  omitted

  :/string/

  :/string/, /string/
  ```

- n parameter

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

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

  omitted

  ;n

  ;*

- Comment

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

Each command, including a possible comment, must be contained within a single line. Each element must appear in the sequence shown. Pressing the carriage return initiates the operation of the command.

Table 2-1 shows an outline of EDIT command formats, and table 2-2 shows examples of the various formats.

TABLE 2-1. EDIT COMMAND FORMATS

Command Word	String Specification	n Parameter	Comment
Identifying name of the EDIT command	omitted	omitted	omitted
	:/string/	;n	$characters
	:/string/, /string/	;*	

TABLE 2-2. EDIT COMMAND FORMAT EXAMPLES

Format	Example
name	RESET
name:/string/	CHANGE:/UNCONDITIONAL GUARANTEE/
name:/string/;n	FIND:XSECTIONX;4
name:/string/;*	DELETE:'THIS IS PREPAID';*
name;n	ADD;12
name;*	LIST;*
name:/string/, /string/	CHANGE:?THIS INFORMATION?, ?BE FOUND?
name:/string/, /string/;n	INSERTS:(DATE DUE: (,)JUNE 6, 1977);5
name:/string/, /string/;*	RS:X1976X, Z1977Z;*
name$comment	RESET$ THE POINTER IS RESET TO THE BEGINNING
name:/string/$comment	DELETE:/$436.00/$THIS DELETES THE LINE CONTAINING SUM
name;*$comment	SET;*$THIS SETS THE POINTER TO END-OF-FILE

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.

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

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

The delimiter character can be used within the string only in response to an ENTER TEXT request. However, if a character identical to the delimiter character appears at the end of a line or is the character before a blank which is at the end of a line, Text Editor may interpret the character as the closing delimiter and the ENTER TEXT request would be terminated.

NOTE

If IAF users enter the cancel line character as the closing delimiter, it must be followed by a space, or the system interprets it as the cancel line command.

Use of the delimiter character within the string definition of a Text Editor command is not allowed and if used, EDIT responds:

```
command      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)

NOTE

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 <u>FORM OF SHORTHAND FOR LONG</u> 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 <u>THE TARGET STRING EXTENDS OVER SEVERAL LINES LIKE THIS ONE.</u>

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 RESET, LINE, LISTAB, CLEAR, NUMBER(S), DEFTAB, and END commands. 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 is 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 this method is used by an IAF user, the closing delimiter cannot be the terminal control character or any other character recognized as a terminal definition command (refer to the IAF Reference Manual). If a carriage return alone is entered on a line, a final blank line is added to the text, and an exit from the enter text mode occurs (that is, a return to command mode).

PROCESSING TERMINAL INTERRUPTS

The time-sharing user may control his edit session with terminal interrupts, which are exercised under three circumstances.

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

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

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

  ```
  DISREGARD PREVIOUS TEXT ?
  ```

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

  ```
      READY.
     ?
  ```

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

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

```
 READY.
?
```

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

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

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

```
INTERRUPT AT LINE n.
```

The output status is then followed by the inquiry:

```
COMMAND CONTINUE?
```

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

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 B contains a summary for all Text Editor messages and requests.

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. Execution of the LIST command does not change the position of the search pointer.

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

```
     -END OF FILE-
```

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

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 adjacent text, beginning at the search pointer. If n=*, all lines from the search pointer 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/, /string2/.
LIST:/string1/, /string2/;n	Prints the first n occurrences of lines or groups of lines containing the ellipsis /string1/, /string2/ (n can equal *).

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 (n can equal *).
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/ (n can equal *).

†The position of the search pointer does not change when using any form of the command.

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

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

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

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.

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

Entry/Reponse

Commentary

```
edit

 BEGIN TEXT EDITING.
? list;*
ØØ1ØØ PRINT"AREA AND PERIMETER OF A SQUARE."
ØØ11Ø PRINT"WHAT IS THE LENGTH OF "
ØØ12Ø PRINT"ONE SIDE (TO THE NEAREST CM)";
ØØ13Ø INPUT S
ØØ14Ø PRINT"THE AREA IS"S^2"SQ. CM."
ØØ15Ø PRINT"THE PERIMETER IS"4*S"CM."
ØØ16Ø END
 -END OF FILE-
```

Print all lines from the search pointer's position to the end of the edit file. The search pointer does not move.

```
? set;4
? l;3
ØØ14Ø PRINT"THE AREA IS"S^2"SQ. CM."
ØØ15Ø PRINT"THE PERIMETER IS"4*S"CM."
ØØ16Ø END
```

Advance the search pointer four lines from its current position. Print the line identified by the search pointer and the next two lines.

```
? list:/cm/;3
ØØ14Ø PRINT"THE AREA IS"S^2"SQ. CM."
ØØ15Ø PRINT"THE PERIMETER IS"4*S"CM."
 -END OF FILE-
```

Print the following three lines containing string /cm/ (only two lines were found).

```
? reset
? list:/cm/;3
ØØ12Ø PRINT"ONE SIDE (TO THE NEAREST CM)";
ØØ14Ø PRINT"THE AREA IS"S^2"SQ. CM."
ØØ15Ø PRINT"THE PERIMETER IS"4*S"CM."
```

Move the search pointer to the beginning of the edit file. Print three lines containing string /cm/.

```
? set:/input/
? l;4
ØØ13Ø INPUT S
ØØ14Ø PRINT"THE AREA IS"S^2"SQ. CM."
ØØ15Ø PRINT"THE PERIMETER IS"4*S"CM."
ØØ16Ø END
```

Advance the search pointer to the first line containing string /input/.

```
? set;-2
? list
ØØ11Ø PRINT"WHAT IS THE LENGTH OF "
? end
 END TEXT EDITING.

 READY.
```

Move the search pointer back two lines and print that line.

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 or if n equals *, 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 the FIND and LINE commands.

<u>Entry/Response</u> — <u>Commentary</u>

```
edit

 BEGIN TEXT EDITING.
? list;*                                        List to end of file.
00100 PRINT"RECTANGLE DRAWING"
00110 PRINT"WHAT IS THE LENGTH AND ";
00120 PRINT"WIDTH (NEAREST CM)";
00130 INPUT L,W
00140 FOR X=1 TO L
00150 PRINT"*";
00160 NEXT X
00170 PRINT
00180 LET R=W-2
00190 IF A=1 THEN 00250
00200 FOR Y=1 TO R
00210 PRINT"*";TAB(L-1);"*"
00220 NEXT Y
00230 LET A=1
00240 GOTO 00140
00250 PRINT"*=1 CM."
00260 END
 -END OF FILE-
? find:/for/                                    Advance search pointer to first line con-
00140 FOR X=1 TO L                              taining string /for/ and print the line.
? list;2
00140 FOR X=1 TO L
00150 PRINT"*";
? line                                          Check the position of the search pointer.
 FILE AT LINE NUMBER      5.
? finds:/*/;4                                   Advance the search pointer to the line
              *                                 containing the fourth occurrence of
? ln                                            string /*/ and print the string.
 FILE AT LINE NUMBER     16.
```

Entry/Response | Commentary

```
? find:/for y/
 PHRASE NOT FOUND.
? reset
? find:/for y/
00200 FOR Y=1 TO R
```

String /for y/ is not found in any of the lines after line 15. Move the search pointer to the beginning of the edit file and try again.

```
? f:/print/,/tab/
00210 PRINT"*";TAB(L-1);"*"
? 1;3
00210 PRINT"*";TAB(L-1);"*"
00220 NEXT Y
00230 LET A=1
```

Advance the search pointer to the next line that contains the ellipsis string /print/, /tab/ and print the line. Start search at present position of search pointer.

```
? fs:/1/;3
             1
```

Advance the search pointer to the line that contains the third occurrence of string /1/ and print the string.

```
? line
 FILE AT LINE NUMBER      14.
? find;2
00250 PRINT"*=1 CM."
? end
 END TEXT EDITING.

 READY.
```

Advance the search pointer two lines and print the line.

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 Text Editor inserts string2 immediately after each of n occurrences of string1, beginning at the search pointer. /string1/, /string2/ is not an ellipsis search string, and the command does not change the position of the search pointer.

†The position of the search pointer does not change when using any form of the command.

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

Entry/Response | Commentary

```
edit,filel
```

Text Editor is called, creating empty working file file1.

```
 BEGIN TEXT EDITING.
? add
```

The file is built using the ADD command.

```
 ENTER TEXT.
? /  the add command can be very
? useful when creating a textual
? file.  /
 READY.
? adds:+file. +
```

Text is added immediately after the first occurrence of the string /file. /.

```
 ENTER TEXT.
? / in fact, it is one
? of the few methods that can be
? used to build a direct access
? file./
 READY.
? list;*
```

List to end of file. The search pointer remains at the beginning of the edit file.

```
  THE ADD COMMAND CAN BE VERY
USEFUL WHEN CREATING A TEXTUAL
FILE.  IN FACT, IT IS ONE
OF THE FEW METHODS THAT CAN BE
USED TO BUILD A DIRECT ACCESS
FILE.
 -END OF FILE-
? inserts:/access/,/ permanent/
```

The string /permanent/ is inserted after the first occurrence of the string /access/.

```
? 1;2
  THE ADD COMMAND CAN BE VERY
USEFUL WHEN CREATING A TEXTUAL
? find;4
USED TO BUILD A DIRECT ACCESS PERMANENT
```

Advance pointer four lines.

```
? as:,file,
 ENTER TEXT.
```

Text is added directly after the first occurrence of the string /file/.

```
? t, providing it is the edit filet
 READY.
? a;*
 ENTER TEXT.
```

Text is added to the end of the file.

```
? =  it is also useful when adding
? text to a previously existing file.=
 READY.
? reset
? 1;*
```

Reset search pointer to beginning of file and list to end of file.

```
  THE ADD COMMAND CAN BE VERY
USEFUL WHEN CREATING A TEXTUAL
FILE.  IN FACT, IT IS ONE
OF THE FEW METHODS THAT CAN BE
USED TO BUILD A DIRECT ACCESS PERMANENT
FILE, PROVIDING IT IS THE EDIT FILE.
  IT IS ALSO USEFUL WHEN ADDING
TEXT TO A PREVIOUSLY EXISTING FILE.
 -END OF FILE-
? a:/existing/
```

Add text after the line containing the string /existing/.

```
 ENTER TEXT.
? /  later it will be demonstrated how to
? use the add command to remove text
? from the string buffer./
```

Entry/Response

Commentary

```
 READY.
? s;7
? l;4
```

Advance search pointer seven lines.
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 or 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
USED TO BUILD A DIRECT ACCESS PERMANENT
FILE, PROVIDING IT IS THE EDIT FILE.
  IT IS ESPECIALLY USEFUL WHEN
ADDING TEXT IN THE BODY OF A FILE.
  IT IS ALSO USEFUL WHEN ADDING
TEXT TO A PREVIOUSLY EXISTING FILE.
  LATER IT WILL BE DEMONSTRATED HOW TO
USE THE ADD COMMAND TO REMOVE TEXT
FROM THE STRING BUFFER.
 -END OF FILE-
? end
 END TEXT EDITING.
READY.
```

Listing of altered file.

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. If the user deletes the line specified by the search pointer, the pointer advances to the next line; otherwise, it does not move.

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 position of the search pointer does not change when using any form of the command.

The following example illustrates the use of the 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).LT.NEXTSEED) GO TO 10
00130 NEWNUM = INCRMENT
00140 20 CONTINUE
00150 END
 -END OF FILE-
? delete:+bignumbr+;*                              Delete every line in file that contains
      2  OCCURRENCES OF PHRASE FOUND.              the string /bignumber/.
? 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
 ENTER TEXT.                                       position of search point (line 4).
? /00050 big = 2.**48
? 00060 fraction = hundreth*big/
 READY.
? blanks:xhundrethx                                Blank first occurrence of string
? l;2                                              /hundreth/.  The pointer remains
00040          = 1./100.                           at line 4.
00050 BIG = 2.**48
? is:/00040 /,/h100/                               Insert string /h100/ immediately after
? l                                                string /00040 /.
00040 H100         = 1./100.
? ds:/        /                                    Delete eight spaces denoted by string
? l                                                /        /.  The pointer remains at
00040 H100 = 1./100.                               line 4.
? blanks:)undreth)
? line                                             Blank first occurrence of string
 FILE AT LINE NUMBER       4.                      /undreth/.
? f;2
00060 FRACTION = H        *BIG                     Advance search pointer two lines and
? adds:/h/                                         list line.
 ENTER TEXT.                                       Add text after first occurrence of
                                                   string /h/.
```

Entry/Response | Commentary

```
? /100/
 READY.
? 1
00060 FRACTION = H100        *BIG
? ds:r        r
```

Remove blanks.

```
? 1
00060 FRACTION = H100*BIG
? r
```

Reset search pointer.

```
? f;13
00140 20 CONTINUE
? ds:/20/,/end/
```

Delete string specified by ellipsis /20/. /end/.
Note: string deleted, not line.

```
? 1;*
00140
 -END OF FILE-
? is:*00140 *,*print,newnum*
```

Insert string /print, newnum/ immediately after string /00140 /.

```
? 1
00140 PRINT,NEWNUM
? a
```

Add text after line indicated by search pointer.

```
 ENTER TEXT.
? /00150 20 continue
? 00160 end/
 READY.
? reset
```

Reset search pointer.

```
? list;*
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-
? end
 END TEXT EDITING.
READY.
```

Altered file is listed to end of file.

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 Ⓒⓡ 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 Ⓒⓡ 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.

†The position of the search pointer does not change when using any form of the command.

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	Replaces the first n occurrences of string1 with string2, beginning at the search pointer.

†The position of the search pointer does not change when using any form of the command.

The following is an example of the CHANGE and RS commands.

Entry/Response | Commentary

```
edit

 BEGIN TEXT EDITING.
? 1;*
```

List to end of file.

```
ØØØ1Ø PROGRAM RANDNUM
ØØØ2Ø MAINSEED = 5**13
ØØØ3Ø NEXTSEED = 5**15
ØØØ4Ø H1ØØ = 1./1ØØ.
ØØØ5Ø BIG = 2.**48
ØØØ6Ø FRACTION = H1ØØ*BIG
ØØØ7Ø IFRAC = INT(FRACTION)
ØØØ8Ø DO 2Ø I = 1,1Ø
ØØØ9Ø INCRMENT = Ø
ØØ1ØØ NEXTSEED = NEXTSEED*MAINSEED
ØØ11Ø 1Ø INCRMENT = INCRMENT+1
ØØ12Ø IF((INCRMENT*IFRAC).LT.NEXTSEED) GO TO 1Ø
ØØ13Ø NEWNUM = INCRMENT
ØØ14Ø PRINT,NEWNUM
ØØ15Ø 2Ø CONTINUE
ØØ16Ø END
 -END OF FILE-
? change
```

Change line indicated by current setting of search pointer.

```
 ENTER TEXT.
? /ØØØ1Ø program random (output)/
 READY.
? rs:+nextseed+,+nextsd+;*
       4  OCCURRENCES OF PHRASE FOUND.
```

Replace each occurrence of the string /nextseed/ with /nextsd/. The search pointer remains at the beginning of the edit file.

```
? f;2
ØØØ3Ø NEXTSD = 5**15
? f;7
ØØ1ØØ NEXTSD = NEXTSD*MAINSEED
? cs:rseedr
```

Change first occurrence of string /seed/ (relative to search pointer) with the string /sd/.

```
 ENTER TEXT.
? /sd/
 READY.
? 1
ØØ1ØØ NEXTSD = NEXTSD*MAINSD
? reset
? cs:/seed/,/15/
```

Change ellipsis string specified by /seed/, /15/. The search pointer remains at the beginning of the edit file.

```
 ENTER TEXT.
? 8sd = 5**13
? ØØØ3Ø nextsd = 5**178
 READY.
? 1;3
ØØØ1Ø PROGRAM RANDOM (OUTPUT)
ØØØ2Ø MAINSD = 5**13
ØØØ3Ø NEXTSD = 5**17
? rs:.incrment.,.inc.;5
? rs:#fraction#,#frac#;2
```

Replace first five occurrences of string /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.
```

To add text to the beginning of the file, it is necessary to use the CHANGE command. Reenter line identified by the search pointer along with new lines.

```
? 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 TEXT EDITING.
READY.
```

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

LOADING AND CLEARING THE STRING BUFFER

The string buffer (described in section 2) can be loaded with the EXTRACT command. Information is transferred from the string buffer to the edit file with the ADD and CHANGE commands. The buffer is not automatically cleared when information is transferred by either of these commands; it is cleared only with the CLEAR command.

EXTRACT COMMAND

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

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. If less than n lines are found, the last line containing the specified string is copied.
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/. If less than n occurrences are found, the last line or group of lines containing the specified ellipsis is copied.

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 COMMAND (CLEAR OR CL)

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

The format is:

```
CLEAR
```

Operand fields are never used with this command.

The following example illustrates the use of the 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 is inserted into the file.

```
 ENTER TEXT.
? $
 READY.
? l;*
     THE EXTRACT COMMAND CAN BE
LINES OF TEXT.
VERY USEFUL IN REARRANGING
LINES OF TEXT.
 -END OF FILE-
```

The EXTRACT command does not delete the information that it copies into the buffer.

```
? s:#lines#;2
? delete
```

Advance search pointer to line containing second occurrence of string /lines/ and delete line.

```
 -END OF FILE-
? r
? a
 ENTER TEXT.
? )used to restructure individual)
 READY.
? es:+rearranging+
```

Copy string /rearranging/ to string buffer.

```
? a;*
```

Add contents of string buffer to end of file.

```
 ENTER TEXT.
? $
 READY.
```

The text from previous EXTRACTs remains in the string buffer, and subsequent EXTRACTs cause 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
 -END OF FILE-
? clear
```

Clear string buffer.

```
? s:/very/
? d;*
 -END OF FILE-
? es:*restructure*
```

Copy string /restructure/ to string buffer; command is relative to position of search pointer.

```
 -END OF FILE-
? reset
? es:*restructure*
? changes:kindividualk
 ENTER TEXT.
```

Change string /individual/ to contents of string buffer.

```
          Entry/Response                                   Commentary

? $
 READY.
? l;*                                          List to end of file.
     THE EXTRACT COMMAND CAN BE
USED TO RESTRUCTURE RESTRUCTURE
LINES OF TEXT.
 -END OF FILE-
? ds:/restructure/                             Delete first occurrence of string
? a;*                                          /restructure/.
 ENTER TEXT.
? (     remember that the string
? buffer is not cleared after an
? add or change $ command.
?      to remove text from the string
? buffer, use the clear command.(
 READY.
? clear                                        Clear string buffer.
? es:# string#,#buffer, #                      Note the difference between these two
? ds:# string#,#buffer#                        ellipsis strings.
? adds:.that the.                              Contents of string buffer is inserted
 ENTER TEXT.                                   after string /that the/.
? $
 READY.
? 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                                        Clear string buffer.
? 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-
? 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.

NOTE

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, the new line length, must be specified and must be in the range 6 through 150. The user should make sure that n is smaller than the page width.

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 because 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 position of the search pointer does not change when using any form of the command.

The following example illustrates the use of the 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
? width;2Ø
? l;*
```

Truncate edit file at line 4.
Set width indicator to 20.
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.
 -END OF FILE-
```

The WIDTH command does not affect the file unless an ALIGN command is used.

```
? 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
```

Align line specified by search pointer.

```
? l;4
     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                                         Set width to 62 and align entire file.
? 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#                             Blank string /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                                      Align four lines from search pointer.
? 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 ALIGN COMMAND.
 -END OF FILE-
? end
 END TEXT EDITING.
READY.
```

TABULATION COMMANDS

The DEFTAB, TAB, and LISTAB commands 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, and t_i is greater than 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 preestablish 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

```
get,matrix

READY.
edit,matrix

 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.
? 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-
? f:/#/
#THIS#IS#TAB#LINE#1.
? extract;4
? r
? d;*
 -END OF FILE-
? a
 ENTER TEXT.
? +l;*
? f:/defgh/
? deftab:/#/
? tab:/6,18,22,40,45/
```

Commentary

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.

The added text has the tab character # included in each line. However, the user has failed to enter tabulation specifications.

A listing of the edited file reveals the oversight. The tab character is still there instead of the desired spacing.

Position to the first line of inserted text and extract the four lines of new text. This is put in the string buffer.

The entire original file is deleted. Only the name MATRIX remains.

An input file of edit directives is built.

Entry/Response

Commentary

```
? add
? l;*
? end+
 READY.
? l;*
L;*
F:/DEFGH/
DEFTAB:/#/
TAB:/6,18,22,40,45/
ADD
L;*
END
 -END OF FILE-
? f:/add/
ADD
? add
 ENTER TEXT.
? $
 READY.
? l;*
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.

The text in the string buffer is added to the input file.

```
? rs:/#/,/+#/
? rs:/#4./,/#4.+/
? r
? 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-
? end
 END TEXT EDITING.
READY.
```

The necessary delimiters are added.

The input file is checked with a listing.

Exit is made from Text Editor in order to use control statements.

```
rename,inl=matrix
```

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,,inl
```

Text editing is reinitiated with the full edit command. The edit file is matrix, the mode is normal, and the input file is inl.

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 the TAB, DEFTAB, and LISTAB commands.

```
          Entry/Response                                  Commentary
edit
 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:/#/                                      Defines the character # as the tab
? tab:/5,1Ø,2Ø/                                   character with stops at 5, 10, and 20.
? add;*
 ENTER TEXT.
? /a#b#c#d
? e#f#g#h/
 READY.
? list;*
  THE DEFTAB AND TAB COMMANDS
ARE EFFECTIVE ONLY IF GIVEN PRIOR
TO AN ENTER TEXT REQUEST. THUS,
THE FOLLOWING WILL NOT BE TABULATED.
1#2#3#4
5#6#7#8
NOW DEFINE A TAB CHARACTER
AND A SET OF TAB STOPS.
A   B    C          D
E   F    G          H
 -END OF FILE-
? listab
 TAB STOPS    5    1Ø    2Ø
? deftab                                          Clears previous tab character.
? tab                                             Clears existing tab stops.
? lt
 TAB STOPS NONE.
? end
 END TEXT EDITING.
READY.
```

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 local 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 A). 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 or the IAF 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 (CR) 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 the MERGE command.

Entry/Response

```
lnh,f=txt2

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

READY.
edit

 BEGIN TEXT EDITING.
? add
 ENTER TEXT.
? /  this file is being built
? using the text editor add
? command./
 READY.
? 1;*
  THIS FILE IS BEING BUILT
USING THE TEXT EDITOR ADD
COMMAND.
 -END OF FILE-
? merge:/txt2/;*
? 1;*
  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/
? 1;*
  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.
READY.
```

Commentary

Before entering Text Editor, time-sharing commands are used to list temporary file txt2, which is a copy of a permanent file.

Temporary primary file xyz is created, releasing temporary file txt2.

Enter Text Editor with empty primary file xyz as the edit file.

Merge permanent file txt2 to end of edit file.

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

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. The string can be either single phrase or ellipsis.

The following example illustrates the use of the NUMBER command.

```
                 Entry/Response                          Commentary
edit

 BEGIN TEXT EDITING.
? 1;*                                               List to end of file.
ØØØØ5***THIS PROGRAM GENERATES
ØØØØ6*  1Ø RANDOM NUMBERS BETWEEN
ØØØØ7*  1 AND 1ØØ.
ØØØ1Ø PROGRAM RANDOM (OUTPUT)
ØØØ2Ø MAINSD = 5**13
ØØØ3Ø NEXTSD = 5**17
ØØØ4Ø H1ØØ = 1./1ØØ.
ØØØ5Ø BIG = 2.**48
ØØØ6Ø FRAC = H1ØØ*BIG
ØØØ7Ø IFRAC = INT(FRAC)
ØØØ8Ø DO 2Ø I = 1,1Ø
ØØØ9Ø INC = Ø
ØØ1ØØ NEXTSD = NEXTSD*MAINSD
ØØ11Ø 1Ø INC = INC+1
ØØ12Ø IF((INC*IFRAC).LT.NEXTSD) GO TO 1Ø
ØØ13Ø NRAN = INC
ØØ14Ø PRINT 3Ø, I, NRAN
ØØ145 3Ø FORMAT(I2,2X,I4)
ØØ15Ø 2Ø CONTINUE
ØØ16Ø END
 -END OF FILE-
? number                                            Request line count from search pointer
       2Ø LINES TO EOF.                             to end of file.
? number:/nextsd/                                   Request count of number of lines con-
     3  OCCURRENCES OF PHRASE FOUND.                taining string /nextsd/.
? numbers:/nextsd/                                  Request count of number of occurrences
     4  OCCURRENCES OF PHRASE FOUND.                of string /nextsd/.
? s;1Ø
? n                                                 Request line count from search pointer
       1Ø LINES TO EOF.                             to end of file.
? ns:/inc/,/Ø/                                      Request count of occurrences of
     4  OCCURRENCES OF PHRASE FOUND.                ellipsis string /inc/, /0/.
? lists:/inc/,/Ø/;*
     INC = Ø                                        Actual occurrences of ellipsis string
        INC = INC+1                                 /inc/, /0/ are listed.
Ø         INC*IFRAC).LT.NEXTSD) GO TO 1Ø
             INC
Ø
? n:/inc/,/Ø/                                       Request count of number of lines con-
     3  OCCURRENCES OF PHRASE FOUND.                taining ellipsis string /inc/, /0/.
? r
? ns:/mainsd/                                       Request count of occurrences of string
     2  OCCURRENCES OF PHRASE FOUND.                /mainsd/ in file.
? ns:/print/                                        Request count of occurrences of string
     1  OCCURRENCES OF PHRASE FOUND.                /print/ in file.
? end
 END TEXT EDITING.
READY.
```

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 that was in use before the user entered Text Editor.

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).

STOP COMMAND (STOP)

The user can 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 is unpredictable, and all output files can be lost.

The user might use the STOP command to end an edit session in either a terminal session or a batch job if he wishes only to examine the contents of a file where the file is not required after the edit session.

GLOSSARY A

ASCII Mode	Use of the American National Standard Code for Information Interchange; 128-character set. It includes both uppercase and lowercase letters.
Batch Job	Instructions and data submitted as a complete unit without further user intervention. The job can be punched on cards or created and submitted from a terminal.
Command	A set of characters used to signify a specific set of instructions.
Comment	In Text Editor, a comment consists of a set of characters the user adds to an EDIT command for the user's own information. A comment must begin with a $, must be contained on one line, and has no effect on the command.
Delimiter	A character used to define a string or text that is entered. It can be any nonblank character except $. In #ADD#, the # is the delimiter.
Direct Access File	A permanent file that can be attached to the user's job. All changes to this file are made on the file itself rather than a temporary copy of the file.
Edit File	The single record file on which Text Editor is working. It can be the primary file, a temporary file, or a direct access file and is specified with the EDIT command. The file line has a limit of 150 (6-bit) characters. Lines longer than 150 characters are truncated.
Ellipsis String	A string whose delimiters are strings. The first string delimiter does not have to be on the same line as the last string delimiter.
-End of File-	A boundary within a sequential file. When in Text Editor, -end of file- signifies the end of the edit file.
File	A set of information that begins at beginning-of-information (BOI), ends at end-of-information (EOI), and is referred to by a local file name.
Indirect Access File	A permanent file that is accessed only by making a temporary copy of the file (GET, OLD, or LIBRARY command). It is created or altered by saving or substituting the contents of an existing temporary file (SAVE or REPLACE command).
Interactive Facility (IAF)	Software product which allows a time-sharing terminal user to enter commands and to communicate with an executing program.

Interrupt	To stop the execution of an EDIT command, the user enters the interruption sequence applicable to his terminal.
Job	In time-sharing mode, a job consists of all computer related activity associated with a user from the time he logs in to the time he logs off. A batch job consists of instructions and data that is submitted as a complete unit.
Line Mode	The mode in which edit operations are performed with a line of the edit file as the basic unit of operation.
Local File	Any file currently associated with a job. Local files include all temporary files and all attached direct access files.
Merge File	A local or permanent file that the user inserts into the edit file.
Normal Mode	Use of the standard 64- or 63-character set where all lowercase letters are converted to uppercase.
Output File	The file on which the system writes information to the user. Unless another file name is specified, the OUTPUT file is printed at the terminal.
Permanent File	A file which is created and stored by the user so that it can be used at some time in the future, by another user, or more than once in a specific terminal session.
Primary File	Any temporary file created with the OLD, LIBRARY, PRIMARY, or NEW command. There can be only one primary file at a time among the user's local files, but the user can change the primary file during a terminal session.
Record	A unit of information within a file. The edit file contains only one record.
Search Pointer	A position indicator which identifies the line at which EDIT starts processing a command.
String	A set of characters bounded on each end by a nonblank character (called a delimiter).
String Buffer	A temporary storage area for information that is to be moved within the edit file.
String Mode	The mode in which edit operations are performed with a character string as the basic unit of operation.
Tab Character	A character that represents a specific indentation when read by Text Editor.
Tab Stop	The indentation number for the tab character.

Temporary File	This file is either a file created by the user that is not a permanent file, or it is a copy of a file that already exists in the system. All temporary files no longer exist once they are returned to the system (either specifically or at user logout).
Text	Information that is in a file. Text can consist of program statements, data, or any set of characters.
Time-Sharing Commands	Commands that allow the user to manipulate files, select terminal characteristics, and process jobs. Some examples are LIST, BATCH, SAVE, and NORMAL.
Time-Sharing Subsystem	Mode of communication for a time-sharing user. The subsystems available are BASIC, BATCH, EXECUTE, FORTRAN, FTNTS, and NULL.

EDIT MESSAGES

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

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
BEGIN TEXT EDITING.	Informative command indicating text editor is ready to begin accepting commands.	None.	EDIT
COMMAND CONTINUE?	Text editor inquiry as to whether or not an interrupted command should continue to be processed.	Respond with YES or NO.	EDIT
CONTROL CARD ERROR.	An illegal or invalid parameter was specified on the control statement.	Correct control statement and retry.	EDIT,
DISREGARD PREVIOUS TEXT?	Text editor inquiry as to whether or not the text that has been entered in response to a text-entering command should be retained or discarded.	Respond with YES or NO.	EDIT
EDIT FILE NOT IN WRITE MODE. DO YOU WISH TO CONTINUE?	The user did not assign the file in write mode.	Enter NO to terminate the job step. Enter YES to edit the file. The job step will abort when EDIT attempts to write on the edit file.	EDIT
-END OF FILE-	Informative message indicating that text file is positioned at end-of-file or end-of-file was encountered during LIST or FIND command.	None.	EDIT
END TEXT EDITING.	Informative message indicating termination of EDIT session.	None.	EDIT
ENTER TEXT.	Requests entry of new or replacement text for ADD(S) or CHANGE(S) command.	Enter line(s) of text to be processed (enclosed by delimiters).	EDIT
ENTER TEXT FILE NAME.	This message is issued to request entry of the text file name if it has not been passed with the Text Editor call and input is from the terminal (as specified by the I=parameter).	Enter text file name.	EDIT
ENTER *YES* OR *NO*.	User responded to a question with something other than yes or no.	Enter yes or no.	EDIT
FILE AT LINE NUMBER n.	Text file is currently positioned at line number n.	None.	EDIT
ILLEGAL COMMAND.	The characters entered do not constitute a valid command.	Ensure the accuracy of the command entry.	EDIT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL DELIMITER.	An invalid delimiter was used in response to the ENTER TEXT request from a local or remote batch job.	Retry using correct delimiter.	EDIT
ILLEGAL DELIMITER - REENTER TEXT.	An invalid delimiter was used in response to the ENTER TEXT request from a time-sharing job.	Retry using correct delimiter.	EDIT
ILLEGAL FILE NAME.	The file name passed with the text editor MERGE command is illegal.	Specify legal file name.	EDIT
IMPROPER TRUNCATION.	Length specified in text editor LENGTH;n command is equal to or greater than previous length specified.	Specify length less than that previously specified.	EDIT
INTERRUPT AT LINE n.	Informative message indicating the current position of an interrupted command.	None.	EDIT
MERGE ERROR, SECONDARY FILE EMPTY.	One of the following conditions exists. - The file to be merged with the edit file is empty. - The file to be merged does not exist. - The file to be merged is a direct access file that is not local to the job.	Verify merge file.	EDIT
PHRASE NOT FOUND.	The specified search string was not found.	None.	EDIT
READY.	Informative message indicating next command can be entered.	None.	EDIT
RESERVED FILE NAME.	A reserved file name was incorrectly used.	Choose a nonreserved file name.	EDIT
RESERVED FILE NAME.	This message is written to the output file when a reserved or dupilicate file name is used in the MERGE command. Editing continues with the next command. If a reserved or duplicate file name is detected on the EDIT control statement or after entering the EDIT file name (if none was specified on the control statement), the job step is aborted and this message is issued to the user's dayfile.	Choose a nonreserved file name.	EDIT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
TAB STOPS NONE.	No tab stops are currently established.	None.	EDIT
TAB STOPS t1 t2 ... tn	Text editor tab stops issued in response to LISTAB command.	None.	EDIT
n LINES TO EOF.	Informative message indicating number of lines in text file before end-of-file.	None.	EDIT
n LINES TO INTERRUPT.	Informative message indicating the number of lines that were processed before the user terminated a command.	None.	EDIT
n OCCURRENCES OF PHRASE FOUND.	End-of-file was encountered before number of iterations specified in command were completed.	None.	EDIT
command SYNTAX ERROR.	Improper syntax used with text editor command.	Retry using correct syntax.	EDIT

INDEX

COMMENT SHEET

MANUAL TITLE CDC Text Editor Reference Manual

PUBLICATION NO. 60436100 REVISION G

FROM: NAME:

BUSINESS ADDRESS:

COMMENTS:

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

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

FOLD ON DOTTED LINES AND STAPLE

AA3419 REV. 1/79 PRINTED IN U.S.A.

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

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