EDIT2 is a compiled BASIC program for use on the HP 3000 Computer System. This reference manual defines and describes the various commands and files that constitute EDIT2, and instructs the reader in their use.

The manual is organized by the various kinds of tasks performed by EDIT2. Sections are divided into the following groups:

Section I.	Introducing EDIT2/3000
Section II.	The Editing Environment
Section III.	General Operating Commands
Section IV.	Input Commands
Section V.	Text Modifying Commands
Section VI.	Formatting Commands
Section VII.	File Oriented Commands

A feature of special note: This manual was produced with the program it describes -- EDIT2 -- and is an example of the kind of documentation EDIT2 can generate.

Third Edition June 1978

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

CONVENTIONS USED IN THIS MANUAL

[]	An element inside brackets is optional. Several elements stacked inside a pair of brackets means the user may select any one or none of these elements.
{}	When several elements are stacked within braces, the user must select one of these elements.
()	Parentheses are used in some commands to set off a particular parameter; if that parameter is used, then it must be enclosed by parentheses.
RETURN	The word RETURN or return (italics) indicates use of the RETURN key on the terminal.
CONTROL-Y	CONTROL-Y means pressing the CONTROL (CNTL) key and the Y key at the same time.
COMMAND	Command names are shown throughout the text of the manual in upper case characters. When using EDIT2 you may of course enter commands in upper or lower case characters as you prefer.

CONTENTS

SECTION I: Introducing EDIT2

SECTION II: Operating EDIT2

CONTENTS

CONTENTS

SECTION V: Text Modifying Commands

CONTENTS

WHAT IS EDIT2?

EDIT2 is an interactive text processing program that permits you to edit and format text from a terminal. Lines, strings, and words of text can be added, deleted, replaced, changed, stored, retrieved, justified, copied, and moved at your command.

EDIT2 can be used to prepare and manipulate text such as contracts, proposals, correspondence, technical manuals, and reports.

All the capabilities necessary for efficient editing of text or source language programs through a modern on-line computer system are made available through EDIT2. It makes full use of the multiterminal, mass storage, and high speed input/output capabilities of the HP 3000 Computer System to reduce the time it takes to produce a final version of a document or source language program. Multiple versions of a program or document can be stored and instantly retrieved for further revision.

In order to use EDIT2 you do not have to learn a complicated command language. The commands provided by EDIT2 are easy to use, and tutorial assistance is provided by EDIT2 whenever

In addition, EDIT2 provides extensive formatting capabilities including line filling, hyphenation, and justification.

HOW DOES EDIT2 WORK?

EDIT2 is an interactive program. This means that you and EDIT2 correspond with each other in a conversational fashion through a terminal. You enter commands and text at your terminal, and EDIT2 responds by performing the action commanded on the text provided. In addition, EDIT2 may prompt you for more information or notify you that it found an error in the data you entered.

To illustrate: when you tell EDIT2 that you want to add text to your file, EDIT2 asks you for the text to be added line by line, tells you if the text violates the boundaries of your file, and if no error is found adds the text to the file as you instructed. Similarly, when you tell EDIT2 that you want to change your text, for example to change all "xx" to "yy" in your text, EDIT2 makes the change and shows you the lines of text affected.

When you want a printed copy of your text, you command EDIT2 to print the text in the format you desire and EDIT2 complies via the printing device you select. Or if you want to store your text for later editing or use, EDIT2 creates a file for you and copies the text into the file at your command.

The types of commands that you may enter, and the kinds of tasks which EDIT2 can perform are varied. But the interactive relationship between you and EDIT2 remains the same. That is how EDIT2 works.

EDIT2 FEATURES

With EDIT2 it is possible to:

- Create and build a file for text-in-progress, called a WORK file, by entering commands and lines of text from a terminal.
- Copy the contents of the WORK file into a permanent storage file called a TEXT file.
- Copy or merge a TEXT file back into the WORK file for further editing.
- Change all occurrences of a character, word, or string in the WORK file to another character, word, or string with just one command.
- Delete text from the WORK file.
- Locate a string or all occurrences of a string in the WORK file.
- Move, copy, or renumber portions of text in the WORK file.
- Insert text into the WORK file.
- Display all or a portion of the WORK file at the terminal, or at another output device.
- Interactively modify any line of text in the WORK file.
- Define commands to perform EDIT2 tasks that you use frequently.
- Execute pre-stored EDIT2 commands from a file (called the USE file).
- Fill and justify lines in the WORK file.
- Produce fully formatted output with dynamic page heading information.
- Save and restore an editing environment tailored to a specific editing task.

o Append text to the WORK file.

EDIT2 also provides the following services to users.

o Automatic margin control on input.

o Tabulation for input on HP 2640 series terminals.

o Help when you don't know how to respond to an EDIT2 prompt.

o Explanatory text when you don't understand the function of a command.

EDIT2 FILE DEFINITIONS

THE WORK FILE

The WORK file contains the text in a form suitable for editing. Text not undergoing work may be stored in a TEXT file. To minimize the possibility of accidentally destroying a stored version of text, EDIT2 permits change to text only when it is in the WORK file. Thus when a work file is created and text is added, or when another file is copied for modification, the text is written into the WORK file and all modifications are performed there. Similarly, text to be modified must be brought into the WORK file before any editing can be performed.

Only one user in an account may access any given WORK file with EDIT2. Once the WORK file has been specified it is not available to other users. However, additional WORK files may be created as needed.

The time required to copy text from the WORK file to a permanent file or vice versa is proportional to the amount of text in the file (not to the size of the file). When documents become large (for example, greater than 500 to 1000 lines), the time for copying becomes significant. Therefore, it is suggested that you make use of natural boundaries that occur in documents such as chapters and sections to break the document into reasonable size sections. Such segmentation of the WORK file also serves to speed the operation of certain EDIT2 commands such as CHANGE and FIND.

You may choose to always keep your documents in separate WORK files and never store them to other files (using the KEEP command). This will give you very fast access to them for editing because they don't have to be copied into a WORK file; it also means that you could loose the entire document by a careless action. A somewhat safer mode of operation is to store the

document periodically (e.g., after each significant revision); then if you do something that has disastrous effects in the WORK file, you can restore an earlier version.

TEXT FILES

A permanent copy of your document may be stored in a TEXT file. When the document is ready to be stored, you use an EDIT2 command to copy it from the WORK file to a TEXT file. If further editing is necessary, the document is copied from the TEXT file back into the WORK file when you are ready to perform the edit. Following editing, the original version in the TEXT file can be replaced by the newly edited version, or a separate TEXT file can be used for the edited version of the document. In the latter case, both the original and the edited versions remain available.

Each record in a TEXT file corresponds to one line in the WORK file. If you specify that the line numbers are to be saved with the lines, the numbers are converted to ASCII data and affixed to the beginning of the line of text. One blank character separates the number from the text. The line numbers may also be affixed to the end of the lines if explicitly directed to do so.

USE FILES

USE files enable EDIT2 to perform in a "batch-like" mode. A USE file is an unnumbered text file which contains responses to EDIT2 prompts.

When the USE command is entered, EDIT2 reads records from the file and interprets their content as replies to prompts. Care must be taken in the construction of USE files to ensure that the appropriate number of responses are available at each juncture. This is especially true when replies are being supplied for normally interactive commands such as MODIFY.

USE files may reference other USE files but they may not be nested, i.e., EDIT2 will not return to other than the beginning of previously referenced USE file.

OTHER FILES

During the operation of some EDIT2 commands, temporary files are automatically created in your account. Such files are given names derived by EDIT2 from the following format:

EDnnnZ (nnn is the terminal port number + 100)

These temporary files remain in your account until you terminate your session with EDIT2. Upon termination, these files are automatically purged.

You should not assign file names constructed with the format shown above to other files in your account. The inappropriate use of such file names may result in EDIT2 not being able to execute certain commands, and in the loss of data, as EDIT2 purges files with such names automatically at the end of each session. If EDIT2 is terminated incorrectly, such termination interferes with the purging of these temporary files which may then have to be purged manually.

EDIT2 COMMANDS

A summary of EDIT2 commands is presented in Table 1-1. Included is the command name, the purpose of the command, and the number of the paragraph in this manual where a complete description of the command can be found.

TABLE 1-1. EDIT2 COMMANDS

COMMAND	PURPOSE	PAGE
ADD	Used to enter text into the WORK file from the terminal or from another file.	4-1
CENTER	Centers the contents of a line between LBOUND and RBOUND.	6-9
CHANGE	Used to alter specific portions of the WORK file contents.	5-6
COMPRESS	Replaces multiple blanks in the text with single blanks.	6-7
COPY	Copies text from one location in the WORK file to another.	5-17
DEFINE	Used to name and assign a character value to a pseudo command or command argument.	3-3
DELETE	Used to remove text from the WORK file.	5-10

COMMAND	PURPOSE	PAGE
DISPLAY	Displays a message at the terminal from the USE file.	7-11
EXIT	Terminates your interaction with EDIT2.	3-2
EXPLAIN	Displays summary descriptions of EDIT2 commands at the users request.	3-2
FILL	Fills each line in the specified range with whole words (if possible) so lines in the file contain as many words as possible.	6-1
FIND	Used to locate a specified position in the WORK file.	2-30
GET	Exits the current WORK file and enters the specified WORK file without leaving EDIT2.	7-1
HELP	Same as EXPLAIN.	3-2
INSERT	Inserts character strings or lines of text into the WORK file at a specified position.	4-8
JUSTIFY	Adjusts the contents of a line to exactly fill the space designated as one line.	6-5
KEEP	Saves the contents of the WORK file in a separate file for later use.	7-2
LIST	Displays all or a portion of the WORK file at your terminal.	3-5
MERGE	Merges numbered text from a file with text already in the WORK file.	7-6
MODIFY	Interactively modifies text in the WORK file line by line through the use of three subcommands: insert, replace, and delete.	5-1
MOVE	Used to move text from one location to another in the WORK file.	5-14
PRINT	Used to print the final, fully formatted version of your document.	6-11
RENUMBER	Used to renumber lines of text in the WORK file.	5-12
REPLACE	Used to replace lines of text in the WORK file.	4-5

COMMAND	PURPOSE	PAGE
SET	Used to establish the environment and various optional conditions relative to the WORK file.	2-22
SHOW	Displays the status of any or all EDIT2 options.	2-29
TEXT	Copies the contents of a TEXT file into the WORK file.	7-4
USE	Designates a file to be used as a source of input to EDIT2 prompts.	7-8
//	Used to terminate an on-going action. Causes the same result as pressing CONTROL-Y.	3-1
//ASK	Used to prompt at the terminal from a USE file.	7-12
//EXIT	Terminates your interaction with EDIT2.	3-2
//HELP	Used to obtain helpful information about the current operation.	3-1
//MODIFY	Used to modify the line just added.	4-4

THE EDITING ENVIRONMENT

EDIT2 resides in the System Public Library account PUB.SYS and can be run from any account on the system. Procedures for initiating and terminating an interactive session are described briefly in this section; however, more extensive descriptions can be found in the HP 3000 Guide for the Terminal User.

In addition to information regarding terminal sessions, this section describes the conventions and rules used to enter commands, correct typing errors, and perform similar EDIT2 functions. It also covers three commands. The SET and SHOW commands control the editing environment. The FIND command is used to illustrate the use of the position and range specifications used in many other EDIT2 commands.

INITIATING AN EDIT2 SESSION

To initiate an interactive session, you must:

- Log on
- Run EDIT2.PUB.SYS
- Specify the name for your work file.

LOGGING ON

Once a connection is established between your terminal and the computer, you log on by entering your assigned user name. For example,

```
:HELLO HENRIETTA.EDITOR
```

If you are using an HP 2640 series terminal your TERMTYPE should be set to 10 or 11. If TERMTYPE is normally set to some other value, add ";TERM=10" to your HELLO command. This will ensure that features such as tabs, cartridge tapes, and terminal-connected printers will be operational. This also lets EDIT2 use display features not supported for other terminals. EDIT2 verifies that you actually have a 2640 series terminal and if not, it will not use, or permit you to use, certain features.

If the log on is successful, the computer responds by printing a return message, usually the session number, day, date, and time, and the version of the system as follows:

```
SESSION NUMBER = #S32
FRI, NOV 9, 1977,   8:45 AM
HP32002B.00.68
:
```

THE EDITING ENVIRONMENT

You are now ready to run EDIT2.

RUNNING EDIT2

Use the RUN command to run the program EDIT2, as demonstrated below:

```
:RUN EDIT2.PUB.SYS
```

Once running, EDIT2 prompts for the name of your WORK file. Respond with the appropriate name as shown:

```
WORK file? MYWORK
```

EDIT2 then looks for the WORK file specified. If it does not exist one will be created for you with the name you gave and you will be prompted for the maximum number of lines you wish to be able to work with this work file. You should allow sufficient room for your document plus half again as many lines for additions as there is no way to lengthen the work file. When the file has been created, EDIT2 displays the message: 0 LINES OF TEXT IN YOUR WORK FILE and then issues the command prompt (>>) to request your first command.

If the WORK file is not newly created, the environment previously in effect for the file is automatically restored. In this case, EDIT2 displays the number of lines currently in the file and displays those SET parameters which do not match standard (default) values.

Other file conditions may result in appropriate messages from EDIT2. If the file is not a EDIT2 work file, EDIT2 issues the message: filename IS NOT AN EDIT2 WORK FILE, and again asks for the name of a WORK file.

STOPPING EDIT2

There is only one correct way to stop interaction with EDIT2 and return to the HP 3000 command interpreter (: prompt). Simply type:

```
//EXIT or EXIT
```

The command //EXIT may be entered in response to the command prompt (>>), the text prompt (:), the number sign prompt (#), or the WORK file? prompt. EXIT may only be entered in response to the command prompt (>>).

Any other method of terminating EDIT2 interaction, such as pressing the BREAK key and typing ABORT, results in a disorganized WORK file. If this happens you will be notified the next time you specify that file as your WORK file. EDIT2 automatically attempts to recover the entire contents of your WORK file, and notifies you that such a recovery is in progress.

Following an improper exit, EDIT2 automatically restores your WORK file to the fullest extent possible; however, under certain circumstances lines may be lost (usually in the vicinity of the last addition or revision) or may be assigned different line numbers during the recovery process (especially if stopped during RENUMBER). When recovery is complete it is advisable to list the contents of the WORK file to verify the results before proceeding.

You may terminate the recovery process by pressing CONTROL-Y before you receive the first advisory message about the number of lines recovered. This clears the work file and you may then copy in your own backup TEXT file.

Once you have typed EXIT and returned from EDIT2 to the HP 3000 Operating System, you log off by typing:

```
BYE
```

All connection with the computer is now ended. To use EDIT2 again, you must log on and run EDIT2 as described above.

EDIT2 CONVENTIONS

The following rules and conventions apply to the general operation of EDIT2 and should be read and understood prior to using EDIT2.

ENTERING DATA

Each line of text must terminate with a carriage return (RETURN key). Data cannot be entered while the terminal is typing a response from the computer. EDIT2 prompts for commands with two greater than signs (>>), for text with a colon (:), and for line numbers with a number sign (#).

THE EDITING ENVIRONMENT

ERASING CHARACTERS

Typing H while pressing the CONTROL (CNTL) key directs
computer to ignore the previous character. Each time
is pressed, one previously-typed character is ignored.
terminal will print a backarrow (<-) or an underline c
(_), or simply backspace each time CONTROL-H is presse
BACK SPACE key may be used on HP 2640 series terminals
CONTROL-H.

ERASING LINES

CONTROL-X directs the computer to ignore the entire li
input. Three exclamation marks (!!!) are then printed
of the line, and the computer proceeds to a new line.
is not repeated; simply type command or text correctly

THE BREAK KEY

The BREAK key may be used to interrupt EDIT2 to commun
the system, however this is normally not necessary as
permits you to use most system commands directly by en
colon (:) and then the MPE command. To illustrate:

```
>>:LISTF
   or
>>:TELLOP Please change to narrow paper on LOWLP.
```

NO EDIT2 commands may follow an MPE command.

THE FUNCTION OF CONTROL-Y

CONTROL-Y is used to direct EDIT2 to stop its current
and issue a command prompt (>>). CONTROL-Y may also b
interrupt a listing, terminate an undesired operation,
generally to signal that you are ready to use another
Each command description further defines the effect of
during its execution.

When entering commands, CONTROL-Y will cause the comma
be ignored; four exclamation marks signify that effect
2640 series terminals, the command line is erased.

// COMMAND

The // command may be entered in response to a text prompt (:) or a number sign prompt (#). Like CONTROL-Y, the // command is used to stop an operation. Unlike CONTROL-Y, the // command must be entered as the first and only entry on a line followed immediately by pressing RETURN. It may not be typed at the end of a line of text.

// is intended for use primarily in a USE file to signal EDIT2 to terminate one command and proceed to the next.

TIME-OUT WARNINGS

When EDIT2 judges an operation to be potentially dangerous (to the contents of your work file or a text file, for instance) it issues a warning in the form of a statement that usually contains the words "about to". For example:

```
>>DELETE ALL
WORK file about to be cleared._
```

The cursor or print head remains at the end of the statement for the amount of time specified by SET WARNTIME. Press RETURN to cancel the impending action. If you do not press RETURN before the time limit expires, the stated action is carried out. If WARNTIME is set long enough, you may type any character, except "N", (e.g., OK) followed by RETURN to approve the action before the time limit. Typing anything that begins with an "N" will cancel the operation just as RETURN alone does.

ERROR and //HELP MESSAGES

An appropriate message is given whenever an error is detected or //HELP is used. When you are using an HP 2640 series terminal, the cursor remains at the end of the message until you press RETURN. When you press RETURN, the message and, for errors, the offending input line, is erased and the prompt is repeated. If you wish to preserve the message on the screen just type any character (one is sufficient) before pressing RETURN or just press CONTROL-Y.

MULTIPLE COMMANDS

When EDIT2 prompts with the command prompt, you may enter one or more commands. Use a semicolon (;) to separate one command from

another. For example, three commands are entered on the following line:

```
>>FIND "discard"; CHANGE "discard" TO "throw away"; LIST ALL
```

ENTERING LONG LINES OF DATA

It is possible to construct lines of commands or text longer than you can type in one line on your terminal. If your terminal has the automatic carriage return/line feed feature, you may continue to type lines up to 255 characters long. Otherwise, or if you prefer, you can signal EDIT2 that you want to continue a line by typing an ampersand (&) as the last character of a line. In this event you will be prompted with four question marks (????) on the next line to indicate that the continuation of the line is anticipated.

Under some circumstances, you may be forced to use the ampersand convention even if your terminal is capable of producing an automatic carriage return/line feed. This depends on the length of the buffer allocated to the port you are using. Typically the buffer is able to accept fewer than 255 characters before it overflows. When overflow occurs the computer automatically generates a carriage return. In such a case the "ampersand convention" must be used for long lines to avoid suprise.

You should always use the ampersand to enter long lines when using a 2640 series terminal otherwise the erase that occurs in some commands will occur on the wrong line.

WORKING WITHIN BOUNDS

EDIT2 provides two parameters in the SET command called LBOUND and RBOUND which define the left and right column bounds of the text entered. Generally, these bounds are used as margins for operations on text while it is in the WORK file. However, unlike margins on a typewriter, these bounds cannot be violated when you are creating or changing lines of your document.

Normally, LBOUND is set at column one so it has no genuinely restrictive effect. RBOUND is set as a right margin and does limit text. Input commands such as ADD, REPLACE, and INSERT will not normally accept text beyond RBOUND. However, there is another parameter of the SET command called AUTOMARGIN which, when on, allows you to ignore the bounds as you enter text.

AUTOMARGIN may be either ON or OFF. When it is OFF, EDIT2 discards text entered from the terminal that extends beyond RBOUND and warns you of that event. When AUTOMARGIN is ON, text beyond RBOUND is automatically carried over to the next line. When this happens, the next prompt for text is immediately followed by a display of the text that was carried over and the cursor or printhead is positioned one column beyond the end of this text waiting for subsequent input. If the bounds delimit short lines, your initial input may become two or even three lines before you are again prompted for more text.

When you use a Hewlett-Packard terminal in the 2640 family, the setting of AUTOMARGIN has special effects not seen on other terminals. When AUTOMARGIN is OFF and you are adding, replacing, or inserting lines, a white bar appears on the screen one column beyond RBOUND. The bar may be interpreted as a "wall". When text reaches the "wall" no additional text may be entered. Characters typed on or beyond the "wall" are discarded. When AUTOMARGIN is ON, no bar appears, and you may continue entering text without regard for RBOUND. AUTOMARGIN automatically places the text which exceeds the bound on a new line.

Bounds are also useful when making changes on text in which position is important, such as columnar data or certain program statements. The operation of the following commands is strictly bounded by the settings of LBOUND and RBOUND.

ADD	CHANGE	CENTER	FIND
REPLACE	MODIFY	COMPRESS	LIST
INSERT	JUSTIFY		

This means that text outside the bounds is not affected in any way by the operation. The commands FIND, LIST, and PRINT don't affect the text anyway, but it is important to note that a location which is out-of-bounds will not be found nor will text that is out-of-bounds be listed.

When AUTOMARGIN is ON, however, CHANGE and MODIFY will permit extending a line beyond RBOUND to make alterations; they then create a new line to contain the out-of-bounds text.

On the other hand, the following commands are not limited by line boundaries:

KEEP	DELETE	COPY	MERGE
TEXT	RENUMBER	MOVE	

The FILL command observes the bounds in a special way. It disregards the bounds to obtain text, but ensures that the resulting text is within the bounds.

LOCATING A POSITION IN YOUR DOCUMENT

In order to use EDIT2 effectively, you must be able to find any line in your document or within a selected portion of your document. You may also want to locate particular characters or words within your document or within portions of your document.

MOst EDIT2 commands have parameters that allow you to specify a position to be edited. Depending on the command, the parameter may limit you to a particular type of position or may allow you a broad selection of ways to locate text. For example, the ADD command (fully described in Section IV) allows you to specify only a single line number, whereas the FIND command (fully described in Section III) allows you to specify a line number, a column number within a line, the current line or current column, the first or last line of your document, the first or last column in the line, or to locate the line and column by means of the contents of the line. A further characteristic of FIND is that you may elect to search the entire document or only portions of the document, that is, a range of text delimited by line numbers, or certain columns within a range of text denoted by beginning and ending words.

In general, parameters are of two types: those that allow you to locate lines of text only, and those that allow you to locate characters, words, or column positions within lines. Whenever a parameter is modified by the word "character" it indicates the most general type of position specification is allowed, whereas the word "line" in a parameter limits the position specifications to lines of text. The elements used to build these two types of parameters are illustrated in Figure 2-1.

As shown in Figure 2-1, a position parameter alone indicates a single line of text; a position parameter followed by a colon (:) or a slash (/) and a second position, indicates a range of text. If more than one range is specified separated by commas, the range is called a rangelist. Examples of range and rangelist are shown after the components of the position parameter are defined.

The position parameter itself is either a line position indicating a particular line of text, a line position modified by a column to indicate a particular character position, or a string that indicates a character position within a line of text.

LINE POSITIONS

```
line rangelist = line range, line range, ...

                 {line position                 }
    line range = {line position:line position}
                 {line position/line position}
                 {ALL                           }

                 {line number}
                 {FIRST       }     [+ increment]
line position =  {LAST        }     [- increment]
                 {*           }
                 {string      }
```

NOTE: By convention, an empty string ("" or @@) is interpreted to mean a blank line.

CHARACTER POSITIONS

```
character rangelist = character range, character range, ...

                      {character position                          }
    character range = {character position:character position}
                      {character position/character position}
                      {ALL                                         }

                      {line position                       }
character position =  {line position (column position)}
                      {string                              }

                      {column number}
                      {FIRST        }
                      {LAST         }
  column position =   {*            }   [+ increment]
                      {LEFT         }   [- increment]
                      {RIGHT        }
                      {string       }

          string =    {"... "}
                      {@... @}
```

NOTE: An empty string is not permitted in a column position.

Figure 2-1. Parameters Used to Locate a Position

LINE POSITION

A line position is indicated in its simplest form by a line number. A line number is the value assigned to each line of your document. The values range from .01 to 9999.99.

A line position may also be specified by one of the following keywords or as a string:

*	current line
FIRST	first line in WORK file or within a range
LAST	last line in WORK file or within a range

Thus if you want to edit the current line, use the asterisk (*) to indicate its position. For example,

MODIFY * this command displays the current line followed by a prompt so that characters in the line may be modified.

To edit the first or last line in your text or within a specified range, use FIRST or LAST:

REPLACE FIRST this command displays the first line in the WORK file followed by a prompt so that the replacement line can be entered.

DELETE LAST this command displays the last line in the file and then deletes the line from the WORK file.

If you want to determine the position of the current line, you may use the command FIND *. This form of the FIND command displays the current line and also points to the current character position in the line.

NOTE: In the FIND and INSERT commands, the asterisk (*) means both current line and current character position. In all other commands, the asterisk is used as a line indicator and refers to the leftmost accessible column of the line.

A line position may also be specified by using a string (either quoted ("characters") or enclosed in commercial at signs (@word@)). When a string is used as a line position, it refers to the line containing the specified string. For example, to modify the ine containing the word "mispeled" enter:

```
>>MODIFY @mispeled@
```

A line may also be specified as a relative line number; that is, as a line so many lines before or after a specified line number.

An integer preceded by a plus (+) or minus (-) sign is appended to the line number keyword in this specification.

For example:

253+9	the 9th line after line 253
37-3	the 3rd line before line 37
FIRST+4	the 4th line after the first line in the WORK file
LAST-6	the 6th line before the last line in the WORK file
*-1	the line before the current line
"home"+3	the 3rd line after the line containing "home"
""	the next blank line

CHARACTER POSITION

A character within a line can be specified by a column indicator or by a string. A column indicator is used to specify a position within a line rather than a character in the line. The string is used when content but not position is known.

COLUMN

A column is almost always a qualifier to the line number as in the line position(column) construct. Except when column is specifically indicated as a parameter, the column is enclosed in parentheses and follows a line specification. Column cannot be specified in any parameter containing the word line, such as line position, line range, or line rangelist.

The simplest form of the column parameter is the column number. A column number is an integer in the range 1 through 94. Note, however, that this range is, in practice, limited by the boundaries on text established by LBOUND and RBOUND. LBOUND is the leftmost column that can be accessed and RBOUND is the rightmost. By default, LBOUND is column 1 and RBOUND is column 65.

A column can also be identified by one of the following keywords:

*	current column
FIRST	first non-blank column within LBOUND and RBOUND
LAST	last non-blank column within LBOUND and RBOUND

```
LEFT        leftmost accessible column (LBOUND)
RIGHT       rightmost accessible column (RBOUND)
```

For example, assuming LBOUND is at column 1 and RBOUND is at column 50:

```
>>FIND 5(FIRST)
   5        Yes, this is line 5 of the document.
            ^(5)

>>FIND 5(LEFT)
   5        Yes, this is line 5 of the document.
        ^(1)


>>FIND 5(LAST)
   5        Yes, this is line 5 of the document.
                                                ^(39)

>>FIND 5(RIGHT)
   5        Yes, this is line 5 of the document.
                                                    ^(50)
```

A column may also be specified as a relative column, relative to a specified column number or keyword. As with the relative line specification, the increment is specified as an integer preceded by a plus (+) or minus (-) sign. For example:

```
(12+5)      column 17

(55-12)     column 43
```

The relative column specification can also be used with the keywords FIRST, LAST, LEFT, and RIGHT as illustrated below:

```
>>FIND 5(FIRST+10)
   5     Yes, this is line 5 of the document.
                    ^(15)

>>FIND 5(LEFT+10)
   5     Yes, this is line 5 of the document.
                ^(11)

>>FIND 5(LAST-20)
   5     Yes, this is line 5 of the document.
                          ^(19)

>>FIND 5(RIGHT-20)
   5     Yes, this is line 5 of the document.
                                    ^(30)
```

A relative column specification can also be given by a string as illustrated below:

```
>>FIND 5("this")
   5      Yes, this is line 5 of the document.
                ^(5)

>>FIND 5("this"+4)
   5      Yes, this is line 5 of the document.
                    ^(10)
```

STRING

A string may be specified to indicate a line or character position. A string is a group of characters enclosed in quotes ("abcde") or in at signs (@word@). When a string is specified, the first line following the current position (or within a specified range) which contains the string, is located. The location of the first character in the string is the implied column.

When a string is enclosed within quotes, the string is located even if it is part of a word. When a string is enclosed in at signs, the string is located only if it is surrounded by spaces or other non-alphanumeric characters.

```
>>FIND  "the" IN ALL
   1    This is the first line of the document.
                ^(9)
```

The string need not be a separate word. For example:

```
>>FIND  "is" IN ALL
   1    This is the first line of the document.
              ^(3)
```

When used as a column indicator, the column containing the first character in the string is indicated. For example:

```
>>FIND 5("is")
   5    Yes, this is line 5 of the document.
                ^(8)
```

As with other column and line specifications, a relative line or column can be indicated by appending an increment to the string.

For example, assuming that "is" is found first in line 1:

```
>>FIND "is"+4(1)
   5    Yes, this is line 5 of the document
        ^(1)
```

The following illustrates the use of a relative column position based on the position of a string.

```
>>FIND 5("is"+10)
   5    Yes, this is line 5 of the document
                        ^(17)
```

When EDIT2 looks for a string enclosed by "at" signs, the string is found only if it is separated from surrounding text by spaces or other non-alphanumeric characters, or is the first or last word in a line. This allows you to search for a separate word not embedded in another word.

The following examples illustrate the use of word strings.

```
>>FIND @is@ IN ALL
   1    This is the first line of the document
             ^(6)

>>FIND @There@ IN 8
   8     There's nothing like a person.
         ^(1)

>>FIND @in@
   8     in appears within thing and also in nothing.
                                            ^(34)

>>FIND @I@ in 55
  55     PRINT H,I,J,K
                 ^(9)
```

You may enter strings which include special non-printing characters or even characters not available on your terminal keyboard by using numeric equivalents.

Each character recognized by EDIT2 has a numeric equivalent. You may use the numeric equivalent of a character rather than the character itself to specify a character, provided the numeric equivalent is preceded by an apostrophe. A character so denoted must not be enclosed in quotation marks or at signs.

The following examples illustrate acceptable usage of this convention.

```
'7"End of listing"          ('7 is equivalent to BELL)
"any string"'46             ('46 is a period)
'91"any string"'93          ("[any string]")
'65'66'67                   (equivalent to ABC)
'42"re"'45"enter"'42        (equivalent to "*re-enter*")
```

The number following the apostrophe may range from 0 to 255.

This convention is used for certain programming language statements and is generally not applicable to textual documents.

RANGE AND RANGELIST

The combination of line and column parameters in the form line(column) designates a position within the document that is always a single line. When you want to designate a range of lines extending from a beginning line through an ending line, the range is indicated by a beginning position and an ending position separated by a colon (:) or a slash (/). In its simplest form, a range consists of a single line:

```
10      range is line 10
```

When a range of several lines is desired, the boundary positions of the range are separated by a colon or a slash:

*:LAST	range from the current line through the last line
36(10):45(9)	range from column 10 in line 36 through column 9 in line 45; note that column numbers cannot be specified if the parameter is qualified by line in the command format.

The entire WORK file can be designated by ALL:

ALL	range including all lines in the WORK file

If position is indicated by string, the range extends from the first character in the first string through the last character in the second string. When only a line number is specified as the

end position in a range, the range extends through column RBOUND in the line. For example,

"this string"	range consisting of the string "this string" only
"BEGIN"/"END"	range from the first character in "BEGIN" through the last character in "END"
*/"STOP"	range from the first character in the current line through the last character in STOP

Note that string means the line containing the string when it is used to indicate position, not column, in the examples below.

```
                    range from first character in line 7 through
                    last character in line 9
>>DEL 7:9
   7     This is line 7.
   8     This is line 8.
   9     This is line 9.

                    special range ALL to indicate entire file
>>LIST ALL
   1     This is the first line of the document.
   2     This is line two.
   3     This is line three.
   4     This is the end.
```

Whenever rangelist is allowed as a parameter, a series of ranges can be specified separated by commas. For example:

10:20,23:46	two line ranges, from line 10 through 20 and from line 23 through 46
50:60,85,100:LAST	three line ranges, lines 50 through 60, line 85 and line 100 through the last line in the WORK file
"write"/"read","key"/"now"	two character ranges: from the first character of "write" thru the last character of "read", and from the first character of "key" through the last character of "now"

COMMAND CONVENTIONS

Each command can be abbreviated to those characters which make it unique. For CHANGE, DELETE, EXPLAIN, MODIFY, and REPLACE, one character defaults to these commands even though there are other commands beginning with the same letters. Characters required to identify a command are underscored or shown in boldface type in the command descriptions later in this manual. Additional characters, if entered, must be in correct sequence.

A space is required between a command name and the first parameter when that parameter is a keyword such as FIRST or LAST. Blanks should also be used to separate elements in commands that are not separated by commas. Otherwise, spaces are not required but may be used between the command and its parameters or between multiple parameters for the same command.

FILE NAMES

All HP 3000 Computer System files are referenced by name. File names are composed of from one to eight alphanumeric characters, but they may not begin with a number. No distinction is made between upper and lower case letters, i.e., the file name Abcd is equivalent to the file name ABCD.

File names may be optionally qualified by appending a period followed by the group and the account in which they reside. For example:

```
FILE1.PUB.EDIT
```

SPECIAL FILE NAMES

When you execute EDIT2 from an HP 2640 series terminal equipped with cartridge tape units, you may use the special file names #TCL and #TCR to reference the left (#TCL) and right (#TCR) units as files. However, cartridge tape units may not be specified as WORK or USE files. See the FILE ORIENTED COMMANDS section for more details on the use of these special file names.

For producing output to a terminal-connected printer, use the special file name #PRINTER in either the LIST or PRINT command.

FILE RANGES

A file range is a line range in which the position specifications are limited to line numbers, i.e., relative lines such as FIRST, LAST, and ALL are not permitted, nor are relative values such as plus (+) or minus (-) a number of lines. When no file range is supplied, the entire file is implied.

If the associated command permits the UNNUMBERED parameter and it is used, the line numbers given in a file range are interpreted with respect to the first line in the field. To illustrate:

```
>>ADD 100 BY 1 FROM XYZ(3/5), UNNUMBERED
```

The ADD command above adds the third through fifth lines (i.e., records--not necessarily lines having those numbers) from the file XYZ. These lines are assigned line numbers 100 through 102 in the WORK file.

WHAT TO DO WHEN YOU DON'T KNOW WHAT TO DO

Type //HELP. You may type //HELP in response to any EDIT2 prompt. In response, EDIT2 prints a message that defines the current prompt level, and tells you what responses may be made.

To illustrate:

```
>>//HELP
To see a brief description of an EDIT2 command, type the
command 'EXPLAIN' followed by the command name, e.g.:

        >>EXPLAIN COPY

To see a list of all EDIT2 commands, type EXPLAIN.


>>ADD
 679    :///HELP
Type the text of the line you want to ADD.
Press CONTROL-Y to terminate this operation.


>>ADD#
#///HELP
Type the number of the line you want to ADD.
Press CONTROL-Y to terminate this operation.
```

The information provided by EDIT2 in response to a request for //HELP is tailored to the specific situation in which the request is entered.

WHAT TO DO WHEN YOU NEED MORE INFORMATION

Type the command EXPLAIN or HELP; they are synonymous. These commands provide descriptions of other EDIT2 commands. Thus, when you find that you are not sure what command to use in a particular situation, or you do not remember the exact format of the command you wish to use, simply enter:

>>EXPLAIN	>>HELP
or	or
>>EXPLAIN command name	>>HELP command name

When these commands are entered without a specific command name attached, a list of all EDIT2 commands is displayed. When a command name is specified, a brief description of that command is given, followed by the prompt CONTINUE?. If you require no further information regarding the command, type N or NO or CONTROL-Y. If additional information is required, press RETURN, and EDIT2 will continue to display information about the command with intermittent requests to CONTINUE? until you respond NO or until the available explanatory text is exhausted.

Commands that may be specified by name with EXPLAIN are:

ADD	DELETE	GET	MERGE	SET
CENTER	DISPLAY	HELP	MODIFY	SHOW
CHANGE	EXPLAIN	INSERT	MOVE	TEXT
COMPRESS	EXIT	JUSTIFY	PRINT	USE
COPY	FILL	KEEP	RENUMBER	
DEFINE	FIND	LIST	REPLACE	
//	//ASK	//EXIT	//HELP	//MODIFY

The following may also be specified:

POSTION RANGE

THE EDITING ENVIRONMENT

ENVIRONMENTAL CONTROL

After logging onto the system and running EDIT2 you are ready to develop or edit your document. You are able to alter some of the characteristics exhibited by EDIT2 and to set certain global values that together constitute the editing environment. The commands that give you access to these parameters are SET and SHOW.

SET

The following parameters may be entered in any order. Commas must separate multiple settings given in one SET command.

```
SET  AUTOMARGIN = {ON }
                  {OFF}

     BOTTOMSPACE = lines before footing, lines after footing

     CONTROL = {ON }
               {OFF}

     DISPLAY = {ON }
               {OFF}

     ECHO = {ON }
            {OFF}


     FACING = {OFF    }
              {offset}

     FILLER = {OFF       }
              {character}

               {OFF    }
               {CENTER}
     FOOTING = {LEFT   } [,footing string]
               {RIGHT  }
               {FACING}

               {OFF    }
               {CENTER}
     HEADING = {LEFT   } [,heading string]
               {RIGHT  }
               {FACING}

     HYPHENATION = {ON }
                   {OFF}

     INCREMENT = increment

     LABELS = {ON }
              {OFF}

     LBOUND = column number
```

```
NONPRINT = {OFF          }
           {"character"}

PAGESIZE = lines per page

PAUSE = {OFF             }
        {lines per pause}

RBOUND = column number

        {OFF           }
TABS = {POINT         }
        {tab positions}

TOPSPACE = lines before heading, lines after heading

WARNTIME = number of seconds to wait

WIDTH = terminal width

//COMMANDS = {ON }
             {OFF}

STANDARD
```

STANDARD

When STANDARD is specified, all settings are restored to default values. (See SHOW ALL example for standard settings.)

AUTOMARGIN

When AUTOMARGIN is ON, lines added, inserted, modified, or changed are automatically folded between words if they are too long to fit between the designated margins (LBOUND and RBOUND). New lines are added automatically to contain the folded text.

CONTROL

Control characters are NOT permitted during input of commands or text when CONTROL is ON.

DISPLAY

The DISPLAY parameter generally lets you enable or inhibit the display of line numbers when listing or editing. Occasionally, DISPLAY also suppresses the display of text as well as line

numbers depending on the operation involved (see the description of each command for specifics).

ECHO

When ECHO is ON, the expanded command string (which results when defined names are replaced) is displayed prior to its execution.

FILLER

The FILLER parameter specifies a character to be used in place of spaces in your text. Use of a filler character ensures that words are not separated when lines are being filled and the number of spaces between words is not increased when the text is justified.

The FILLER character may be any printing character. When text containing filler characters is printed (using the PRINT command), filler characters are replaced with spaces. The default for FILLER is OFF.

NONPRINT

The NONPRINT parameter lets you specify a character to be printed whenever a nonprinting character is encountered in your text. This character is used for listings and for the display of a line for modification using the MODIFY command. (The MODIFY command will use @ even if NONPRINT is OFF.)

INCREMENT

INCREMENT defines the increment to be used by commands which add successive lines of text. Smaller increments will be used automatically when the situation does not permit the use of the specified increment. INCREMENT can also be temporarily changed by using the BY option in some commands.

PAUSE

The value of PAUSE determines the number of lines displayed at the terminal before a pause. At the pause, EDIT2 issues the CONTINUE? prompt; respond YES or RETURN to continue; NO, //, or CONTROL-Y to stop. When paginated documents are being previewed on the terminal display, PAUSE temporarily uses the value of PAGESIZE to permit loading paper on discrete forms terminals and the CONTINUE? prompt is omitted.

LBOUND and RBOUND

The LBOUND and RBOUND parameters permit you to set boundaries within which text and operations on text are to be confined. Each command description includes details on the effect of these bounds on the operation of the command. (Also see the discussion entitled WORKING WITHIN THE BOUNDS earlier in this section.)

LABELS

The LABELS parameter permits you to enable or disable the recognition of labels. The automatic recognition of labels allow operations that wrap words (e.g., ADD) to detect the logical beginning of the line so that the wrapped text can be indented automatically. This occurs most often when numbered lists, outline format, or bulleted lines are entered.

The first element or word in a line is recognized as a label if it is:

a. followed by 2 spaces and its last character is not a sentence ending punctuation mark (. ? !), or

b. a bullet (lower case "o", asterisk, or hyphen), or

c. composed of an optional left parenthesis followed by:

   - a single letter (upper or lower case), or

   - a number (any number of digits), or

   - a Roman numeral (upper or lower case);

   all of which may contain one or more embedded, and MUST contain at least one trailing: period, right parenthesis, slash, dash, or colon.

When the conditions for a label described in (b) or (c) are met, only one space is required between the label and the first word of the line.

A label must be a single word or group of characters with no intervening spaces.

The following examples illustrate some different kinds of labels:

```
1. A simple label
IX. A Roman label
iii. Also a Roman label
(B) A letter label with parentheses
a) Also a label
```

```
o A bullet
* Also a bullet
1.1.2  A label because of the two spaces following it
TEXTUAL  A textual label because of the two spaces.
```

HYPHENATION

The HYPHENATION parameter enables or disables hyphenation when using the FILL command.

TABS

TAB = tab positions is the format used to set tabs by column number; separate the column numbers with commas. Up to 10 tab positions may be set.

TAB = POINT is used to set tabs by spacing across the line under a column guide. Type any character in the columns to which you wish to set tabs.

TABS are set and cleared automatically on HP 2640 series terminals. You may use the TAB key on such terminals and any others whose TAB key transmits a CONTROL-I character to the computer. Otherwise, you will have to use the CONTROL key to send the CONTROL-I to indicate tab.

When set, tabs are in effect for all textual input. You may use tabs with the following commands: ADD, REPLACE, INSERT, MODIFY, and MERGE#. If you try to use tabs and you get the message: NO CONTROL CHARACTERS ALLOWED, you probably have not set tabs in the current WORK file. This happens most often when you go from a WORK file that has tabs set to one that dosen't.

//COMMANDS

Set //COMMANDS = OFF when it is necessary to enter a line of text which begins with the characters //.

WARNTIME

The WARNTIME option permits you to specify the delay for time-out warnings. When set to zero, no warnings are given. When set to any other value (up to 255 seconds), you have at least that long to think about the advisability of allowing the impending action to occur. To cancel the action you must press RETURN or type something beginning with "N" (NO, NEGATIVE, etc.) before the time limit. To explicitly approve and get on with it, you may type any character(s) other than "N" followed by a

RETURN. The default value for WARNTIME is 3 seconds which is just long enough to catch a blooper but not too long to require explicit response for every warning.

WIDTH

WIDTH specifies the actual line length accomodated by your terminal. The value of WIDTH is used only during terminal listing operations. Lines are split at a space if possible when they exceed the width of your terminal display.

The default value is 79 which also happens to be the ideal value for an HP 2640 series terminal. A value greater than 79 on a 2640 terminal will give an extra linefeed with each line that goes into column 80 on the screen. A value less than 50 will not be accepted.

HEADING

HEADING permits you to define and position the heading on paginated documents. When HEADING is OFF, no heading is printed, however, a blank line exists in the formatted page where the heading would normally be. This allows you to print a document with and without headings using the same settings for PAGESIZE and TOPSPACE for both listings.

You may include the page number, date, and/or time in your heading by using the names #PAGE, #DATE, and #TIME; they will be replaced by appropriate values when the document is printed.

FOOTING

The FOOTING parameter controls the value and placement of the footing on paginated documents. No page number is printed if #PAGE is not specified in the FOOTING string. When FOOTING is OFF, the page number is not printed (a blank line is issued instead). However, page numbering can be activated for an instance of a PRINT command by using the PAGENUMBER or MEMO options. Page numbering can also be altered while the document is being printed by embedding the .PAGENUMBER subcommand in the text.

You may also include the date and/or time in the footing by using the #DATE and #TIME indicators.

PAGESIZE

The PAGESIZE parameter determines the number of lines per page including those taken by TOPSPACE and BOTTOMSPACE. See Figure 2-2 for page layout details.

TOPSPACE

The TOPSPACE parameter sets the number of blank lines preceding and following the line on which the heading is printed; each may be set independently.

BOTTOMSPACE

The BOTTOMSPACE parameter sets the number of blank lines preceding and following the line on which the footing is printed; each may be set independently.

FACING

The value of FACING is used to offset the text on entire pages for paginated documents. Text on odd numbered pages is offset to the right by the amount specified; the LEFT parameter of the PRINT command determines the offset on even pages. This is useful for printing a document that will be reproduced on both sides of the paper and then bound or punched.

NOTE

When you use PRINT to print a document, there is a .NEED=4 subcommand (see embedded subcommands in the description of the PRINT command) in effect before each paragraph is printed. This means that a new paragraph will not be printed at the bottom of any page unless there are at least four lines remaining.

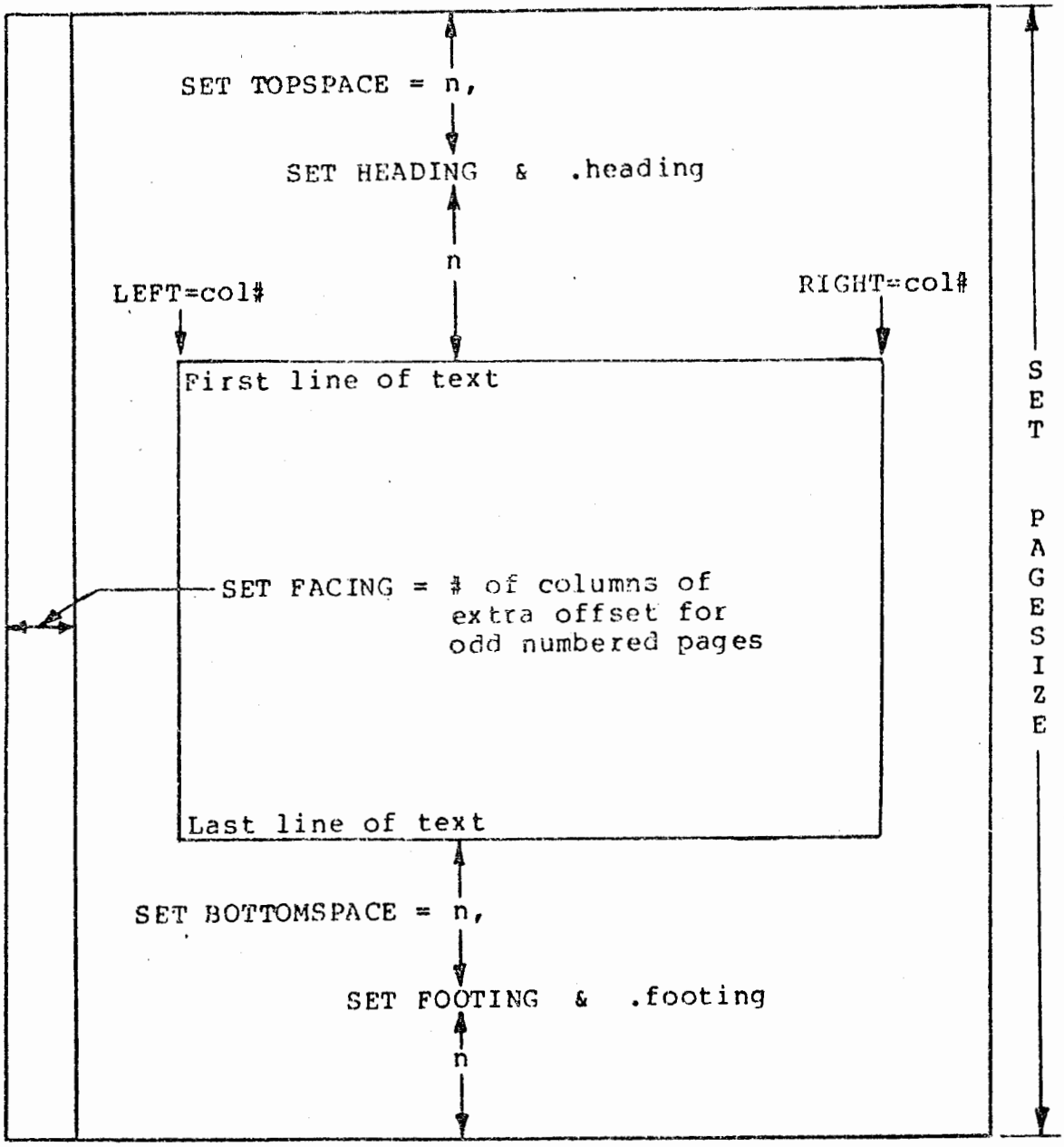

Figure 2-2. Page Dimension Parameters

SHOW

The SHOW command displays the current status of any of the SET parameters. To display values of more than one parameter, supply a list of parameter names separated by commas. Use the keyword ALL to display all parameter values.

For example, the following parameters and their values (shown here with standard values) are displayed when you enter:

```
>>SHOW ALL

AUTOMARGIN...ON      CONTROL......ON      DISPLAY.....ON
ECHO.........OFF     //COMMANDS...ON      LINES....... 0
LBOUND...... 1       RBOUND...... 65      WIDTH....... 79
WARNTIME.... 3       INCREMENT... 1       PAUSE....... 22
HYPHENATION..ON      FILLER.......OFF     NONPRINT.....OFF
LABELS.......ON      TABS.........OFF

PAGESIZE.... 60      TOPSPACE.....2,2     BOTTOMSPACE..2,0
FACING.......OFF
HEADING......CENTER,'EDIT2/3000 - #DATE, #TIME'
FOOTING......CENTER,'#PAGE'

DEFINITIONS:
     fix <= fill */""
     go <= list */last

        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345
```

Additional values are available by specifying:

LINES	- Displays the number of lines currently in your WORK file.
CHARACTERS	- Displays the number of characters in your WORK file.
FILES	- Displays the names of currently accessed files.
DEFINITIONS	- Displays your definitions.
GUIDE	- Displays the column guide.

GUIDE

When GUIDE is specified, a column guide is displayed at your terminal. The guide serves as a reference to the column numbers between LBOUND and RBOUND and shows tab postions if any are set.

FIND

The FIND command is used to locate a specific location in the WORK file and to set the pointer (current editing point) to that position.

FORMAT

```
FIND[D]  [character position                                        ]
     [Q]  [(ALL) string  (IN character rangelist)                   ]
          [ OUT (IN line rangelist)                                 ]
```

DESCRIPTION

The FIND command performs three distinct functions. Its primary function is to locate a specified character position in the WORK file and to set the pointer to that position. For example:

```
>>FIND 15("X")
  15   Here is the X.
                   ^(13)
```

As a secondary function, FIND locates all occurrences of a given character string within an optional character rangelist. Its third function is to locate all lines which have text beyond RBOUND.

CONTROL-Y

Pressing CONTROL-Y terminates a FIND command. If the position has not been located, the pointer is not moved.

LIMITATIONS

FIND operates within the limits established by LBOUND and RBOUND. NO character position outside of these bounds will be found unless OUT has been specified.

FINDD and FINDQ

When DISPLAY is ON or FINDD is specified, FIND lists the line containing the specified position and indicates the location of the pointer with an up-arrow (^) followed by the column number

enclosed in parentheses. On HP 2640 series terminals the position is highlighted by inverse video.

When DISPLAY is OFF or FINDQ is specified, neither the line nor the pointer is displayed.

FIND character position

The following are examples of character position specifications:

```
>>FIND FIRST        - sets the pointer to the first accessible
                      column in the first line.

>>FIND *            - displays the current position of the
                      pointer.

>>FIND 100+1        - sets the pointer to the first accessible
                      column in the line following 100.

>>FIND*+5(FIRST)    - sets the pointer to the first non-blank
                      character in the fifth line following the
                      line currently having the pointer.

>>FIND "line" ("column") - sets the pointer to the word
                           "column" in the line containing
                           the word "line"
```

When character position is a string (i.e., a collection of characters enclosed by quotes or @), the FIND command searches forward in the WORK file beginning one character beyond the present position of the pointer until it locates the string or the end of the last line in the file is reached. If the string is located, the pointer is moved to the column containing the first character of the string. The line containing the string and the position of the pointer are then displayed. If the string is not located, the message: 'string' NOT FOUND is issued.

FIND string IN character rangelist

The search for a string may be limited to a certain range of lines called a character rangelist. Multiple ranges may be specified, provided they are separated by commas. The pointer is set to the first occurrence of string in the rangelist; subsequent ranges are not searched. For example:

```
>>FIND @BC@ IN FIRST:200,503:LAST
```

In the above command, if the word BC is located within the first range of the rangelist (FIRST:200), the FIND command terminates without searching the second range. If the string cannot be

found in one of the specified ranges, the message: 'string' NOT FOUND is issued.

FIND ALL string

This form of the FIND command enables you to locate all lines containing one or more occurrences of string. The search begins one character beyond the current location of the pointer and continues through the last line in the WORK file. Each line containing an occurrence of string is listed and the total number of occurrences is reported after the last is found. Only the number of occurrences is reported when DISPLAY is OFF.

Following the search procedure, the pointer is repositioned to the first occurrence of the string, unless the command is interrupted by CONTROL-Y before the first occurrence is encountered.

FIND ALL string IN character rangelist

When IN character rangelist is specified, FIND searches for string within each range regardless of the current position of the pointer. The pointer is then set to the first occurrence. This is true even if a subsequent range includes lines which precede those in the first range. The number of occurrences is reported at the end of each range.

FIND OUT [IN rangelist]

This form of the FIND command enables you to quickly find those lines which have text beyond RBOUND. The search begins at the current location of the pointer unless otherwise directed by a rangelist and goes to the end of the WORK file. Only those lines which have text beyond RBOUND are displayed and only the text that is beyond RBOUND is displayed.

General Operating Commands give you flexibility in using EDIT2 to accomplish your text editing efficiently. There are five commands in this category.

- o // - used instead of CONTROL-Y from USE files or at the terminal.
- o //HELP - used to obtain help at any prompt.
- o EXPLAIN - used to obtain detailed information about a particular command.
- o EXIT - used to stop EDIT2.
- o DEFINE - used to define custom commands.
- o LIST - used to list the contents of your WORK file at the terminal.

This section describes the function of each of these commands in detail, and provides information on how to use each command.

//

The // command is primarily used from a USE file to signal EDIT2 to terminate execution of one command (such as ADD) and go on to the next command in the file.

// may also be entered in response to either the text prompt (:) or the number sign prompt (#), and has the same effect as pressing CONTROL-Y. It must always be the first and only entry in response to the prompt, and may never be appended to other textual input as it is then just text.

//HELP

You may type //HELP in response to any EDIT2 prompt. EDIT2 will then tell you what responses may be made at the current prompt level.

GENERAL OPERATING COMMANDS

EXPLAIN and HELP

The EXPLAIN and HELP commands are used to obtain summary descriptions of EDIT2 commands.

FORMAT

EXPLAIN [command name] or HELP [command name]

DESCRIPTION

When either command is entered without a command name, a list of all commands is displayed. When a command name is specified, a brief description of that command is given followed by the prompt CONTINUE?. If you require no further information concerning the command, type N or NO or press CONTROL-Y. If additional information is desired, press RETURN. Explanatory information is displayed with intermittent requests to continue until the available text is exhausted.

EXIT and //EXIT

There is only one correct way to stop interaction and return to the MPE command interpreter. Simply type:

EXIT (>> prompt only) or //EXIT (any prompt)

//EXIT may be entered in response to the command prompt (>>), the text prompt (:), the number sign prompt (#), or the WORK FILE? prompt.

Any other method of terminating interaction, such as pressing CONTROL-Y, will fail to produce the desired result. Disconnecting the terminal or aborting after pressing BREAK will result in a disorganized WORK file. If this happens you will be notified the next time you specify that file as your WORK file, and an attempt will be made to recover its contents.

DEFINE

The DEFINE command is used to assign a name to frequently used input such as a command or command argument. The format used for this command is:

DEFINE name = [unquoted string]

DESCRIPTION

DEFINE enables you to assign a name to frequently used commands or portions of commands. Once a name is assigned, you may identify the command string by name when entering a command.

For example, if you wanted to use the command string LIST FIRST;LIST LAST repeatedly, you could assign it the name LFL and use the name rather than the entire command string each time. This is done as follows:

DEFINE LFL = LIST FIRST;LIST LAST

Similarly, if you have a portion of a command that you employ often, assign it a name and simply substitute the name for the appropriate part of the command whenever you wish to use it.

All defined names and their meanings are stored in the DEFINE dictionary. To obtain a list of defined names simply enter:

SHOW DEFINITIONS

In response, EDIT2 prints a list of defined names and their associated command strings.

LIMITATIONS

The name assigned to a definition must be a contiguous set of characters (no spaces). Any characters may be used, even quotation marks as they are not used to delimit names. A name may duplicate a command name or parameter such as MOVE or BY; however, when used as defined commands may not be used for their original function. This means that you may redefine a command name provided you don't need the command in its original form.

When constructing definitions, it is advisable to use the shortest abbreviation possible to conserve space in the DEFINE dictionary.

EDIT2 searches for defined names from left to right. When a defined name is found, EDIT2 replaces it with the meaning assigned to it in the DEFINE dictionary. EDIT2 then resumes its search proceeding from the first character following the inserted definition. Thus defined names within other definitions are not found by EDIT2.

USING DEFINED NAMES

A name can be entered in response to a >> prompt, or as part of command. Names must be separated from other command parameters by one or more spaces or non-alphabetic characters.

REMOVING A DEFINED NAME

To remove or "undefine" a name, re-enter the DEFINE command and omit the string portion. This deletes both the name and its meaning from the DEFINE dictionary. To illustrate:

```
>>define namea =
'namea' IS NOW REMOVED.
```

LIST

The LIST command is used to display all or a portion of the WORK file at the terminal. Although it can produce paginated listings OFFLINE, the PRINT command is preferred for document production.

FORMAT

```
LIST[D]      [line rangelist]
     [Q]     [,TO filename] or [,OFFLINE]
             [,COPIES = number of copies]
             [,LEFT = column number][,RIGHT = column number]
             [,SPACING = number of lines]
             [,PAGINATE [ = CONTINUE    ]]
                        [ = page number ]]
```

DESCRIPTION

All of the parameters available with the LIST command are optional. The parameters may be specified in any order but must be separated by commas. Parameters are set to the last specified value or condition if repeated. ALL is assumed when no line rangelist is supplied.

PROMPT and POINTER

The LIST command issues no prompt of its own. Necessary command qualifiers are provided as parameters.

The pointer is positioned to column LBOUND in the first line of the last range listed.

CONTROL-Y

CONTROL-Y may be used at any time to terminate a listing.

LISTD and LISTQ

If DISPLAY is ON or LISTD is specified, the line numbers of each line in the line rangelist will be displayed to the left of the text. If DISPLAY is OFF or LISTQ is specified, the text is listed without line numbers.

LIST (line rangelist,) TO filename (or OFFLINE)

The TO filename parameter enables you to specify a file to which the lines in the line rangelist are to be listed. When specified, the text is listed to a file rather than to the terminal. To send the listing to a device (such as a line printer) you must first name the file in a file equation unless you use OFFLINE. (OFFLINE uses the filename EDITLIST which may be over-ridden by a :FILE command after EDIT2 is running.)

NOTE: The HP 2640 series cartridge tape units cannot be specified as filenames for LIST or PRINT commands; instead the KEEP command must be used for this purpose.

COPIES

The COPIES parameter is used to specify the number of copies desired. However, this can be more efficiently accomplished with the copies parameter of the MPE file equation.

LEFT = column number, RIGHT = column number

The LEFT and RIGHT parameters specify print margins to be used for an incidence of the LIST command. The default values are LEFT = 1 and RIGHT is set automatically to accommodate the text between LBOUND and RBOUND if not set explicitly. The values of LBOUND and RBOUND determine the extent of the text to be listed.

Printed lines, including line numbers if present, begin in the column position specified by LEFT. LEFT must be less than or equal to RIGHT, or the message: MARGINS PRECLUDE TEXT is issued and the LIST command is not executed.

Unless TO filename or OFFLINE is specified, a value for RIGHT that exceeds WIDTH precipitates the message RIGHT MARGIN EXCEEDS WIDTH and the LIST command is not executed. The value of RIGHT may exceed WIDTH only if TO filename or OFFLINE is specified.

SPACING = number of lines

The SPACING parameter causes listed lines to be spaced according to the value specified as number of lines. The default value is 1 (for normal single spacing). Values of 2, 3, or 4 may be specified.

PAGINATE

The PAGINATE parameter is used to format text into optionally titled and numbered pages. The number of lines per page is determined by the value of the SET PAGESIZE parameter.

Page numbering begins at one unless an alternate starting page is specified as page number in the PAGINATE parameter.

The PAGINATE = CONTINUE option permits you to continue consecutive page numbering when listings are concatenated by executing consecutive LIST commands. When CONTINUE is specified, the first page is given a number one greater than the last page of the previous listing.

Paginated listings produced by the LIST command may differ from those produced by the PRINT command. The LIST command should be used to produce OFFLINE listings of programs and other non-English text as it does not recognize paragraphs or try to avoid widowed lines as PRINT does. The LIST command also dosen't execute embedded subcommands that direct EDIT2 to go to a newpage, etc., so it is useful to obtain a document listing which shows such embedded commands as text.

INPUT COMMANDS

Input commands enable you to enter new textual material into your WORK file. There are three primary input commands:

- o ADD - used to enter text into the WORK file from the terminal or from a specified file.
- o REPLACE - used to replace lines in the WORK file with new material.
- o INSERT - used to insert characters into existing lines of text.

This section describes the function of each of these commands in detail, and provides information on how to use each command.

ADD

The ADD command is used to enter text into the WORK file from a terminal or from a specified file.

FORMAT

```
ADD[D]  [    *      ]
   [Q]  [line number] [BY increment]
            [FROM filename [(file range)]  [,UNNUMBERED]]
                                           [,REAR      ]]

ADD#
```

DESCRIPTION

The ADD command adds text to the WORK file from the terminal or from any ASCII file. The command adds text between existing lines in the WORK file if a specific line number is declared, or to the end of the WORK file if line number is not specified. If the specified line already exists, EDIT2 displays that line and prompts you with the next available line number.

ADD begins adding lines at the specified line number and increments by the value of SET INCREMENT (default increment is one) unless the optional BY increment parameter is specified or it is not possible to use the requested increment. For example, if you want line numbers to be temporarily incremented by 10, then specify BY 10 in your ADD command. ADD will follow your lead for fractional line numbers instead of using INCREMENT (e.g., ADD 1.3 implies an increment of .1).

If the line number parameter is absent from the ADD command and the WORK file is empty, text begins at line one. Left and right bounds are determined by the SET LBOUND and SET RBOUND commands.

PROMPT AND POINTER

When EDIT2 is ready to accept a new line of text, it prompts with a colon (:).

```
>>ADD 66
  66  :New text
  67  :Additional text
  68  ://
>>
```

When AUTOMARGIN is ON, the colon may be followed by text from the previous line if the length of that line exceeded RBOUND; you begin typing after that text is displayed.

```
>>ADD 100
 100   :First line of text runs over into second line.
 101   :second line.  You begin typing ...   <--RBOUND
 102   ://
```

When AUTOMARGIN is OFF, characters beyond RBOUND are discarded and must be re-entered on a new line.

Following the addition of text, the pointer is positioned at column LBOUND of the last line added.

ADDD and ADDQ

When the DISPLAY is ON or ADDD is specified, EDIT2 prompts for each new line with a line number followed by a colon. Only the colon is issued if DISPLAY is OFF or ADDQ is used. When adding from a file, the lines are displayed unless ADDQ is used.

CONTROL-Y

Pressing CONTROL-Y at any time after an ADD command has been entered terminates the operation.

All text typed before CONTROL-Y is entered into the WORK file before the command is terminated. If no text was entered before CONTROL-Y, the line number remains unused (i.e., does not exist).

ADD#

ADD# permits you to add lines, one at a time, in any order that appeals to you. EDIT2 issues a number sign prompt rather than the usual colon. You enter a line number and press RETURN. You are then prompted for the text of that line.

```
>>ADD#
#53
  53    :Text of line 53.
#//
>>
```

Pressing CONTROL-Y, typing //, or RETURN terminates the # prompting.

```
ADD [line number] FROM filename [(file range)] [,UNNUMBERED]
                                               [,REAR       ]
```

The FROM filename parameter causes lines of text to be added to the WORK file from a file. If a starting line number is not specified, the lines will be added to the end of the WORK file.

When an entire file is to be added, simply supply the file name. If only a select group of lines within the file is to be added, specify them as the file range (first and last line numbers or first and last relative lines if the file is unnumbered). The line numbers in a numbered file are used for reference only; you may direct ADD to place them wherever you want by specifying a starting line number.

//MODIFY

The //MODIFY command is used to modify the text of the line just entered using the ADD command.

FORMAT

//MODIFY

DESCRIPTION

The //MODIFY command permits you to correct a typing error made while you were adding the previous line before you have begun to enter text on the current line. You may only issue the //MODIFY command in response to the colon prompt issued by ADD. For example,

```
>>ADD
   1    :This lnie has a transposition.
   2    ://MODIFY
   1     This lnie has a transposition.
        :      tt
   1     This line has a transposition.
        :<return>
   2    :We are now back to adding lines.
   3    ://
>>
```

//MODIFY is not permitted when AUTOMARGIN has wrapped one or more words onto the next line. See the MODIFY command for details of its operation.

REPLACE

The REPLACE command is used to replace lines of text in the WORK file.

FORMAT

```
REPLACE[D] [line rangelist]
       [Q] [FROM filename [(file range)] [,UNNUMBERED]]
                                         [,REAR      ]]

REPLACE*
```

DESCRIPTION

The REPLACE command replaces lines of text that were previously entered into the WORK file. You indicate which line is to be replaced, then enter the new text for that line. Replacement text may be entered from your terminal or directed to the WORK file from another file.

You may specify a single line or a range of lines to be replaced. Replacement occurs only within the limits of LBOUND and RBOUND.

PROMPT AND POINTER

When EDIT2 is ready to receive a line of replacement text, it prompts with a colon (:). If the DISPLAY option is ON, the current content of the line is displayed before the prompt. In response you enter the replacement text and press RETURN.

The new text entirely replaces the text between LBOUND and RBOUND of the line. Pressing RETURN without entering replacement text deletes the content of the line between LBOUND and RBOUND (i.e., replaces it with spaces when there is text beyond RBOUND), but does not delete the line number or any text outside the boundaries. If a line containing out-of-bounds text is replaced, the out-of-bounds text will still be present after the in-bounds portion is replaced.

Following the replacement operation, the pointer is positioned at column LBOUND in the following line if one exists; otherwise it is positioned at column RBOUND of the replaced line.

CONTROL-Y

Pressing CONTROL-Y in response to the colon prompt terminates the replacement of the current line and any following lines in a line rangelist. The line is not replaced, but preceding lines which have already been replaced remain in their changed form. When CONTROL-Y is pressed after text has been entered, that line is the last to be replaced in the line rangelist.

LIMITATIONS

REPLACE operates solely wihtin the boundaries established by LBOUND and RBOUND. Text to the left of LBOUND and to the right of RBOUND is not replaced by this command. If the length of a replacement string exceeds RBOUND, the message: RBOUND EXCEEDED, REPLACEMENT NOT MADE IN LINE line number is issued.

REPLACED and REPLACEQ

When REPLACED is specified or DISPLAY is ON, EDIT2 prompts by displaying the line number and content of the line to be replaced. When REPLACEQ is specified or DISPLAY is OFF, the text line is displayed without a line number.

REPLACE#

Appending a number sign to the command indicates that you want to supply line numbers of lines to be replaced. The # permits you to replace lines, one at a time, in any order. EDIT2 prompts with a # for the number and then displays the text of the line. The original content of the line is displayed first followed by a prompt for the new text.

For example:

```
>>REPLACE#
#75
  75   This is the original content of line 75.
  75  :This text replaces that shown above.
#//
>>
```

REPLACE line rangelist FROM filename [(file range)] [,UNNUMBERED]
[,REAR]

Text for replacement lines may originate from a file. When the entire file is to be used as a replacement, simply specify the file name, and the lines in the file will replace the specified

lines in the WORK file. The line numbers in the file (if any) are used for reference only.

If only a selected group of lines within the file is to be used for replacement, specify them as the file range.

The file supplying the replacement text may contain more lines than the line rangelist specifies, but must contain at least that many lines or the message: *** WARNING: FULL RANGE NOT REPLACED is issued.

If DISPLAY is OFF, the message: line number WAS THE LAST LINE REPLACED is displayed; otherwise, the original text of each replaced line is displayed.

INPUT COMMANDS

INSERT

The INSERT command is used to insert text within lines or entire lines of text into the WORK file at a specific position within a line.

FORMAT

```
INSERT[D]  [character position] [BY increment]
      [Q]
```

DESCRIPTION

The INSERT command inserts text into an existing line of your document. You specify the line and column position where the insertion is to be made, then enter the text to be inserted. The INSERT command always inserts text just before the indicated postion.

You may insert as little text as one character or you may insert many lines of text at a specified position in your document. Spaces and blank lines may be inserted wherever needed.

When only a line number is specified as the character position, EDIT2 displays the line and positions the colon prompt just before column LBOUND. You may then enter the text to be inserted. The text entered is inserted at the beginning of the line; the existing text is moved to the right of the inserted text. For example:

```
>>INSERT 23
   23    old text
        :new text <CONTROL-Y>
   23    new text old text
```

When a column position is also specified in the character position , EDIT2 prints the colon prompt one column before the specified column and positions the cursor under the specified column. Text is inserted in the line at the position specified. Both leading and trailing spaces accompanying inserted text are retained. For example:

```
>>INSERT 23("old")
   23    new text old text
                :and <CONTROL-Y>
   23    new text and old text
```

When DISPLAY is OFF, EDIT2 displays the text but does not display the line number nor does it prompt for successive lines with line numbers; only the colon appears as a prompt for new lines.

Text is always inserted before the specified column. If the column specified is LBOUND, all inserted text precedes the remainder of the line. If the insert column is some other position, inserted text precedes the former contents of that position.

When ther is not text beyond RBOUND, EDIT2 continues to prompt for text, line by line, until CONTROL-Y is pressed. Text beyond RBOUND precludes additional lines and if the insert lengthens the line so that it no longer fits between LBOUND and RBOUND, the message: RBOUND EXCEEDED, INSERTION NOT MADE is issued.

PROMPT AND POINTER

When EDIT2 is ready to receive an insertion, it prompts with a colon. If DISPLAY is ON, the current content of the line is displayed with its line number before the prompt is issued. In response you enter the text to be inserted and press CONTROL-Y. If DISPLAY is OFF the line number is not displayed.

After an insertion is made, the pointer is positioned one column beyond the end of the inserted text. If the insertion ends in column RBOUND, the pointer is positioned to column LBOUND of the following line. When this occurs in the last line of your work file, the pointer remains at column RBOUND in the current line.

CONTROL-Y

An insertion operation must be terminated by CONTROL-Y or by //. The following conventions let you control the placement of inserted text and the remainder of the line in which the insertion is made.

- Pressing CONTROL-Y informs EDIT2 that this is the last line to be inserted, and that any remaining text is be appended to this line. Existing text to the right of the insert is not affected by the insert; it is placed at the end of the inserted text on the same line if there is room, otherwise it is placed on a new line below the inserted text (assuming there is no text beyond RBOUND and AUTOMARGIN is ON). To illustrate:

```
>>INSERT 12
  12  short line.
     :this is a <CONTROL-Y>
  12  this is a short line.
```

- Pressing RETURN causes EDIT2 to place text originally to the right of the insert column on a new line. Text to the left of the insert column is unchanged. To illustrate:

```
>>INSERT 15
  15     short line.
        :this is a <return>
  15     this is a
  16    ://
  16     short line.
>>
```

- To insert a blank line between text to the left and the right of the column specified, press RETURN to the first and second colon prompts, then press CONTROL-Y in response to the third colon prompt. To illustrate:

```
>>INSERT 16("A")
  16       last line of a paragraph.  A new paragraph
                                     :<return>
  16       last line of a paragraph.
  16.1    :<return>
  16.2    :<CONTROL-Y>
  16.2     A new paragraph
>>LIST 16:16.2
  16       last line of a paragraph.
  16.1
  16.2     A new paragraph
>>
```

LIMITATIONS

An insertion must always be made in an existing line. Use the ADD command to add lines between or following existing lines.

INSERT respects the setting of LBOUND and RBOUND at all times. Only text within these boundaries is available for specification as a character position. Text beyond these boundaries is not available for insertion; nor can a multi-line insert begin in a line which has text to the right of RBOUND.

When AUTOMARGIN is ON and text is inserted in a line that contains no text to the right of RBOUND, new lines are automatically created to contain text which exceeds the limits of one line. Thus, if insertion text consists of more characters than are allowed on one line, EDIT2 creates new lines as necessary to accommodate the additional text.

New line numbers are determined by the SET INCREMENT value if the BY increment parameter is not used. The insert operation is terminated and the error message: INSERT BLOCKED BY AN EXISTING LINE is issued if an existing line is encountered when incrementing to form new lines.

When AUTOMARGIN is OFF, text is not automatically carried to the next line and RETURN must be pressed before the insertion text reaches RBOUND. If you attempt to extend text beyond RBOUND in this case, EDIT2 issues the message: RBOUND EXCEEDED, TEXT BEYOND RBOUND DISCARDED, and you must re-enter the discarded text on a new line. New line prompts are automatically supplied until you terminate the insert operation with // or CONTROL-Y.

INSERT

When the character position is omitted, a direct insertion is performed. This means that you must use the FIND command to position the pointer to the desired position for the insertion (unless of course, the pointer is already a the desired position).

INSERT character position

The character position parameter determines where the insertion is to be made. You may specify a line number alone, a line number and a column position in parentheses or just a string. The character position specified mst be an accessible position or the message: POSITION NOT FOUND is issued. The insertion is made immediately prior to the specified character position.

Examples of valid character positions are:

```
INSERT 16          line number
INSERT 16(5)       line and column
INSERT 16("A")     specific character in a line
INSERT @word@      a specific word in your document
```

INSERT character position BY increment

The BY increment parameter enables you to set the increment used for this particular insertion operation. The value supplied for increment temporarily overrides the value of SET INCREMENT.

If and existing line is encountered while incrementing to form new lines for inserted text, EDIT2 terminates the insertion and issues the message: INSERT BLOCKED BY AN EXISTING LINE.

TEXT MODIFYING COMMANDS

The commands described in this section enable you to modify text which already exists in the WORK file. There are six text modifying commands:

- o MODIFY - used to alter lines interactively.
- o CHANGE - used to alter all occurences of a specified string.
- o DELETE - used to remove text from the WORK file.
- o RENUMBER - used to renumber the lines in a specified range within the WORK file.
- o MOVE - used to move text from one location to another in the WORK file.
- o COPY - used to copy text from one location to another in the WORK file.

This section describes the functions of each of these command in detail, and provides information on how to use each command.

MODIFY

The MODIFY command is used to modify lines of text interactively.

FORMAT

MODIFY[D] [line rangelist]
[Q]

MODIFY#

DESCRIPTION

The MODIFY command enables you to change specific portions of the WORK file text interactively. You specify the line or range of lines to be modified, then identify the character to be modified, and the type of changes to be made.

TEXT MODIFYING COMMANDS

MODIFY enables you to specify five types of modifications:

D - Deletes the character directly above it. A consecutive string of characters can be deleted by typing D below each character or below the first and last characters in the string to be deleted.

I - Inserts the characters typed following the I before the character directly above the I.

R - Replaces characters starting with the character directly above the R with characters typed following the R.

L - Replaces a space with leader characters to expand the line to RBOUND. Enter the desired leader character following the L (space is assumed if no character supplied).

T - Transpostion marker. Two T's are necessary to indicate the start and end of the transposition; a ^ or a P placed between the T's indicates the pivot point. The pivot is omitted when transposing two adjacent characters.

Once you have specified the line or range of lines to be modified, EDIT2 displays the accessible portion of the line and prompts with a colon. Use the space bar to position the cursor or print head directly below the character to be modified, then specify the modification (D,I,R,L, or T). Following the modification specification, press RETURN. EDIT2 then displays the line in its modified form and prompts for additional modifications.

When you have completed all necessary modifications, press RETURN in response to the colon prompt. EDIT2 then replaces the original line in the WORK file with the modified line.

When AUTOMARGIN is ON, new lines will be created automatically to contain text which exceeds RBOUND after all modifications have been made.

Only one type of modification (D,I,R,L, or T) may be used per line, with two exceptions: an insertion (I) may follow a deletion and a greater than sign (>) may follow a single D to indicate that the line from the D onward is to be deleted.

```
>>MODIFY 23:25
  23    This is line 23.
        :               r in total.
  23    This is line 23 in total.
        :                 d     dicomplete
  23    This is line 23 complete.
        :<return>
```

```
    24    This is line 24.
         ://
```

The following examples illustrate transposition and leadering:

```
>>MODIFY 24
  24    This is line 24.
        :              tt
  24    This is line 42.
        :     t ^   t
  24    This line is 42.
        :<return>

>>MODIFY 1.3
   1.3  Section I   Page 1
        :          L.
   1.3  Section I ..................... Page 1
        :<return>                              ^--(at RBOUND)
```

PROMPT AND POINTER

Once you have specified the line or range of lines to be modified, EDIT2 displays the in-bounds portion of the line and prompts with a colon. Use the space bar to position the cursor or print head directly below the character to be modified, then specify the modification (D,I,R,L,T).

After a modification has been made, the pointer is positioned one column beyond the modification.

CONTROL-Y

CONTROL-Y may be used in three ways: to signal completion of a modification, to cancel all modifications made in a line, or to terminate a MODIFY command. To signal completion, press CONTROL Y immediately after entering the modification; the line is accepted and the next line of a range is presented for modification.

Pressing CONTROL-Y in response to a colon prompt after one or more modifications have been made to a line, causes the line to be redisplayed in its original form. You may then redo your modifications or press RETURN to leave the line unchanged.

To terminate the MODIFY command without modifying the indicated line, press CONTROL-Y before any modifications have been made; you will be returned to the >> prompt.

LIMITATIONS

Only the SPACE bar or TAB key (if tabs are set) may be used to advance the cursor or print head. The BACKSPACE key or CONTROL-H must be used to move the cursor or printhead in reverse. Cursor-right and the other cursor control keys should not be used.

MODIFY adheres to the limitations imposed by LBOUND and RBOUND. Only text between these boundaries is available for modification.

MODIFYD AND MODIFYQ

When MODIFYD is specified or DISPLAY is ON, EDIT2 displays both the line number and the current contents of the line to be modified. A colon prompt appears on the next line. EDIT2 also displays the line after the modification is made. Additional modifications may be made at this point, or the line may be accepted in its current state by pressing RETURN.

When MODIFYQ is specified or DISPLAY is OFF, the line number is not displayed, only the text of the line and the colon prompt appear.

Appending the letter D or Q the the MODIFY command temporarily override the status of DISPLAY, and effect only the current execution of the command.

MODIFY#

Appending a number sign (#) to the MODIFY command instructs EDIT2 to prompt with a number sign. You in turn enter the number of the line to be modified. This permits you to specify lines for modification in any order. For example:

```
>>MODIFY#
#126
 126   This is line 126.
      :              d dione-hundred twenty-six
 126   This is line one-hundred twenty-six.
      :<return>
#//
>>
```

You must enter the number of an existing line followed by RETURN. Pressing CONTROL-Y, entering //, or pressing RETURN terminates the # prompting.

MODIFY line rangelist

You may specify a single line to be modified by entering its line number or a string within that line, or you may specify a range of lines to be modified by entering a range of line numbers such as 100/125, or a list of such ranges separated by commas. When a line rangelist is specified, EDIT2 prompts for modification one line at a time. When you press RETURN to signal that all changes to a line have been made, EDIT2 presents the next line for modification.

TEXT MODIFYING COMMANDS

CHANGE

The CHANGE command is used to alter specific portions of the WORK file.

FORMAT

```
CHANGE[D] {string                        } TO [//ASK] string
      [Q] {start column[:]stop column}
                        [/]
          [IN line rangelist]
```

DESCRIPTION

The CHANGE command enables you to change all occurrences of specific portions of the text in the WORK file. To use CHANGE, you specify:

- the old text to be changed either as a string of characters (string) or as a specific position in a line (start column : stop column)
- the new text (string) to be used in place of the old text
- the line or group of lines (line rangelist) in which the change is to be made

You may make changes to certain text occurrences such as

```
>>CHANGE @is@ TO "will be" IN 10
```

or to the contents of certain positions in a line such as

```
>>CHANGE 5:10 TO "******" IN 12.5
```

Changes may be specified for a single line

```
>>CHANGE "CHAPTER" TO "SECTION" IN 23
```

or in groups of lines such as

```
>>CHANGE "CHAPTER" TO "SECTION" IN 23/95,110/125
```

or for the entire WORK file

```
>>CHANGE "Edit2" TO "EDIT2" IN ALL
```

When AUTOMARGIN is ON, new lines are created automatically to contain text which exceeds RBOUND as a result of changes made within the line unless there is text beyond RBOUND in the original line.

You may also use the CHANGE command to change one specific portion of text rather than all occurrences of such text. The alteration of one portion of text is called a directed change because you direct EDIT2 to that portion of text to be changed by positioning the pointer.

The FIND command must be used to position the pointer for a directed change. To illustrate:

```
>>FIND 10
  10    A B C D E F H H I
        ^(1)
>>CHANGE "H" TO "G"
  10    A B C D E F G H I
```

A directed change changes only the first occurrence found beyond the current location of the pointer.

PROMPT AND POINTER

Since you specify the type and place of the change operation in the context of the command itself, CHANGE does not issue its own prompt. However, when the CHANGE command is used form a USE file, you may use the //ASK command to prompt for a replacement string.

After each change operation, the pointer is positioned to the column immediately following the last character of the changed string.

CONTROL-Y

You may press CONTROL-Y to terminate the operation of the CHANGE command at any time. Lines changed before CONTROL-Y is pressed remain in their changed form.

LIMITATIONS

The CHANGE command effects text only within the limits set by LBOUND and RBOUND. If there is text beyond RBOUND and the anticipated result of any change violates RBOUND, the warning message: RBOUND EXCEEDED, NO CHANGES MADE IN LINE line number is issued.

TEXT MODIFYING COMMANDS

CHANGED and CHANGEQ

When CHANGED is specified or DISPLAY is ON, every changed line is displayed immediately following the operation. The display includes only text between and including LBOUND and RBOUND. Changed lines are not displayed if CHANGEQ is specified or DISPLAY is OFF.

CHANGE start column:stop column TO string

This form of the CHANGE command is a directed replacement. The FIND command must be used to set the pointer to the desired line. The contents of start column:stop column are replaced by string and the length of the line is adjusted as necessary. Start and stop column specifications must not violate LBOUND and RBOUND parameters.

CHANGE start column TO string

This form of the command is a directed insertion rather than a replacement. String is inserted in the line of text beginning in the column specified as start column. The original contents of start column to RBOUND are adjusted to the right to make room for string. FIND must be used to position the pointer to the desired line.

CHANGE...TO //ASK string

The optional //ASK string parameter may only be used from a USE file; it may not be entered directly from the terminal as part of the CHANGE command.

The string which follows //ASK is issued as a prompt to the person at the terminal. The text which that person enters in response to the prompt becomes the replacement string.

To use this feature, embed a special character sequence in your document wherever the solicited text is to be placed then change all occurrences of that string to the text supplied by the person at the terminal. For example, the paragraph below has three special strings to be supplied by the person at the terminal.

> Dear <first name>
>
> Your interview has been scheduled for <time>, on <day>.
> We look forward to seeing you on that date.

IN line rangelist

When IN line rangelist is specified, all occurrences of string or start column:stop column are replaced by TO string in the line or group of lines specified. The use of only a start column causes the string to be inserted at the start column in all lines encompassed by the line rangelist.

TEXT MODIFYING COMMANDS

DELETE

The DELETE command is used to delete ranges of text, including entire lines, from the WORK file.

FORMAT

DELETE[D] [character rangelist]
[Q]

DELETE#

DESCRIPTION

The DELETE command deletes text from the WORK file. You may specify a portion of a line, an entire line, a range of lines, or ALL which indicates that the entire content of the WORK file is to be deleted.

PROMPT AND POINTER

With the exception of DELETE#, the DELETE command does not issue its own prompt as all parameters are specified as part of the command.

DELETE positions the pointer one column beyond the deleted text (at column LBOUND of the next line if entire line is deleted and at column RBOUND of the previous line if the last line in the WORK file is deleted).

CONTROL-Y

You may terminate an active DELETE command at any time by pressing CONTROL-Y. However, any text deleted prior to the reception of interrupt is irretrievably lost.

LIMITATIONS

WORK file contents removed with a DELETE command are not recoverable.

The DELETE command ignores LBOUND and RBOUND limits when deleting text. When a range of text begins in one line and ends in another, both the content and line numbers of intervening lines are deleted. When an entire line is deleted, the line number is also removed from the WORK file.

DELETED and DELETEQ

When DELETED is specified, EDIT2 displays the text of lines which are deleted entirely from the WORK file. When only a part of a line is deleted, the remaining text in that line is displayed. Nothing is displayed when ALL is specified as the range. DELETEQ turns off the display option; instead, a message indicates the number of lines deleted.

DELETE#

When # is appended to the DELETE command, no rangelist is permitted. Instead of a rangelist, you must enter the numbers of lines to be deleted one at a time. EDIT2 prompts you with a number sign prompt for the line number of the next line to be deleted; only entire lines can be deleted in this manner. Use CONTROL-Y, //, or RETURN to discontinue prompts for deletions.

DELETE character rangelist

You may specify any range of text for deletion. When only part of a line is deleted, the remaining text in the line is compressed, regardless of RBOUND and LBOUND settings, and the effected line is displayed. For example:

```
>>DELETE @how@
 705.12 The point is, however, how it is done.
>>LIST*
 705.12 The point is, however, it is done.
```

When a range of text encompasses entire lines, the content and the line numbers of those lines are deleted from the WORK file.

RENUMBER

The RENUMBER command is used to renumber a range of lines in the WORK file.

FORMAT

RENUMBER[D] line range [TO line number] [BY increment]
[Q]

DESCRIPTION

The RENUMBER command enables you to change the numbers assigned to some or all of the lines in the WORK file. You specify a single range of lines to be renumbered (line range), then optionally you specify the renumber options that you want EDIT2 to use:

- o renumber lines according to SET INCREMENT value by omitting the BY parameter.
- o renumber lines to line 1 by omitting the TO parameter.
- o renumber lines starting with the line number specified as the TO line number.
- o renumber lines according to the interval specified as BY increment.

PROMPT AND POINTER

RENUMBER does not issue its own prompt as all necessary information is included in the command parameters.

Following the renumbering operation, the pointer is positioned to column LBOUND of the line following the last renumbered line. If the last line in the WORK file is renumbered the pointer is positioned to column RBOUND of that line.

CONTROL-Y

The operation of the RENUMBER command cannot be interrupted or terminated during its execution. Pressing CONTROL-Y after the command has been entered has no effect.

LIMITATIONS

The RENUMBER command will not interleave or move lines of text. The only function that it will perform is the renumbering of lines. For example, you cannot renumber 50/60 to line 1 (the default) if there is even one line in the range of 1 to 49.99; if you try you will get the message: RENUMBER BLOCKED BY AN EXISTING LINE.

RENUMBERD and RENUMBERQ

When DISPLAY is ON or RENUMBERD is specified, both the old and new line numbers are displayed for each line renumbered. An arrow (=>) separates the two line numbers. To illustrate:

```
>>RENUMBERD 53:55 BY .1
   53    => 53
   53.01 => 53.1
   53.04 => 53.2
```

RENUMBERQ is used to disable the display facility. When Q is appended to the command, nothing is displayed during the renumbering operation. When the range ALL is specified, the new line numbers are not displayed regardless of the setting of DISPLAY.

MOVE

The MOVE command moves text from one location to another in the WORK file.

FORMAT

MOVE[D] character range [TO line number] [BY increment]
[Q]

DESCRIPTION

The MOVE command enables you to remove text from one location in the WORK file and place it in another. The repositioned text is renumbered according to the current value of SET INCREMENT unless you specify BY increment parameter. When using the MOVE command, you may specify only a single range of text to be moved (character range) and the starting line number to which the text is to be moved. You may move whole lines or parts of lines.

MOVE does not respect the settings of LBOUND and RBOUND when searching for a position, extracting text from a line range, or forming new lines. MOVE totally disregards boundary restrictions.

PROMPT AND POINTER

MOVE does not issue a prompt.

Following a move operation, the pointer is positioned at column LBOUND of the first line moved.

CONTROL-Y

MOVE assembles all lines to be moved into a temporary file before actually performing the operation. CONTROL-Y may be used to terminate the move during this part of the operation and will result in no change to the original position of the text. However, once the process of moving lines to their new location has begun, the MOVE operation cannot be interrupted or terminated. Pressing CONTROL-Y at this point has no effect.

LIMITATIONS

MOVE does not respect the settings of LBOUND and RBOUND when searching for a position, extracting text from a line range, or

forming new lines.

MOVE will not allow you to move text to an existing line number unless that number falls within the range of the move text.

MOVED and MOVEQ

When DISPLAY is ON or MOVED is specified, both the old and new line numbers of the moved text are displayed, separated by an arrow (=>). To illustrate:

```
>>MOVE 10:13 TO 70
   10    => 70
   10.1  => 71
   11    => 72
```

An asterisk is affixed to the arrow (*=>) when only part of a line is moved to a new location. The old line number is preserved if any text remains, otherwise it is deleted.

The parameter Q turns DISPLAY off. When MOVEQ is specified nothing is displayed during the move operation.

MOVE character range [TO line number]

You must specify the beginning and ending positions of text to be moved. These parameters must be separated by a colon or slash, such as 10:12 or 123/456, and constitute the range of text to be moved. If only one line is to be moved, only that line number need be used in the specification. For example to move line 10 to line 20 enter:

```
>>MOVE 10 to 20
```

You may specify the line number of the first line to receive the moved text. The line number specified may be outside the range of line numbers used for the current text, or a line number which falls between existing lines. If no line number is given, the text will be moved to the end of the work file.

BY increment

The BY increment parameter is optional. When specified, line numbers assigned to moved lines are incremented by that value if possible. When BY increment is omitted, the value of SET INCREMENT is used.

MOVE ALL TO 1

Sooner or later you will be editing a document and get a message that indicates that your WORK file is full. When this happens, the normal course is to use KEEP to store the document in a text file, use GET (naming the current work file) to make note of any special settings, then, EXIT, purge the work file, and run EDIT2 again, re-creating your old WORK file and giving it a larger limit for the maximum number of lines. However, if you have extensively revised your document, there will usually be some unused space (because this space exists as small fragments scattered throughout the work file) that can be recovered. When you MOVE ALL to 1 (or any starting line number), the entire WORK file is reconstructed internally; this recovers all unused fragments of space. You can then use the SHOW FILES command to find out how many more lines you can add.

COPY

The COPY command is used to copy text from one location in the WORK file to another.

FORMAT

COPY[D] character range [TO line number] [BY increment]
[Q]

DESCRIPTION

The COPY command copies a portion of text from one location in the WORK file to another. Upon completion of the operation, identical text exists in both the old and the new location.

The character range may include part of a line, an entire line, or a range of lines. You must specify the line number to receive the copied text, and may specify the increment by which succeeding line numbers are to be determined. When no increment is specified, an appropriate increment is used.

PROMPT AND POINTER

COPY issues no prompt since all necessary information is provided within the parameters used with the command.

Following the copy operation, the pointer is positioned at column LBOUND of the first line containing copied text.

CONTROL-Y

CONTROL-Y may be used to terminate the execution of the COPY command at any time. However, lines already copied are not automatically deleted.

LIMITATIONS

The COPY command will not replace existing lines with copied lines. Therefore, there must be a sufficient number of vacant line numbers to accommodate the specified range of text to be copied or no copy takes place and EDIT2 issues the message: INSUFFICIENT NUMBER OF LINES, NOTHING COPIED. You can obtain extra space in the target area by renumbering some of the adjacent lines or you can choose a sufficiently small increment for the BY specification.

COPY does not observe the settings of RBOUND and LBOUND. Copied text is left justified to column one in the new lines.

COPYD and COPYD

When DISPLAY=ON or COPYD is specified, both the old and the new line numbers involved in the copy operation are displayed with an arrow prefixed by an asterisk (*=>) between them. To illustrate:

```
>>COPYD 24:26 TO 150
   24    *=> 150
   24.5  *=> 151
   25    *=> 152
   25.5  *=> 153
   26    *=> 154
```

When DISPLAY is OFF or COPYQ is specified, nothing is displayed. In addition, when the range ALL is specified nothing is displayed regardless of the setting of DISPLAY.

COPY character range [TO line number]

You must specify the beginning and ending positions of text to be copied. These parameters must be separated by a colon or slash, such as 10:12 or 123/456, and constitute the range of text to be copied. If only one line is to be copied, only that line number need be used in the specification. For example to copy line 10 to line 20 enter:

```
>>COPY 10 to 20
```

You may specify the line number of the first line to receive the copied text. The line number specified may be a line at the end of the current text, or a line number which falls between existing lines. If no line number is given, the text will be copied to the end of the work file.

BY increment

The BY increment parameter is optional. When specified, line numbers assigned to copied lines are incremented by that value if possible. When BY increment is omitted, the value of SET INCREMENT is used.

EDIT2 provides a group of commands that are used exclusively to position and prepare the contents of the WORK file for printing. These command are called formatting commands and are listed below:

- o FILL - used to fill each line with whole words so that the space between LBOUND and RBOUND contains the maximum amount of text possible.
- o JUSTIFY - used to adjust a line to exactly fill the space between LBOUND and RBOUND.
- o COMPRESS - used to replace multiple spaces in a line with single spaces.
- o CENTER - used to center a line between LBOUND and RBOUND.
- o PRINT - used to print the final fully formatted contents of the WORK file.

FILL

The FILL command is used to fill each line in a specified range with as many whole words as posible, so that the space between LBOUND and RBOUND contains the maximum number of words.

FORMAT

FILL[D] [line range] [BY increment]
[Q]

DESCRIPTION

The FILL command takes the collective contents of the lines in the line range and forms new lines which each contain the maximum number of words that will fit between LBOUND and RBOUND. The starting line is the first line in the specified line range and the lines are renumbered using the value of SET INCREMENT, the optional BY increment value, or an appropriate value if none is specified. New lines are generated as required, and resulting empty lines are automatically deleted.

When HYPHENATION is used (SET HYPHENATION = ON), words are displayed for you to hyphenate; you indicate the placement of the hyphen by spacing to the character in the word which should follow the hyphen, typing a hyphen, and pressing RETURN.

The following rules are used during a FILL command operation:

- A single space is inserted between the last word of a line and the first word taken from the next line.
- Two spaces are inserted after a period, a question mark, or an exclamation point when they are followed by two or more spaces or occur at the end of the original line.
- When the last character of a line is a hyphen, the hyphen is removed if HYPHENATION is ON, otherwise it is preserved and no spaces are inserted between the hyphen and the first word taken from the next line.
- A line is not filled if it is followed by a blank line. Filling resumes with the next non-blank line.
- A line is not filled if it is followed by a labeled line. Filling resumes with the labeled line.
- A line is not filled if the first word of the following line is not aligned with the logical beginning of the current line. The logical beginning is the first non-blank character following a label if LABELS is ON or the first non-blank character in the line.
- Blank lines which already exist in the document remain, blank lines resulting from the fill operation are deleted automatically.

The FILL command will adjust text to make it fit between LBOUND and RBOUND. Generally, it is not necessary or advisable to set LBOUND to other than column 1 as FILL automatically follows indentation, filling only aligned text as described above. RBOUND may be moved freely to achieve the desired line width for your final document.

FILL will always bring all text which is outside the bounds within the bounds so it can be used to reduce the line width in any range. If you want to reduce the line width for a note or quoted passage for example, you can reset both LBOUND and RBOUND to achieve indention on both sides.

POINTER AND PROMPT

The FILL command issues no prompt.

The pointer is positioned at column LBOUND of the last line produced by the filling operation.

CONTROL-Y

The FILL command makes use of a temporary file which is automatically created. No line is replaced by a filled line until the entire range of lines has been successfully processed and it has been determined that there is sufficient space in the range to accommodate the filled lines. Therefore, the filling operation may be terminated at any time before the first line is displayed. Once FILL actually begins replacing the original lines with filled lines, its operation cannot be terminated.

LIMITATIONS

FILL operates on entire lines only. It does not respect the settings of LBOUND or RBOUND when it extracts the contents of a line, but ensures that the filled lines begin at LBOUND and end as close to RBOUND as possible.

If lines are too long, they are broken at spaces if possible and new lines are created as required. If there are no spaces in a line that is too long, it is broken at RBOUND.

FILLD and FILLQ

When FILLD is specified or DISPLAY is ON, all filled lines are displayed following the filling operation. If FILLQ is specified or DISPLAY is OFF, nothing is displayed. Hyphenation decisions are displayed regardless when HYPHENATION is ON.

FILL line range BY increment

You may specify the lines to be filled by supplying a single range of lines. Only one range may be specified at a time; a rangelist is not allowed here. The first line is used as the starting line number for the filled text; subsequent lines are always renumbered as required.

When you are revising a document, you will find that the blank line notation ("" or @@) is quite useful to refill a paragraph that has been altered. It is also useful to remember that the pointer is in the modified line. This makes it easy to refill the text from the modified line to the end of the paragraph; use the command: FILL */"", or better still, make a definition for FILL */"" so you don't have to retype the entire command each time.

HYPHENATION

During execution of the FILL command, you may be presented with words that need to be hyphenated. You indicate placement of the hyphen by spacing to the character in the word which should follow the hyphen, typing a hyphen, and then pressing RETURN.

For example:

```
   HYPHENATE THE FOLLOWING WORD(S):

   breakf ast
        -<return>

   dinne r
      -<return>
```

You must place the hyphen on or before the space provided in the word. If there is no valid breaking point available for the word simply press return and the word will remain intact.

If a slash (/) appears in a hyphenation decision "word", typing a hyphen under the slash (or at the space if it immediately precedes the slash) will cause the slash to be used at the break point instead of a hyphen. For example,

```
   HYPHENATE THE FOLLOWING WORD(S):

   EDIT2/3 000
        -<return>

   printer /keyboard
          -<return>
```

JUSTIFY

The JUSTIFY command is used to adjust the contents of a line to exactly fill the space between LBOUND and RBOUND.

FORMAT

JUSTIFY[D|Q] [line rangelist]

JUSTIFY#

DESCRIPTION

The JUSTIFY command adjusts the contents of a line by replacing existing spaces with two or more spaces so that a line of text is proportioned across the length of the line. JUSTIFY places the first character of the line in column LBOUND, and the last non-blank character in column RBOUND. The following rules apply to the operation of the JUSTIFY command:

- Leading spaces are not affected; justification begins with the first space following the first non-blank character in the line.
- A line is not justified if the following line in a line range is a blank line.
- Justification is performed within each range separately, regardless of the number of ranges specified in the line rangelist.
- Single lines may be specified for justification, and are justified if each contains at least one internal space.

JUSTIFYD and JUSTIFYQ

When JUSTIFYD is specified or DISPLAY is ON, all lines in the range are displayed as they are justified. When JUSTIFYQ is specified, nothing is displayed during the operation.

LIMITATIONS

The JUSTIFY command operates only within the limits set by LBOUND and RBOUND. If a line contains text which falls outside of these boundaries, justification occurs only within the boundaries.

PROMPT AND POINTER

When JUSTIFY# is specified, EDIT2 prompts for the line number of each line to be justified. Once all lines have been specified, CONTROL-Y, //, or RETURN will terminate the # prompting.

At the completion of a justification operation the pointer is positioned at column LBOUND of the line following the last line justified.

CONTROL-Y

CONTROL-Y may be used to terminate the justifying operation at any point. Lines justified before CONTROL-Y was pressed remain justified.

COMPRESS

The COMPRESS command is used to replace multiple spaces with single spaces. It is particularly useful for "unjustifying" previously-justified lines.

FORMAT

COMPRESS[D] [line rangelist] [,LEFT]
[Q]

COMPRESS#

DESCRIPTION

The COMPRESS command replaces multiple spaces in each line in the line rangelist with single spaces according to the following rules:

- Leading spaces in a line are not replaced.
- When a word ends with a period, a question mark, an exclamation point, or a colon, following multiple spaces are replaced by two spaces.
- Multiple spaces in all other circumstances are replaced by a single space.

COMPRESS operates only within the limits set by LBOUND and RBOUND.

PROMPT AND POINTER

When COMPRESS# is specified, EDIT2 prompts for the line number of each line to be compressed. Once all lines have been specified, CONTROL-Y, //, or RETURN will terminate the # prompting.

After a line is compressed, the pointer is positioned at columnn LBOUND of the following line.

CONTROL-Y

CONTROL-Y may be used to terminate the compress operation at any point. Lines compressed before CONTROL-Y was pressed, remain compressed.

COMPRESSD and COMPRESSQ

When COMPRESSD is specified or DISPLAY is ON, each line is displayed after it is compressed. If COMPRESSQ is specified, nothing is displayed during the compress operation.

COMPRESS line rangelist, LEFT

When the LEFT option is specified, COMPRESS does not compress spaces internal to the line of text; rather, it removes all spaces before the first character in the line. This funtion is equivalent to the command CHANGE LEFT/FIRST-1 TO "" IN line rangelist. The option name (LEFT) cannot be abbreviated.

COMPRESS#

Appending a number sign to the COMPRESS command allows you to specify the numbers for lines you want compressed in any order. EDIT2 then prompts with a number sign for each line you wish to compress. Press RETURN with no number, //, or CONTROL-Y to terminate the number sign prompting.

CENTER

The CENTER command is used to center a line between LBOUND and RBOUND.

FORMAT

CENTER[D] [line rangelist]
[Q]

CENTER#

DESCRIPTION

The CENTER command adjusts the position of the contents of a line so that it is centered between LBOUND and RBOUND.

CENTER operates only within LBOUND and RBOUND. Only the text within these parameters is centered, even if text exists outside of the boundaries.

PROMPT AND POINTER

When CENTER# is specified, EDIT2 prompts for the line number of the next line to be centered. Once all lines have been specified, CONTROL-Y, //, or RETURN will terminate the # prompting.

After a line is centered, the pointer is positioned at column LBOUND of the following line.

CONTROL-Y

CONTROL-Y may be used to terminate the centering operation at any point. Lines already centered remain so.

CENTERD and CENTERQ

When CENTERD is specified or DISPLAY is set to ON, each line is displayed after it is centered. If CENTERQ is specified, nothing is displayed.

CENTER#

Appending a number sign to the CENTER command allows you to specify the numbers for lines you want centered in any order. EDIT2 then prompts with a number sign for each line you wish to center. Press RETURN with no number, //, or CONTROL-Y to terminate the number sign prompting.

PRINT

The PRINT command is used to print fully formatted listings of your document.

FORMAT


```
PRINT   [line rangelist]
        [,DRAFT]
        [,MEMO]
        [,WAIT]
        [,OFFLINE] or [,TO filename] or [,TO #PRINTER]
        [,COPIES = number of copies]
        [,LEFT = column number]
        [,RIGHT = column number]
        [,SPACING = number of lines]
        [,PAGENUMBER  [ = CONTINUE    ]]
                      [ = page number ]]
```

The following embedded subcommands are only executed when you use the PRINT command (LBOUND must equal one).

```
          {OFF    }
.FLAG = {ON     } [,"flag character"]
          {integer}
```

.HEADING = string	.FOOTING = string
.NEED = integer	.SKIP [= integer]
.NEWPAGE	.SPACING = integer
.ODDPAGE	.SUPPRESS
.PAGENUMBER = page number	
.UNON	.UNOFF

DRAFT

DRAFT causes line numbers to be issued along with the formatted document for reference when editing. Lines that have no numbers were generated by .SKIP subcommands; missing numbers may result because of top-of-page formatting or subcommand lines.

MEMO

MEMO causes the heading and footing to be suppressed on the first page of a document.

OFFLINE

OFFLINE issues a file equation for, and directs the listing to the device named LP. (If there is no such device on your system use TO filename and your own file equation; be sure to add CCTL to your equation.) The file equation for OFFLINE is :FILE EDITLIST; DEV=LP; CCTL. You may override this equation after EDIT2 is running with a :FILE equation as illustrated below.

```
>>:FILE EDITLIST; DEV=FASTLP; CCTL
```

TO filename

The TO filename parameter enables you to specify a file to which the lines in the line rangelist are to be printed. When specified, a filename results in the text being printed to that file rather than to the terminal.

To send the listing to a device (such as a line printer), you must first name the device in a file equation. For example,

```
>>:FILE MYOUT; DEV=MYPRINT; CCTL
>>PRINT TO *MYOUT
```

The special file name #PRINTER may be used to print directly to a terminal-connected printer; be sure to use the WAIT parameter if single sheet paper is used.

NOTE

The HP 2640 series cartridge tape units cannot be specified as filenames for the PRINT (or LIST) commands; instead the KEEP command must be used for this purpose.

COPIES

The COPIES parameter is used to specify the number of copies desired. However, this can be more efficiently accomplished with the copies parameter in the MPE file equation. For example,

```
>>:FILE EDITLIST; DEV=LP,,3; CCTL
                            <--- number of copies
```

LEFT = column number -- RIGHT = column number

The LEFT and RIGHT parameters specify print margins to be used for an incidence of the PRINT command. The default values are LEFT = 5 and RIGHT is set automatically to accommodate the text between LBOUND and RBOUND if not set explicitly. The values of LBOUND and RBOUND determine the extent of the text to be listed.

Printed lines, including line numbers if present, begin in the column position specified by LEFT. LEFT must be less than or equal to RIGHT, or the message: MARGINS PRECLUDE TEXT is issued and the PRINT command is not executed.

Unless TO filename or OFFLINE is specified, a value for RIGHT that exceeds WIDTH precipitates the message: RIGHT MARGIN EXCEEDS WIDTH and the PRINT command is not executed. The value of RIGHT may exceed WIDTH only if TO filename or OFFLINE is specified.

SPACING = number of lines

The SPACING parameter causes listed lines to be spaced according to the value specified as number of lines. The default value is 1 (for normal single spacing). Values of 2, 3, or 4 may be specified.

PAGENUMBER

Page numbering begins at 1 unless an alternate starting page number is specified using the PAGENUMBER parameter.

The PAGENUMBER = CONTINUE option permits you to continue consecutive page numbering when listings are concatenated by executing consecutive PRINT commands. When CONTINUE is specified, the first page is given a number one greater than the last page of the previous listing.

EMBEDDED SUBCOMMANDS

Subcommands are embedded in the WORK file and begin with a period. Furthermore, subcommands must be on lines separate from text and the period for the first subcommand on a line must be in column one. Multiple subcommands on the same line must be separated by semicolons. The period is optional on the second

and following subcommands on a line. The subcommands are described below.

```
        {OFF     }
.FLAG = {ON      } [,"flag character"]
        {integer}
```

Lines are marked by printing the flag character two columns to the right of the right margin. The default character is the exclamation mark (!). This character may be changed to any printing character by specifying the desired character in the FLAG subcommand. Flagging is useful for marking lines that have changed since the last edition. This allows the reader to quickly determine what has changed. (Flagging was purposely not used in this manual to force you to read the whole thing again; you'll be suprised by how much you didn't really understand the first time.)

Lines may be flagged using either of two methods. The .FLAG subcommand may be turned ON causing all following lines to be flagged until a .FLAG = OFF subcommand is encountered. Alternatively, the subcommand .FLAG = integer may be used to flag the number of lines specified by the value of integer. A .FLAG = integer subcommand is ignored if FLAG is already ON. A .FLAG = ON while an integer is in effect, over-rides the integer specification.

.HEADING = string

The .HEADING subcommand permits you to alter the value of the heading string dynamically. The value is altered for the heading on the next page and all suceeding pages or until another HEADING subcommand is encountered. You may use the names #PAGE, #DATE, and/or #TIME to have the page number, date, and or time inserted in your heading when the document is printed.

.NEED = integer

The .NEED subcommand is used to request that a section of text not be split across a page boundary. The value of integer specifies the number of lines which are to remain contiguous if possible. The text is automatically printed at the top of the next page if insufficient space remains on the current page to fill the NEED requirement.

.NEWPAGE

The .NEWPAGE subcommand permits you to force text to the top of a new page. If a new heading is desired, it must be specified before .NEWPAGE is encountered to be in effect for the new page.

.ODDPAGE

The .ODDPAGE subcommand allows you to force text to the beginning of the next odd numbered page. If a new heading is desired, it must be specified before .ODDPAGE is encountered to be in effect for the new page. If the current page is an odd number page, the intervening page is numbered but left void of text.

.PAGENUMBER = page number

The .PAGENUMBER subcommand permits you to dictate the number of the page anytime before it is printed (it must be done before the new page if #PAGE is used in the heading). This is useful to reset the page number to 1 at the beginning of a new section or appendix of a document.

.FOOTING = string

The .FOOTING subcommand permits you to alter the value of the footing string dynamically. The value is altered for the footing on the current page and all suceeding pages or until another FOOTING subcommand is encountered. You may use the names #PAGE, #DATE, and/or #TIME to have the page number, date, and or time inserted in your footing when the document is printed.

.SKIP [= integer]

The .SKIP subcommand permits you to skip any number of lines, specified by integer. If "= integer" is omitted, one blank line is printed.

.SPACING = integer

The .SPACING subcommand permits you to override the default spacing or the value assigned by the SPACE parameter of the PRINT command. Whenever a .SPACING subcommand is encountered, the value specified replaces whatever value is in effect. The new value stays in effect until it is changed by another .SPACING subcommand or the end of the line rangelist is encountered.

.SUPPRESS

The .SUPPRESS subcommand suppresses the linefeed on the following line. This subcommand permits you to underline or overprint a line on a device which permits suppression of the linefeed.

.UNON and .UNOFF

The .UNON and .UNOFF subcommands provide an alternate method for underlining text. When used, any text enclosed in braces (e.g., {underline this text}) will be underlined. UNON enables this feature and is normally the first line in your work file if used. UNOFF disables the feature for all following lines.

CAUTION: The braces take up room in the line, so justified lines will be short by the number of braces in the line.

A single brace will cause text from the beginning of the line to the brace, or from the brace to the end of the line, to be underlined. For example,

```
>>ADD
    1  :This is an {example of
    2  :underlining} using UNON.
```

File commands enable you to perform various file-related operations. The WORK file is the primary file used with EDIT2; however, other files may also be employed from time to time to store text or commands for later use. The following commands enable you to create and control such files from EDIT2. There are seven commands related to the use of files.

- o GET - gets an existing WORK file or designates a new WORK file.
- o KEEP - saves the contents of the WORK file.
- o TEXT - copies (ASCII) TEXT files into the WORK file.
- o MERGE - combines numbered lines from a TEXT file with your current WORK file.
- o DISPLAY - displays a message from the USE file at the terminal.
- o //ASK - used in the USE file to request input from the user.
- o USE - designates a file (USE file) as a source of commands.

This section describes the functions of each of these commands in detail, and provides information on how to use each command.

GET

The GET command permits you to change to another WORK file without leaving the EDIT2 program.

FORMAT

GET work file name

DESCRIPTION

When you use GET, the current work file is left in an orderly state and the new work file is accessed. If the supplied name does not refer to an existing work file a new one is automatically created with that name. If the name given is the name of the current work file you will be given the summary information about the status of your work file.

KEEP

The KEEP command is used to save the contents of the WORK file.

FORMAT

```
KEEP { [TO] filename          } [,UNNUMBERED]
     {line range TO filename } [,REAR       ]
```

DESCRIPTION

The KEEP command saves all or part of the contents of the WORK file by storing a copy of it in the specified file. Entire lines from the WORK file are copied without regard for the settings of LBOUND and RBOUND. The contents of the WORK file are not affected by a KEEP operation. Once text is stored in a file, a subsequent KEEP to that file results in the replacement of the ENTIRE CONTENTS of that file with the new text.

PROMPT AND POINTER

KEEP does not issue a prompt.

Following a keep operation, the pointer is positioned at column LBOUND of the line following the last line kept.

CONTROL-Y

Pressing CONTROL-Y will terminate the KEEP operation and return you to the >> prompt.

```
KEEP [TO] filename  [,UNNUMBERED]
                    [,REAR      ]
```

The filename specified refers to the file in which text is to be stored. If the named file exists, EDIT2 will issue the warning: ABOUT TO OVERWRITE FILE 'filename'. If you do nothing to stop it, EDIT2 will purge the existing file and create a new file (of the same name) which is sized to contain the number of lines you have directed it to keep. If the named file does not exist, a new file is created.

Unless the UNNUMBERED parameter is specified, the line numbers in the WORK file are converted to ASCII characters and prefixed to the contents of the line.

When REAR is specified, line numbers are multiplied by 10, padded on the left with zeros, and placed in the last eight columns with no decimal point. Only text within the bounds is kept and the record length of the file is: RBOUND - LBOUND + 9.

```
KEEP  [line range] TO (#TCL) [,NEXT] [,UNNUMBERED]
                      (#TCR)         [,REAR      ]
```

Users of HP 2640 series terminals may use the special file names #TCL and #TCR to refer to the left and right cartridge tapes respectively. These names may optionally be followed by ",NEXT" to indicate that the next available file on the tape should be used. When NEXT is not specified, the text is stored in the first file on the tape. A particular file number on the tape may NOT be selected when storing text but may be used for retrieval. For example:

```
>>KEEP #TCL
230 lines kept to #TCL in file 1.

>>KEEP FIRST+5 / 17 to #TCR,NEXT
12 lines kept to #TCR in file 4.
```

The file names #TCL and #TCR may also be used with the TEXT, ADD, REPLACE, and MERGE commands to retrieve the stored text. See examples under each of these commands for the exact format of the command.

KEEP line range TO filename

You may optionally specify a line range to be saved rather than the entire contents of the WORK file. When a line range is specified, the keyword TO is required before the name of the file. For example:

```
>>KEEP 15/20 TO NAILFILE
6 lines kept to file NAILFILE.
```

TEXT

The TEXT command is used to copy the contents of any ASCII file into the WORK file.

FORMAT

```
TEXT filename [(file range)] [,UNNUMBERED]
                             [,REAR      ]
```

DESCRIPTION

With the TEXT command, you may copy the contents of an ASCII file into the WORK file. TEXT may be used to copy an entire file or only a portion of a file. The text may be numbered or unnumbered. You don't usually have to tell EDIT2 whether the lines in the file are unnumbered or numbered REAR. When you don't specify one of these options, EDIT2 assumes first that the lines have numbers on the front. If that assumption fails, it looks for numbers at the rear of the lines. If that fails, it copies them as unnumbered lines. Warnings are given if you use an option and the file does not contain lines of the type you claim. Sometimes a number (in your text) at the beginning of the first line of the file will mislead EDIT2; in that case assert your superior intelligence and insist that the file contains unnumbered lines by using the ",UNNUMBERED" option. There are other cases where you can assert yourself simply by ignoring the warning messages.

The current contents of the WORK file are entirely replaced by the text copied into the file. (Use the ADD command to append text from a file to the WORK file or the MERGE command to merge numbered lines into the WORK file.)

PROMPT AND POINTER

The TEXT command does not issue a prompt. All required data is provided in the form of attendant command parameters.

Following a TEXT command operation, the pointer is positioned in column LBOUND of the first line in the WORK file.

CONTROL-Y

CONTROL-Y may be used to terminate the TEXT operation at any time. When CONTROL-Y is detected, the number of lines entered into the WORK file to that point is reported.

TEXT filename

To copy the entire contents of a file into the WORK file, simply specify the command TEXT followed by the name of the file to be copied. If the first line has no line number, unnumbered is assumed and vice-versa.

```
TEXT  {#TCL} [,file number] [(file range)] [,UNNUMBERED]
      {#TCR}                                [,REAR      ]
```

Users of HP 2640 series terminals may use the special file names #TCL and #TCR to refer to the left and right cartridge tapes respectively. These names may optionally be followed by a file number to indicate that the text is to be taken from a specific file on the tape. When a file number is not specified, the text is copied from the first file on the tape. For example:

```
>>TEXT #TCL
230 lines in your WORK file.

>>TEXT #TCR,3
12 lines in your WORK file.

>>TEXT #TCL,2 (30/50)
21 lines of text in your WORK file.
```

The file names #TCL and #TCR may also be used with the ADD, REPLACE, and MERGE commands to retrieve the stored text. See examples under each of these commands for the exact format of the command.

```
TEXT filename (file range) [,UNNUMBERED]
                           [,REAR      ]
```

The file range parameter enables you to specify that only a portion of a file is to be copied into the WORK file. This form of the TEXT command without the UNNUMBERED parameter, indicates that lines with the numbers specified are to be copied into the WORK file.

When the UNNUMBERED parameter is also specified, the file range parameter is interpreted as relative lines (records), indicating positions relative to the beginning of the file.

MERGE

The MERGE command is used to merge numbered text from a TEXT file with text already in the WORK file.

FORMAT

```
MERGE[D]  filename [(file range)] [,REAR]
     [Q]

MERGE#
```

DESCRIPTION

The MERGE command enables you to merge numbered text from a TEXT file with text already in the WORK file. You specify the name of the file which contains numbered lines of text. These lines will then be added as new lines or replace lines that have the same numbers as the merged lines. If only a portion of the file is to be merged, you must also provide the line numbers (file range) of the first and last lines to be merged. Alternately, you may supply numbered lines from the terminal.

PROMPT AND POINTER

Appending a number sign to the command indicates that you wish to be prompted for lines of text to be merged into the WORK file. You will be prompted with a number sign for the number of the line you want to merge. If the line number has not been used, you will be prompted with a colon to add the new text. If the line number exists, you will be shown the existing text and prompted for replacement text for the entire line.

Following a merge operation, the pointer is positioned at column LBOUND of the first line following the merged text.

CONTROL-Y

A merge operation may be terminated at any time by pressing CONTROL-Y. If lines are being merged manually (#), pressing CONTROL-Y during or at the end of a line indicates that the line is the last to be merged. Otherwise, press CONTROL-Y in response to the # prompt to terminate the prompting for line numbers.

LIMITATIONS

MERGE does not respect LBOUND and RBOUND restrictions when a file is specified -- entire lines are added or replaced by the command. Merged lines always begin in column one. For merging, lines must always be numbered.

MERGED and MERGEQ

When DISPLAY is ON or MERGED is specified, both the old and new lines are displayed when a replacement is made. Lines which don't replace existing lines are not displayed. When MERGEQ is used, no lines are displayed.

FILE-ORIENTED COMMANDS

DISPLAY

The DISPLAY command is used to display a message at the terminal from a USE file.

FORMAT

DISPLAY unquoted string

DESCRIPTION

You use the DISPLAY command to display a one-line message of up to 255 characters at the terminal. This command is intended for use in a USE file. It permits you to send messages to the terminal to report on the progress being made at critical junctures, to inform you that a specific stage has been reached, or give information on what is expected in response to a //ASK command.

//ASK

The //ASK command is used in a USE file to request input from the terminal.

FORMAT

//ASK string

DESCRIPTION

The //ASK command may only be used when operation is being driven from a USE file. It permits you to design a USE file that in effect tailors the prompts for specific tasks.

Normally, you must anticipate all prompts and supply suitable responses within the USE file. However, the //ASK command lets you prompt the user at the terminal for a textual response rather than supplying the text in the USE file. The content of the string supplied with the //ASK command is printed at the terminal as a prompt to the user in place of the normal prompt for the particular command given. The user's response is accepted as the textual response to the underlying promp. To illustrate:

```
>>SET //=OFF
//COMMANDS..OFF
>>ADDQ
:DISPLAY Add lines in response to the 'New line:' prompt.
:DISPLAY Press CONTROL-Y when done.
:ADD 20 BY 2
://ASK "New line:"
:<next command>
   etc.
```

FILE-ORIENTED COMMANDS

USE

The USE command is used to designate a file, called a USE file, to be used by EDIT2 as a source of input.

FORMAT

USE[D] filename
   [Q]

DESCRIPTION

You employ the USE command to read and execute commands stored in the named USE file. Each string in the file is read as a command string or as textual input.

USE files are used to operate in a "batch-like" mode; that is, you anticipate input that will do a certain task and supply it from a file. A USE file may contain both commands and textual input required by those commands.

A number of commands may be entered on one line separated by semicolons. You may use the "trailing ampersand" convention for long lines. Use // in place of CONTROL-Y but remember that // may not be appended to the end of a line of text; it must be the first and only entry on a line in the USE file.

When a command requires textual input, it must be the last or only command on a line in the USE file. The following lines (until // is encountered or the command requires no further text as in REPLACE) are interpreted as text. The following line may optionally contain the //ASK command.

Execution from the USE file terminates normally when the end of the file is encountered. Control is returned to the terminal user, unless EXIT appears in the USE file.

USED AND USEQ

USE defaults to its own quiet mode (i.e., USE and USEQ are equivalent). When USED is specified, commands from the USE file are displayed just before they are executed; this permits you to "debug" your USE file. The D and Q here have no effect on other commands with the exception of the SET command; verifications are not displayed for settings made from a USE file unless D is specified.

If an error occurs due to faulty command format or during the execution of a command, an error message is printed followed by the message USE FILE ABANDONED.

The following examples demonstrate the creation and use of USE files.

Example 1:

```
>>ADD
   1    :DISPLAY This USE file fills and justifies your
   2    :DISPLAY document, then produces draft and final
   3    :DISPLAY copies on a line printer.
   4    :SET LBOUND=1,RBOUND=65,HYPHENATION=ON
   5    :FILLQ
   6    :SET FOOTING=CENTER,"-#page-"
   7    :JUSTIFYQ ALL
   8    :PRINT DRAFT,OFFLINE
   9    :PRINT LEFT=10,OFFLINE
  10    :DISPLAY
  11    :DISPLAY Your listings are finished.

>>KEEP PRODUCE,UNNUMBERED
New text file PRODUCE about to be created.
11 lines kept to file PRODUCE.

>>TEXT DOCUMENT
WORK file cleared.
19 lines of text in your WORK file.

>>USE PRODUCE

This USE file fills and justifies your
document, then produces draft and final
copies on a line printer.

Your listings are finished.
>>
```

Example 2:

```
>>SET //COMMANDS=OFF
//COMMANDS.... OFF

>>DELETE ALL
WORK file cleared.
```

```
>>ADDQ
:DEL ALL
:DIS
:DIS Type comments in response to the COMMENT: prompt.
:DIS Use CONTROL-Y to stop.
:DIS
:ADDQ
:This is the first line; it came from the USE file.
:Another line was also added from the USE file.
:What do you think of that?
://
:LIST
:DIS
:ADD
://ASK "COMMENT: "
:DIS
:DIS I'm sorry you feel that way about me.
:CHANGEQ LA+2 to "BUT I LIKE IT." IN FIRST+3/LAST
:DIS
:LIST
:DIS See, I always get the last word.
:DIS And you are banished from EDIT2.  Goodbye!
:DIS
:EXIT
:<CONTROL-Y>

>>SET //C=ON
//COMMANDS.... ON

>>KEEP FUN,UNN
New text file FUN about to be created.
24 lines kept to file FUN.

>>USE FUN
WORK file cleared.

Type comments in response to the COMMENT: prompt.
Use CONTROL-Y to stop.

1    This is the first line; it came from the USE file.
2    Another line was also added from the USE file.
3    What do you think of that?

COMMENT: To tell the truth, I'm not impressed.
COMMENT: I suppose that will offend you.
COMMENT: Sorry about that.
COMMENT: //

 I'm sorry you feel that way about me.

1    This is the first line; it came from the USE file.
2    Another line was also added from the USE file.
```

```
3    What do you think of that?
4    To tell the truth, I'm not impressed. BUT I LIKE IT.
5    I suppose that will offend you. BUT I LIKE IT.
6    Sorry about that. BUT I LIKE IT.

 See, I always get the last word.
 And you are banished from EDIT2.  Goodbye!

END OF PROGRAM
```

APPENDIX A

EDIT2/3000 vs EDIT/3000

The following features are available with EDIT2/3000 and not with EDIT/3000:

1. *Rangelists* for LIST and FIND.
2. A string may be used to indicate a column, e.g., INSERT 5("word").
3. Word search indicated by using "@" delimiter.
4. Column guide displayed by SHOW command.
5. MPE commands can be used *directly* without exiting EDIT2 subsystem (BUILD, FILE, LISTF, PURGE, RENAME, REPORT, RESET, SAVE, SECURE, SETMSG, SHOWDEV, SHOWJOB, SHOWOUT, SHOWTIME, STREAM, TELL, and TELLOP).
6. FIND with option ALL will locate and count *all* occurrences of a literal string or a word.
7. FIND can be limited or directed by a rangelist.
8. COPY and MOVE commands (no HOLD file or GATHER command).
9. RENUMBER command (does not move lines).
10. MERGE command (combined add and replace for merging numbered lines into WORK file).
11. "#" convention permitted for ADD, MODIFY, REPLACE, DELETE, MERGE, JUSTIFY, COMPRESS, and CENTER commands. This permits user to be prompted with a "#" for the number of each line to added, modified, etc.
12. Modification of previous line is possible while adding lines (//MODIFY in place of text on next line to be added).
13. Tabs supported for all terminal-oriented textual input (ADD, REPLACE, INSERT, and MODIFY).
14. All SET parameters are saved with the WORK file which is always saved (i.e., EXIT implies KEEPQ).
15. DEFINE command permits users to give names to commonly used command sequences.
16. Multiple copies can be specified in the LIST and PRINT commands.
17. Automatic recovery of work file after system or program bug crash.

EDIT2/3000 — ADVANCED EDITING AND FORMATTING

1. Automatic folding of input lines at right margin during ADD.

2. Recognition of "words" for searching.

3. Recognition of "paragraphs" for formatting.

4. Recognition of "labeled" lines (as used in outlines) for automatic indentation of line fall-off.

5. User prompting available from CHANGE command when USE file is controlling EDIT2/3000. This permits easy fillout of form letter fields.

6. Length of lines can be adjusted to fit any desired margin both during and after input of text.

7. Interactive hyphenation (user makes decision).

8. Indentation automatically preserved during line filling operation.

9. Interactive formatting using FILL, JUSTIFY, COMPRESS, and CENTER commands; all are controlled by range specifications.

10. Leadering with user-specified character (for tables of contents).

11. Automatic page numbering with running heading and footing lines (lines may be positioned flush left, centered, flush right, or facing on opposite pages).

12. Justification by insertion of extra spaces to fill line to right margin.

13. COMPRESS command undoes justification.

14. Up to quadruple line spacing — can be controlled by embedded subcommand.

15. Page breaks automatically controlled to prevent widowed lines or can be forced by embedded subcommands.

16. Space can be reserved for figures with embedded subcommand.

17. Groups of lines can be automatically forced to a new page when there is insufficient space to keep them together on current page.

18. Flag character (user set) can be placed on changed lines using embedded subcommands.

19. Full control over page size, position of heading and footing lines, and margins.

20. Alternate pages can be offset for stapling or hole punches on back-to-back copy.

COMPATIBILITY ISSUES

A user of EDIT/3000 who decides to use EDIT2/3000 will be faced with the following significant differences:

1. More than one character is often required to indicate which command is desired. The rule is that enough characters must be supplied to distinguish commands and keywords (with the exception noted in "2" below). The number of characters required is context-sensitive; for example, "L" is sufficient when referring to the LAST line, but "LA" is required to refer to the last column to distinguish LAST from LEFT.

2. The keywords TO, BY, IN, and FROM must be fully spelled out and may not be replaced by commas as in EDIT.

3. The KEEP command (when UNNUMBERED is *not* specified) places an ASCII representation of a line number at the front of each line of text; the length of the number is variable and may contain a decimal point. (This is not acceptable to compilers.) (See page A-6 for further information about line numbers.)

4. Strings must be delimited by quotes or "@". EDIT permits any delimiter.

5. The GATHER command has been replaced by the COPY, MOVE, and RENUMBER commands.

7. No PROCEDURE or advanced (WHILE, etc.) commands.

8. The concept of formatting *interactively* in the WORK file.

8. WORK files are not automatically purged.

9. Line numbers are limited to 6 digits (.01 to 9999.99).

10. No JOIN command; use ADD *line number* FROM *filename* or MERGE *filename*.

11. Cannot INSERT from a file.

EDIT2 vs EDITOR: SOME DIFFERENCES

An aid for users of EDIT/3000 (:EDITOR) converting to EDIT2/3000

This is a list of some of the differences between EDIT/3000 (:EDITOR) and EDIT2/3000 (:RUN EDIT2.PUB.SYS).

The exact form of many EDIT2 commands is somewhat different from EDITOR command forms. Use the EXPLAIN (E) command as a guide to the exact form of EDIT2 commands whenever you need help.

```
>>EXPLAIN
You may ask to have any of the following explained:
```

Add	Delete	Get	Modify	SEt
CEnter	DIsplay	Insert	MOVe	SHow
Change	Explain	Justify	Print	Text
COMpress	EXIt	Keep	RENumber	Use
COPy	FILl	List	Replace	
DEFine	Find	MErge		
//	//Ask	//Exit	//Help	//Modify
POsition	RAnge			

The above table shows all command names, with the acceptable abbreviations in upper case. Note that the commands may be abbreviated by one, two, or three letters. In some cases, several commands begin with the same letter or letters, and there are conventions for abbreviating these. For example: C for CHANGE, CE for CENTER, COM for COMPRESS, COP for COPY, M for MODIFY, MOV for MOVE, E for EXPLAIN, EXI for EXIT, etc....

When in doubt, use 3 letters and you can't go wrong!

The following table shows some major differences between EDITOR and EDIT2 commands:

Instead of (EDITOR command)	Use (EDIT2 command)
END	EXIT
GATHER	RENUMBER, MOVE, or COPY
HOLD . . . ,ADD . . . ,HOLD	COPY
JOIN	ADD . . . FROM . . .
Q	DISPLAY
VERIFY	SHOW
WHILE	— —
XPLAIN	EXPLAIN
Z	DEFINE

Be particularly careful of using G for the old GATHER command! G means GET (to get another work file). Use MOVE to rearrange lines and use RENUMBER to clean up line numbers in the whole work file.

Similarly, beware of using "J" for JOIN; it means JUSTIFY in EDIT2! Instead, use the ADD *line number* FROM *filename* command to bring in text from a KEEP file.

EDIT2/3000 and 264x TERMINALS

1. Tabs are automatically set and cleared.
2. Tape cartridges are accessible using the special file names #TCL and #TCR.
3. Multiple files on each cartridge supported.
4. Terminal-connected printer is accessible using the special file name #PRINTER.
5. Line modification and wrap-around are shown "in-place" using selective erase, and selective re-display.
6. FIND uses inverse video to highlight position or string(s) found.
7. When AUTOMARGIN is off an inverse video "wall" is used to indicate the location of the right margin.
8. Error messages are erased after the user reads them; certain other messages are also erased.

A FEW WORDS ABOUT LINE NUMBERS

EDITOR/3000 uses the convention that TEXT and KEEP expect line numbers to be placed in the last 8 columns of each line. The ",UNN" option on these commands defeats this convention. The default value for SET LENGTH and SET RIGHT is 72, so numbered files will have 80 characters per record, as most compilers expect. When a file is TEXTed in, LENGTH and RIGHT are set according to the record size of the TEXT file. If the file is numbered, then RIGHT will be 8 characters less than the TEXT file record length.

Since EDIT2 is *document-oriented* rather than *compiler-oriented*, the default is that line numbers are located in the first few columns of each line, with a single space separating the number from the text. The default value for the right margin (RBOUND) is 65, which is useful for working in an 8½ by 11 inch format.

To use EDIT2 for source files, use the following form of the KEEP command:

KEEP *filename*,**REAR** *or* **K** *filename*,**R**

With RBOUND set at 72, this will put the line numbers at the rear of the file in a form which is recognizable to compilers.

There is also a "TEXT *filename*,REAR" option. If the ",REAR" option is *not* used, EDIT2 will check the text file and assume ",REAR" if the last 8 characters are all numeric. Use the ",UNNUMBERED" (or ",U") option to defeat this feature.

The ",REAR" options are useful mainly for program language source text editing, not for document editing.

APPENDIX B

How to get GALLEY and FORMAT documents into EDIT2 WORK files

GALLEY and FORMAT are both formatter programs that have been used to format documents prepared using EDIT/3000. Both use embedded commands exclusively to produce the desired output format. These commands, of course, are not executable by EDIT2; furthermore, it would be difficult to removed them and restructure the entire document manually.

The best approach to getting formatted documents over to EDIT2 is to run them through GALLEY or FORMAT for one last time. But, instead of letting the output go to a line printer, divert it to a permanent ASCII file. This file then contains the fully formatted document which can be copied into an EDIT2 work file using the TEXT command. After TEXTing the document, you will have to do some revision to remove the headings, footings, and blank lines between pages. It is also necessary to remove indentation on the first line of each paragraph to ensure proper line filling following future revision.

You can avoid the bulk of revision by making some changes in the formatting before you transfer the document to EDIT2. If you are still proficient with the commands for GALLEY or FORMATTER, you can check the following and make alterations as necessary.

GALLEY

- \DBL OFF
- \MARGIN 0 (default)
- Remove any \INLFT commands that have been used to indent the first line of a paragraph.
- Remove all \NEW, \NEED, and \SPACE commands to minimize the number of blank lines generated.
- Set \PAGELEN to a very large number (greater than the number of lines in the document). Be sure you have removed all commands that can trigger a new page!

FORMAT

- .BOTTOMSPACE Y=0,Z=0
- .HEADING OFF
- .MARGIN LEFT=1
- .TOPSPACE A=0,B=0
- Remove all .NEWPAGE, .ODDPAGE, .PAGE, and .SPACE >1 commands.
- Set .PAPER to a very large number (greater than the number of lines in document). Be sure you have removed all commands that can trigger a new page!

The procedure for diverting the output is to BUILD a permanent ASCII file, issue a FILE equation that diverts the output to the permaner file, and run the appropriate formatter program.

GALLEY PROCEDURE

1. :BUILD <filename>; REC=-80,1,F,ASCII; DISC=4096 (more or less)

2. :FILE GALLIST = <filename>, OLD; SAVE

3. :RUN GALLEY.PUB.SYS

4. :RUN EDIT2.PUB.SYS
   (Specify a new WORK file with enough lines to contain the document.)

5. >>TEXT <filename>

FORMAT PROCEDURE

1. :BUILD <filename>; REC=-80,1,F,ASCII; DISC=4096 (more or less)

2. :FILE OUT = <filename>, OLD; SAVE

3. :RUN FORMAT.PUB.SYS

4. :RUN EDIT2.PUB.SYS
   (Specify a new WORK file with enough lines to contain the document.)

5. >>TEXT <filename>