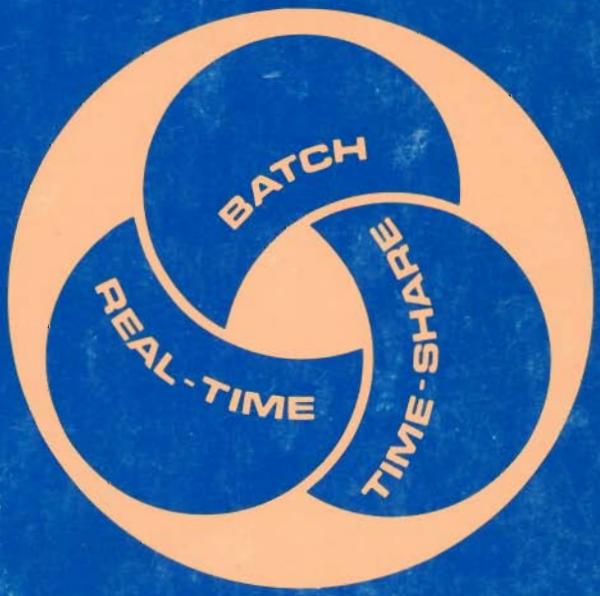
HP 3000

TEXT EDITOR



HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HP 3000

TEXT EDITOR

03000-90012A

November 1972

List of Effective Pages

Pages	Effective Date
Title	Nov. 1972
Copyright	Nov. 1972
iii	Nov. 1972
v to vii	Nov. 1972
1-1 to 1-4	Nov. 1972
2-1 to 2-5	Nov. 1972
3-1 to 3-37	Nov. 1972
4-1 to 4-5	Nov. 1972
A-1 to A-4	Nov. 1972
B-1 to B-4	Nov. 1972
C-1 to C-3	Nov. 1972
Index-1 to Index-3	Nov. 1972

Printing History

Part No.	Date	Update Package	Date
03000-90012A	Nov. 1972		

PREFACE

EDIT/3000 operates as a subsystem of the HP 3000 Multiprogramming Executive Operating System (MPE/3000) and is used to edit text in ASCII files. The user interacts with the subsystem through the EDIT/3000 command language, a language designed for text-editing that requires little or no user programming experience. For the more advanced programmer, or simply for an inquisitive user of EDIT/3000, the subsystem offers several expanded capabilities that allow for a wider scope of editing operations.

The information in this publication focuses on the syntax conventions and use of the command language. As an introduction to EDIT/3000, this manual describes the subsystem's structure and capabilities as well as its interaction with the operating system. The user is provided with instructions for operating the subsystem, including running in batch or interactive mode, entering commands and text, and handling errors. While mentioned on various occasions throughout the manual, the advanced features of EDIT/3000 are discussed more fully in a separate section.

EDIT/3000 is supplied as a User System Library tape.

CONTENTS

TABLES

SECTION I
Introduction

EDIT/3000 COMMAND LANGUAGE

EDIT/3000 is used to generate and modify text in ASCII files. With the EDIT/3000 command language, the user can create a file, insert, delete, or replace characters and lines in that file, move whole blocks of text from one place in the file to another, and combine several files to form one. The command language consists of a basic set of commands with which the user interacts with EDIT/3000 to create original text or to modify existing files. In addition to these basic commands, the advanced user has available a set of metacommands.

METACOMMANDS

The subsystem metacommands function like a special-purpose, interpreted programing language in that they give the user the ability to modify text selectively, contingent on the text itself. They are called *metacommands* because they can be programed to form a "sub-subsystem" of EDIT/3000 that controls execution of the basic commands. While the basic commands provide the less-experienced user of EDIT/3000 with a simple means of creating and modifying text, the advanced user will find the metacommands a powerful supplement to the basic command set.

EDIT/3000 SCRATCH FILES

During operation of the subsystem, the EDIT/3000 commands do not change the user's actual files. Rather, they use as working areas two scratch files: Text and Hold. The Text file is the actual working area of the subsystem. When EDIT/3000 is initialized, the Text file is empty. The user may either create new text in the Text file, or he may place a copy of an existing ASCII file in the Text file. He is then free to add to or change data in this scratch file. EDIT/3000 uses the Hold file for storing text that can be inserted into or appended to the Text file at a later time. Since both the Text and Hold files are treated as scratch areas, any text stored in them is lost when EDIT/3000 terminates, unless the text is saved in a library file.

CONTROL OPTIONS (THE SET COMMAND)

The configuration of EDIT/3000 is set up by default each time the subsystem is initialized. The user can control this configuration, however, by specifying options in the SET command. For example, by default, lines are numbered within the Text file starting with 1 and proceeding by increments of 1. If the user wants, he may override these defaults and tailor the numbering scheme to suit his own preference. For instance, he could specify that the first line in the Text file would be numbered 10 and that numbering would proceed by increments of 5. Other options in the SET command allow the user to set left and right margins in the Text file, shorten subsystem messages and commands, and control the location of line numbers in a user Text file.

DATA PROTECTION

Because the EDIT/3000 commands are capable of manipulating large sections of text, the subsystem has built into its command structure a data-protection feature that keeps the user posted on editing operations. This command structure allows him to stop operations generating unexpected results. For example, the DELETE command is used to remove designated portions of text. By default, the line number of each line affected totally or in part by a DELETE command will be printed, allowing the user to verify the operation or possibly stop it. The user has the option of eliminating this protective feature by adding a Q following the command, i.e., DELETEQ, or by specifying QUIET with the SET command. An operation can still be stopped in "QUIET" mode; however the user will be unaware of how far the operation progressed.

USER AID

If the user is unsure of the spelling or syntax of a command, a command is available (XPLAIN) that prints an explanation of any or all of the EDIT/3000 commands. In addition, if the user cannot recall the exact options he declared or defaulted to in the SET command, the VERIFY command, if entered, prints the values of any or all of those options.

MODES OF OPERATION

EDIT/3000 can operate in any one of three modes:

- Interactive mode, in which commands are entered one at a time from an on-line terminal
- Batch mode, in which all commands and input data are prepared in advance and entered through a batch reading device
- Use mode, which provides a mixed mode of operation that combines interactive and batch (Use mode is an advanced feature of EDIT/3000)

Unless otherwise noted, the interactive mode is presumed throughout this manual.

INTERACTION WITH MPE/3000

As a subsystem, EDIT/3000 does not have direct control over the computer and supporting equipment. However, when EDIT/3000 is operating, it uses the subsystem commands to generate actual system commands so that EDIT/3000 appears to have control over all required system facilities. Before the user can issue any edit commands, he must log onto MPE/3000. He can then call EDIT/3000 and proceed with editing operations. At the end of an interactive session or batch job, he returns control to the operating system.

ERRORS AND ERROR MESSAGES

EDIT/3000 recognizes two kinds of errors: syntactical and contextual. The effect of such errors depends on the mode of operation at the time of the error. In batch mode, syntax errors result in abnormal termination of the job. Syntax errors occurring during an interactive session result in an error message immediately following the error, allowing the user to correct the error and then continue. Context errors in any mode result in appropriate, informative messages. They do not cause termination of the operation but can be used to control further editing operations.

USING THIS MANUAL

To express the EDIT/3000 command language in a consistent manner, several standard syntax and format conventions are used:

1. The keyword elements of the command language and any options declared in the command are shown in uppercase characters.

2. Parameters entered by the user are in lowercase and italicized.

3. Optional fields are enclosed in brackets [ ].

   In the command

   KEEP [Q] *file name* [(*range*)] [,UNNUMBERED]

   Q, *range*, and UNNUMBERED may or may not be entered, but the user *must* replace *file name* with an appropriate entry.

4. Where one of several options is to be selected, the options are shown in braces { } , and if there is a default option, it is underlined. For example, in the command

   SET [, { QUIET / <u>DISPLAY</u> }]

   the user may select either QUIET or DISPLAY, but DISPLAY is the default if the user specifies neither.

5. A triangle Δ indicates one space, or a blank, in the text.

6. *cr* indicates a carriage return at an interactive terminal.

7. In examples in this manual, subsystem responses (if in interactive mode) are underlined; user input is not. For example, following is an example of the ADD command:

/ADD 50.1 *cr*

50.1 LINE 50.1 IS INSERTED BETWEEN LINES 50 AND 51.

EDIT/3000 prompts the user with a slash (/). The user can then enter his command, ADD 50.1, and a carriage return (*cr*). The subsystem responds with line number 50.1, and the user then enters his text into that line.

SECTION II
Operating Procedures for EDIT/3000

LOGGING ON TO THE OPERATING SYSTEM

To use the EDIT/3000 subsystem, the user must first log on to MPE/3000. System commands for logging on, calling EDIT/3000, returning to the system after a break, and logging off in either mode are mentioned briefly in this section but are described completely in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*

CALLING EDIT/3000 IN INTERACTIVE MODE

After the user logs on to the operating system in interactive mode, MPE/3000 prints ":" (the prompt character). To call EDIT/3000, the user responds by typing

:EDITOR [*listfile*]

(Note that ":" is printed by MPE/3000 and is not typed in by the user.)

listfile allows the user to have offline listings sent to an output file of his choice. The default destination for offline listings is the system line printer. (The system command, :FILE may also be used to set the *listfile*.)

When EDIT/3000 is initialized, the prompt character, a single slash /, is printed, which indicates that the program is ready to accept input of EDIT/3000 commands.

ENDING AN INTERACTIVE SESSION

To terminate the session, the user enters an END command from the terminal. To prevent accidental loss of the contents of the Text and Hold files, EDIT/3000 prints a message requesting verification of the intent to terminate before actually terminating. The user can answer YES, Y, NO, or N. When the subsystem receives an affirmative response, the two scratch files (Text and Hold) are cleared, the session is ended, and control is returned to the operating system. If the user enters a negative response, EDIT/3000 continues executing until another END command is encountered.

The END command, used with a matching BEGIN command, does not terminate execution of the subsystem. This use of the command is explained in Section IV, "Advanced Features."

To log off of the operating system, type BYE after MPE/3000 prints ":" (the prompt character).

USING EDIT/3000 IN BATCH MODE

In batch mode, the command

:EDITOR [*listfile*]

should follow MPE/3000 log-on commands and precede any EDIT/3000 commands and object records in the job.

When EDIT/3000 is initialized in batch mode, all commands and object records are read from a batch reading device. Each EDIT/3000 command submitted to the batch device must be preceded by a slash, the EDIT/3000 prompt character. A double slash entered as the first two characters of a line of input is interpreted as an end-of-data marker and is used to inform the subsystem that it can now expect to read an EDIT/3000 command on the succeeding line.

Output includes all commands and all object records entered through the batch reading device and interpreted by EDIT/3000. Syntax errors cause job termination and a message indicating the command that caused the error.

The END command (provided it is not matched with a BEGIN command) terminates execution of EDIT/3000 and returns control to the operating system. An :EOJ statement terminates the batch job.

USE MODE

To operate in use mode, the user instructs the subsystem (with the USE command) to go to a user file and execute the commands it finds there. At the same time (if the subsystem is initially in interactive mode when the USE command is executed), EDIT/3000 expects object records to be entered from the terminal and issues all messages to the terminal. Thus, the subsystem is operating in both an interactive and batch mode simultaneously. The USE command can also be executed during a batch job with similar results, except that object records are read from the batch reading device as part of the batch job.

Use mode, intended primarily for advanced users who are writing sub-subsystems with the metacommands, is explained in greater detail under "Advanced Features," Section IV.

USING THE EDIT/3000 SCRATCH FILES

Lines of text are edited by EDIT/3000 commands after they are placed in the Text file. An ASCII file that already exists in the system library can be retrieved from the library by naming the file in the Text command. A copy of the file is then placed in the Text file, and any of the subsystem commands can be used to edit the Text file, while the original version (still in the library) remains unchanged.

Lines of copy can be entered into the Text file from the terminal keyboard after specifying the ADD command, or they can be copied from the Hold file to the Text file using the ADD, INSERT, or REPLACE command with the HOLD option.

Saving Edited Files

Since the Text and Hold files are actually scratch areas, not permanent files, their contents are destroyed at the end of a terminal session or batch job. The KEEP command saves the contents of the Text file in a user file.

Size of the Text File

At the beginning of a session, if the ADD command is used to place text in the Text file, the Text file will be created by EDIT/3000 to hold approximately 4000 records, by default. Also by default, if the TEXT command is used to copy text from a user file into the Text file, the Text file will be made 50 percent larger than the user file. If space in the Text file is subsequently filled, a message will be issued, indicating that the Text file is filled. When the user requires more space, he saves the Text file using the KEEP command, and then redeclares it with the TEXT command. This results in a Text file that is 50 percent larger than during the previous session.

An option in the SET command (SET SIZE=) allows the user to specify a Text file size other than the default. For example, by declaring SET SIZE=7000, a Text file capable of holding 7000 records is obtained.

LINE NUMBERING WITHIN THE SCRATCH FILES

If the Text file is empty, the first line of text entered with the ADD command is automatically numbered 1, and each subsequent line in increments of 1, unless the user has specified otherwise (using the SET command option).

To allow for the addition and insertion of lines between two adjacent lines, a line number is of the general form

nnnn.nnn

Leading zeros, and trailing zeros following the decimal point, can be omitted. If nothing follows the decimal point, it can also be omitted.

When the contents of the Text file are saved in a user file (with the KEEP command), the line numbers are written in ASCII format into the first or last eight bytes of each line (depending on the SET command option taken), unless the UNNUMBERED option of the KEEP command is specified. If that file is subsequently called back to the Text file, it will be numbered according to those saved line numbers.

Line numbers are kept and retrieved in the same format as used by the HP 3000 language compilers. This establishes compatibility between EDIT/3000 and the editing features of the compilers. Thus, although line number one in the Text file is displayed to the user as 1, it is written to the file system as #00001000 when the file is saved with the KEEP command.

MPE/3000 SPECIAL CONTROL KEYS

To escape from or end an operation, or to erase lines or characters entered inadvertently during an edit operation, the following control keys are available for use on most terminals.

CTRL H (hold control key down and type H) deletes the preceding character. A reverse slash (\) is printed. CTRL H can be typed more than once; each time it is typed, one character is deleted, and one reverse slash is printed.

CTRL X (hold control key down and type X) deletes the line just typed, before the return key has been typed. The system response is a triple exclamation point (!!!) followed by a carriage return and line-feed.

CTRL Y (hold control key down and type Y) terminates an edit operation. This control key can be used to break off an edit operation that is taking too long.

Table 2-1. Special Characters

Character	Meaning
:	MPE/3000 system prompt character
/	EDIT/3000 prompt character
/ /	End of object records; the / / notation is a signal to EDIT/3000 that the subsystem should now expect to find edit commands. (Used in batch mode only.)
*	When used in a range expression in an edit command, an asterisk indicates the current position of the pointer.
^	When the subsystem lists text, a caret indicates the current position of the pointer.
CTRL Y	Indicates the end of an operation.
CTRL X	Erases line entered, prints !!!, and returns the carriage.
CTRL H	Erases the previously typed character, and prints a single reverse slash (\).
&	Indicates that a command or input line is continued on the next line. If used, it is replaced with a blank.
;	Separates commands when more than one appears in the same line.

SECTION III
EDIT/3000 Command Language

EDIT/3000 commands use range parameters to indicate the part of the text to be edited. These parameters define a line or a character in a line in the Text file, or they specify a range that extends over a number of lines or columns. Any command specifying a range refers to a position or pair of positions in the Text file. A position may refer to a line of the Text file, to a location within a line, or to a specified character string. Some commands permit specification of a list containing several ranges separated by commas.

A "pointer" in the Text file is kept by the subsystem to indicate the position in the file where the last editing operation occurred. At the beginning of a session, or if the Text file is empty, the current position of the pointer is the first column of the dummy line zero. As edit operations are executed, the pointer is moved through the file. When text is displayed (for example, during a FIND operation), the pointer appears below the line of text as a circumflex ^ . An asterisk * is used to specify the position of the pointer in a range or position expression.

SYNTAX OF RANGE EXPRESSIONS

The formal syntax of range expressions is described in Table 3-1. Following this table are definitions of terms used in these expressions.

Table 3-1. Formal Syntax of Range Expressions

Expression	Syntax
Range list	{ range range, range list }
Range	{ position null position/position ALL }
Position	Record position [(column position)]
Record position	Record identifier [(relative record position)]
Record identifier	{ line number string * FIRST LAST }
Line number	Unsigned integer that may contain a decimal point
String	One or more characters or a null string delimited by any special character except , ; . ( ) * / + -
Relative record position	Signed integer (±)
Column position	{ (relative character position) (absolute column position) }
Relative character position	Signed integer (±)
Absolute column position	(column specifier [absolute column adjustment])
Column specifier	{ unsigned integer LEFT RIGHT LAST * }
Absolute column adjustment	Signed integer (±)

Position

A position describes the location in the Text file of

- A line
- A character or column in a line
- A delimited string
- The Text file pointer

A position can be expressed as being either absolute or relative to another position in the Text file. If a position expression specifies a line or string only, the first column position of the line or string is assumed with one exception: if the position is the second position in a range expression, the last column of the line or string is assumed.

Range

A range expression in a command informs EDIT/3000 which part of the Text file is affected by the command. A range expression can consist of only one position or it can consist of all characters between and including two positions.

If two positions are declared as a range, the first position must sequentially precede the second position. A slash separates the two positions. A range list consists of several ranges, separated by commas. For example, 7/10, 50, 100/200 is a range that includes lines 7 through 10, line 50, and lines 100 through 200. A null range in a range list is indicated by two commas.

If ALL is specified as the range in a command, the range includes all text currently in the Text file.

When two positions are specified in a range and the column or character position has been omitted, the column position of the first position specified begins the range; the last column position of the second position ends the range. For example, in the range 70/"STRING" the range begins with the first column of line 70, ends with the G in STRING, and includes all text in between.

Where ranges refer to line numbers, it is not necessary that the lines with the numbers actually exist for the range to be valid. When a range consists of a single line, however, that line must exist.

Line

line refers to a line in the Text file. Each line has a unique number that indicates its position. A line number is of the form nnnn.nnn. For example, 1000.001.

Characters and Columns

Each line of the Text file has a predefined number of columns as defined either by the SET command or by the system default, which is dependent on the type of terminal used. On the most common terminal, a teleprinter, each line contains 72 columns. A column position in a range expression specifies a particular column in a line of the Text file. The first column position is 1, unless altered by the SET command. The last column position in a 72 column line is 72, unless altered by the SET command.

A character position is always relative to the left column position of the record identifier. A character position is found by counting nonblank characters from the position. For example, say the left column position of line 31 is 1, and the first nonblank character in that line is in column 10. If a position expression specifies the first *character* in that line, the range of the command is column 10 of line 31.

Specifying Absolute Record Position

An unsigned decimal number *not in parentheses* refers to the absolute position of a single line. For example, 7.85 refers to line number 7.85 in the Text file. A pair of such numbers separated by a slash refers to those two lines and any lines between them. Thus 7.85/10 refers to all columns between and including the first column in line 7.85 and the last column in line 10.

An unsigned decimal number, LAST, LEFT, RIGHT, or * *in parentheses* refer to a column position in a line. For example, 7.85(9) refers to the ninth column position in line number 7.85. The maximum column position of a line is a function of the line length of the terminal and the value defined by default or in the SET command. Specifying LAST refers to the last non-blank character in the line. Specifying LEFT or RIGHT refers to the first or last column position, respectively, in the line, as determined by the SET command options. For example, (LEFT+5) refers to the fifth column to the right of column one. If LEFT is set to 35, (LEFT+5) specifies column 40. (RIGHT-5), if RIGHT is set to 72, specifies column 67. Absolute column adjustments are confined to one line, unlike a relative character adjustment, which may span several lines. If a column position is not specified, the left column in the line is assumed, except where the position is the second position in a range expression, in which case LAST is the default column position.

Specifying Relative Record Position

Position may be specified relative to a line or string position, or to the current position of the pointer. A relative record position is expressed as a signed integer following a line number, a string argument, or an asterisk. The position 155.7+6 specifies is the sixth *line* following the line numbered 155.7, taking into consideration any lines that have been added or inserted. On the other hand, the expression 155.7(+9) indicates the column position of the ninth nonblank *character* following the left column position of line 155.7. "program" (+9) specifies the column position of the ninth nonblank *character* following the "p" in the next occurrence of the string argument "program." And *+9 specifies the ninth *line* following the current line position of the pointer.

String

string refers to a string of characters enclosed within a pair of quotes or any other special characters except those included in the command language syntax (namely , ; . ( ) * / + -). A null string is indicated by two consecutive delimiters (e.g., - -). When referenced in a position or range expression, the search for the specified string proceeds from the present position of the pointer. Series of strings that are to be concatenated are allowed; for example, "ABC""DEF""GHI". Nonprinting characters can appear outside the string, if represented by their numeric equivalent and preceded by apostrophes ('). Thus the string "EDIT/3000""13' 10 represents the characters EDIT/3000, a carriage return (13) and linefeed (10). The decimal representations of ASCII characters appear in Appendix C.

DECLARING PARAMETERS IN EDIT/3000 COMMANDS

The user declares the range of a command according to the syntax of the particular command. For example, the ADD command operates only on line numbers; the REPLACE command can operate on an entire range list. The syntax of the TEXT, USE, and KEEP commands requires the name of a user file. No parameters are entered for the END command. The CHANGE command allows the user to specify a unique range called a column position range. Such a range refers to a single column position in a line, or to a pair of column positions, a range which may span several lines. The legal range expression for each command is defined in the description of the commands (this section) and also in Table B-1 in Appendix B.

Several commands can be declared in one line if they are separated by semicolons (;). For example,

```
SET FIRST = 50, LAST = 132; VERIFY ALL; ADD
```

To continue a line of commands (or a string within a command), enter an ampersand (&) as the last character in the line. For example,

```
SET FIRST = 50, LAST = 132; VERIFY ALL; Q "THIS LINE IS&
CONTINUED ON THE NEXT LINE"
```

The ampersand is replaced with a blank.

Table 3-2. Range Expressions

Expression	Refers to
100	Line number 100; the pointer indicates the left column.
35.2	Line number 35.2; the pointer indicates the left column.
100+10	Left column position of the tenth line following line number 100.
35(2)	Second column position of line 35.
35(+2)	Column position of the second nonblank character following the left column position of line number 35.
100/110	All characters between and including the left column in line 100 and the right column in line 110, including all columns in the lines between.
100/110,150/160	A range list specifying the same range as in the preceding expression, plus the range 150 to 160; any lines between 110 and 150 would not be included.
100(10)/110(10)	All characters between and including the tenth column position of line 100 and the tenth column position of line 110.
*	The current position of the file pointer.
*+2	Left column position of the second *line* after the line containing the file pointer.
*-2	Left column position of the line that is two lines before the line containing the file pointer.
*(+2)	Column position of the second nonblank character following the file pointer.
"HP/3000"	The next occurrence of the string "HP/3000" following the current position of the file pointer. The pointer is moved through the Text file until it encounters the string HP/3000 or until it encounters the end of the file. If the string is found, the pointer comes to a rest at the first character of the string (in this case, the "H" in "HP"). If the end of the file is found before the string is located, a message is issued.
/(LAST)	The range from the current pointer location to the last nonblank character in the same line.
100/"THE PROGRAM"	The range extending from the left column position in line 100 to the "M" in the first following occurrence of the string "THE PROGRAM." If THE and PROGRAM happened to fall on successive lines, the string would still be found (the transition between lines is equivalent to one blank).
/+5	Range from the current pointer location to the last character of the fifth *line* after the line in which the pointer is located.
/(+5)	Range from the current pointer location to the position of the fifth nonblank *character* following that location.

ADD

The ADD command is used to enter lines of text into the Text file. After typing in the ADD command, the user can enter text from the terminal keyboard, from the Hold file, or from a combination of both sources.

Form:

ADD[Q] [*line number*] [,HOLD[Q] [,NOW]]

If neither Q, *line number*, nor HOLD is specified, EDIT/3000 responds with a line number and waits for input from the keyboard. If the Text file is empty, and the default options for line numbering are in effect, the first line number will be 1. The user may then type in the text he wishes added to the Text file from the terminal keyboard, terminating each line with *cr*. EDIT/3000 generates a new line number following each *cr*, continuing until the escape option is entered. To stop entering lines, the user types CTRL Y.

If the user specifies *line number* in the ADD command, the subsystem responds with that number, and the user can then enter lines of text.

To add the contents of the Hold file to the Text file, the user specifies HOLD in the ADD command. When the subsystem responds with a line number, the user can type CTRL Y, and the entire contents of the Hold file will be added to the Text file. Or, the user can enter text from the keyboard, then type CTRL Y, with the result that the contents of the Hold file will be added to the text just entered, beginning on the next line. Lines from the Hold file will be numbered as if they were entered from the keyboard and will be listed at the terminal, unless HOLDQ was specified. No listing will be produced if the user declares HOLDQ. If NOW is specified, EDIT/3000 automatically adds the text from the Hold file, without waiting for input from the user or the CTRL Y.

If the user specifies an ADD command that contains a line number duplicating an existing line number in the Text file, EDIT/3000 rejects the command and issues a message. (The REPLACE command replaces lines.)

Because of the line-number structure of the Text file, it is possible to add lines between two adjacent lines even when the interval between line numbers is 1 (see "Line Numbering Within the Scratch Files," Section 2). For example, if the command ADD 5.1 is used, EDIT/3000 enters line 5.1 between the existing lines 5 and 6 of the Text file. Subsequent lines are numbered in increments of .1 instead of 1, until line 5.9 is entered, at which point the increment between lines is decreased automatically by a factor of ten until a minimum increment value of .001 is reached. (ADD 5.10 results in an initial increment of .01.)

One or more lines can be added before the first line in the Text file by specifying a decimal fraction less than 1, such as .1, .01, .001, if the first line is numbered 1.

If the ADD command takes the form ADDQ, the subsystem will not respond with a line number. Rather, the user—unprompted by EDIT/3000— may type in as much as a full 72-character line (or more on a terminal with a longer line-length than a teleprinter). Text added in this manner is still numbered within the Text file, but the user must obtain a listing to refer to specific lines of text by number.

The ADD command operates on complete lines of text; the INSERT and MODIFY commands are used to manipulate characters within lines.

Examples

```
/ADD cr
   1    THIS IS THE FIRST LINE ENTERED IN THE TEXT FILE.
   2    Note: The user continues entering lines or enters CTRL Y to terminate
             the command.

/ADD 50.1 cr
  50.1   LINE 50.1 IS INSERTED BETWEEN LINES 50 AND 51.
  50.2   LINE NUMBERS WILL BE INCREMENTED BY .1
  50.3   UNTIL LINE 50.9 IS ENTERED.
    .
    .
    .
  50.9   THE NEXT LINE ENTERED WILL BE
  50.91  LINE 50.91
  50.92  CTRL Y
/

/ADD 60, HOLD cr
   60    THE FOLLOWING LINES OF TEXT ARE FROM THE HOLD
   61    FILE
   62    CTRL Y
   62    THIS IS THE FIRST LINE OF THE HOLD FILE.
    .
    .
    .

/ADD 110 cr
**15:**COMMAND WON'T REPLACE OR INTERLEAVE LINES
             Note: Line 110 already exists. (The user presses any character to obtain
                   the complete error message).


/ ADD,HOLD,NOW
    1    THIS IS THE FIRST LINE OF THE HOLD FILE
```

CHANGE

The CHANGE command causes an existing character string or column position range to be changed to a new character string within a specified range.

Form:

CHANGE [Q][{ *absolute column position*[/*absolute column position*] / *character string* }]TO *character string*
[IN *range list*]

Note: Commas can be substituted for TO and IN.

The *absolute column position/absolute column position* range is used only in the CHANGE command and sets column limits within a line between which the new character string is inserted in place of the existing characters. The first column position must precede the second or the operation will fail. If a single column position is specified, the new string is inserted in front of that position.

Character strings must be enclosed by delimiters: in "Examples," below delimiters are shown as quotes but may be any special character except the following:

, . ; ( ) * + -

range list consists of positions or pairs of positions. When a range is only one position, the entire line containing that position is affected. If more than one range is specified, it is separated from the next range by commas as in 25/40, 70/100. If IN *range list* is omitted or there are any null ranges in the *range list*, the change will be made only if the old string is found at the present position of the pointer. Where the first string in the command is a null string, the second string will be inserted only once, before the first position in each range or before the pointer for a null range.

When the CHANGE command specifies an existing string, EDIT/3000 searches all lines within the range or ranges stated in the command and replaces all occurences of the old string with the new string. At the termination of this command, the file pointer is left resting at the end of the last range in the *range list.*

Every time the user changes a line of text with a CHANGE command, the subsystem prints that line in its altered form. If the user is not satisfied with the change, he may end the operation with CTRL Y. Otherwise he can continue changing text until the range list is processed. To eliminate the listing of the changed text, the user can specify CHANGEQ rather than CHANGE when he enters the command.

If a change to a line contains a replacement longer than the string to be replaced and the result is a line with nonblank characters extending beyond the column specified with SET LENGTH= (default is 72), the subsystem sends a warning message. However, the subsystem will not make changes that would increase the length of a line more than 50 percent above the original maximum length of the lines in a file. Instead, the CHANGE operation will be halted, and a message will be sent. In addition, a CHANGE operation fails when the absolute column position range extends outside of the margins set by the SET command (SET LEFT=, RIGHT=).

Example

/CHANGE "RECORD" TO "LINE" IN 1/5, 7/9, 12 *cr*

The word "RECORD" is changed to "LINE" each time it occurs in lines 1 through 5, 7 through 9, or in line 12.

/CHANGE "THE" TO "A" IN 100/500 *cr*

If line 100 of the Text file reads THEN THEY WENT TO THE THEATER, this command would result in AN AY WENT TO A AATER. Surrounding both strings with spaces in the command, i.e., CHANGE "ΔTHEΔ" TO "ΔAΔ" in 100/500, would result in THEN THEY WENT TO A THEATER,

/CHANGE 3/10 TO "NEVER" IN 51/53 *cr*

This command deletes the characters in the third to the tenth column positions (inclusive) of lines 51, 52, and 53, and inserts the string "NEVER" in their place.

/ CHANGE 3/3 TO "A" IN 50 *cr*

The character in column 3 of line 50 is deleted and replaced by A.

/CHANGE 3 to "A" IN 50 *cr*

The character "A" is inserted before column 3 in line 50. The content of column 3 is unchanged.

DELETE

The user can remove characters and lines from the Text file using the DELETE command.

Form:

DELETE[Q] [*range list*]

If no *range list* is specified, the subsystem deletes the line in which the pointer is currently resting. When a range is a single position, the entire line containing that position is deleted. A DELETE command that specifies a pair of positions results in the deletion of all characters between the two positions, including any lines that fall totally between those limits. Deleted portions of the Text file are lost.

The line number of any line that is affected totally or in part by a DELETE command is listed on the terminal, allowing the user to stop the operation with an CTRL Y if it appears that too much is being deleted. Line numbers are not printed if the user enters DELETEQ as the keyword. The user specifies DELETEQ if he feels he does not need the protection or that listing would take unnecessary time.

Example

/DELETE *cr*

This command deletes the entire line in which the file pointer is currently resting.

/DELETE 60(10) *cr*

This command deletes all of line 60.

/DELETE 60(10)/70(5) *cr*

The character in the tenth column position of line 60 and all characters to the right in that line, any lines between line 60 and line 70, and the characters in the first five columns of line 70 are deleted.

/DELETE 50/75, 100/110, 125/130, 150(3)/155(7) *cr*

Lines 50 and 75 and all lines between are removed; any lines between 75 and 100 are unaffected; 100 through 110 are removed; 125 through 130 are removed. The character in the third column position in line 150 and the remainder of line 150, any lines between 150 and 155, and characters in the first seven column positions of line 155 are removed.

The next example assumes the following contents of the Text file:

```
10    SINCE LANGUAGE IS THE ONLY REAL EVIDENCE OF CONSCIOUSNESS
11    IN HUMAN BEINGS, IT WOULD SEEM FOOLISH TO DIMINISH
12    ITS IMPORTANCE. WORDS ARE MORE THAN MEANING, MORE
13    THAN STRUCTURES. THEIR USE DEFINES HUMAN BEHAVIOR
14
 .
 .
 .
_/ DELETE "SINCE△"/"SINCE△" cr
```

This deletes the first word in line 10, assuming the pointer location preceded this word. The line is left-justified.

```
_/ DELETE "IT" / "IMPORTANCE.△" cr
```

Also assuming the pointer location is before the first string ("IT"), this command deletes the word "IT" and all characters following in that line. The characters "ITS IMPORTANCE." are deleted from the next line, which is left-justified. (If another line had existed between line 11 and 12, that line would also be completely deleted.)

The Text file now looks like this:

```
10    LANGUAGE IS THE ONLY REAL EVIDENCE OF CONSCIOUSNESS
11    IN HUMAN BEINGS,
12    WORDS ARE MORE THAN MEANING, MORE
13    THAN STRUCTURES.  THEIR USE DEFINES HUMAN BEHAVIOR
14
 .
 .
 .
```

If the location of the pointer is unknown or if it follows the text to be deleted, the FIND command can be used to position it so that it precedes that part of the text that the user wishes to delete.

END

The END command terminates execution of EDIT/3000 and returns control to the operating system.

Form:

```
END
```

When the END command is used without a matching BEGIN command, EDIT/3000 prints a message asking for confirmation before terminating. This permits the scratch files to be saved with the KEEP command.

For a discussion of how the END command is used as a metacommand, see Section IV, "Advanced Features."

FIND

The FIND command moves the pointer to a specified location in the Text file.

Form:

FIND [Q] *range*

When *range* consists of a single position, the pointer is moved to that position. If the specification is a line number, the pointer is set to the first column in the line; if it contains a column position, the pointer is moved to the column specified. If a string is specified, the pointer is moved from its location at the time the FIND command is executed to the column position of the first character of the next occurrence of the string.

In a range consisting of two positions, the first position is found only if it is not preceded by the second position. Thus the command FIND 800/600 is invalid and causes the operation to end without moving the pointer. A search for a string, however, is not limited by the second position unless the second position is a position relative to the location of the pointer. For example, FIND "THIRTY"/800 would cause EDIT/3000 to search from the pointer to the end of the Text file to find both positions, which could take considerable time. To avoid a lengthy or unnecessary search, the command could be issued as FIND "THIRTY"/*+10, assuming that the string is valid only if it occurs within the next 10 lines.

A search for a string operates only on text within and inclusive of the left and right margin options declared in the SET command. In other words, if the margins are set with the command SET LEFT=50,RIGHT=72 (see SET command), the FIND command will not locate a string if it occurs totally or in part in columns 1 to 49 or 73 to the last column position in a line.

Unless the QUIET option has been declared in the SET command or the FINDQ form of the command has been entered, the line containing the pointer is printed at the terminal if the FIND operation is successful. A circumflex (^) on the line below indicates the precise location of the pointer.

The FINDQ form of the command or the SET QUIET option results in no listing of found text positions.

A message will be printed if the subsystem cannot comply with the FIND command. Reasons for failure include specifying a location or string that does not exist, specifying a range in the wrong order (last position found before first position), and specifying a range outside the left or right margins of the Text file.

Example

/ FIND * *cr*

Finds the file pointer and prints the line, indicating the pointer location (unless SET QUIET was specified).

```
73      THIS IS THE 73RD LINE OF THE TEXT FILE.
        ^
```

Assuming that the file pointer is at the beginning of line 73, as shown above.

```
_/ FIND "73RD" cr
```

Positions the pointer under the digit 7 in "73RD":

```
73      THIS IS THE 73RD LINE OF THE TEXT FILE.
                    ^
_/ FIND "THIS"/*+4 cr
```

The next occurrence of "THIS" is found if it is within the next four lines.

GATHER

The GATHER command moves specified lines from one location in the Text file to another, renumbering lines as specified and deleting the lines from their original location.

Form:

GATHER [Q] *range* TO *line number* [BY *increment*]

Note: TO and BY can be replaced by commas.

All lines within *range* are moved to a point in the Text file starting at *line number.* Lines are deleted from their original location in the text and renumbered starting with the number specified in *line number.*

If BY *increment* is not specified, the increment between lines is 1, unless another numbering option was specified in a SET command. If BY *increment* is omitted and *line number* specifies a location between existing line numbers, EDIT/3000 decreases the increment by a factor of 10 until the minimum increment (.001) is reached in order to fit the lines moved into the space available between existing lines. If the range does not fit even though the increment has been decreased as far as possible, the operation will halt, a message will be printed, and no text will be moved.

An integer specified in the BY increment option overrides any other numbering options or defaults. The subsystem will not decrease the increment in the event that the range will not fit but will halt the GATHER operation with no text moved.

Lines not included in the range are not moved and remain in their original location in the Text file.

EDIT/3000 calculates whether or not the range will fit relative to the line that follows the line number specified in TO *line number*, even though that line may be within the range to be moved. This does not apply if the range specified is ALL, in which case, the entire Text file is renumbered (and a fresh line directory is prepared). When renumbering a section of text where new numbers will be overlapping old, the following alternative command sequence should be used:

HOLD *range*
DELETE *range*
ADD *line number*, HOLD

Unless the GATHERQ form of the command is stated, each line number change is listed. The user may stop the GATHER operation at any point by typing CRTL Y.

Examples

```
_/ GATHER 50/55 TO 5.1 BY .1 cr
```

Assuming that lines are numbered in increments of 1, lines between 50 and 55 are inserted at a point between line 5 and line 6 and renumbered 5.1, 5.2, etc.

```
_/ GATHER 50/55, 5.1, .1 cr
```

Accomplishes the same result.

The listing produced by these commands would be

```
50  :=  5.1
51  :=  5.2
52  :=  5.3
53  :=  5.4
54  :=  5.5
55  :=  5.6
```

HOLD

The HOLD command copies lines from the Text file to the Hold file.

Form:

HOLD [Q] [*range*] [,APPEND]

All lines or portions of lines within *range* are copied from the Text file to the Hold file, and a listing is produced. Lines in the Text file are unaffected. If APPEND is not specified, the Hold file is cleared and the lines specified in *range* replace the previous contents of the Hold file. If APPEND is specified, the lines specified in *range* are appended to the Hold file and will follow any existing contents of the Hold file. If just Hold is specified, without *range* or APPEND, the Hold file is cleared.

If the HOLD command is stated with the Q option, a listing of the text placed in the Hold file will not be produced.

Examples

HOLD 100/150 *cr*

Clears the Hold file and copies lines 100 through 150 of the Text file into the Hold file.

HOLD 100/150 APPEND *cr*

Copies lines 100 through 150 of the Text file into the Hold file following any existing lines in the Hold file.

HOLD *cr*

Clears the Hold file.

INSERT

The INSERT command inserts characters or lines into the Text file at a specified position.

Form:

INSERT [Q] *position* [BY *increment*] [,HOLD[Q] [,NOW]]

EDIT/3000 prints the line containing the specified *position*, then indicates the position of the file pointer by printing a circumflex (^) below it, on the next line. Characters typed following the pointer are inserted in the line in front of the pointer. The remainder of the line is saved and printed following the inserted material (or following lines from the Hold file if HOLD is specified). When more than one line is inserted, EDIT/3000 generates a line number each time *cr* is pressed in the same manner that lines are numbered for ADD unless the INSERTQ form is entered. If INSERTQ is declared, the subsystem will not prompt the user with the line number. To terminate the insertion, type CTRL Y.

The BY *increment* option allows the user to temporarily alter the line numbering increment to provide for large insertions.

If HOLD is included in the command, when CTRL Y is typed the entire contents of the Hold file is copied into the Text file following those insertions already made from the keyboard and preceding the remainder of the line specified in the INSERT command. A listing of the Hold file contents is printed at the terminal after the insertion is made unless HOLDQ is specified in which case, this list will not appear. If NOW is specified, EDIT/3000 automatically adds the text from the Hold file, without waiting for input from the user or the CTRL Y.

It may happen that after CTRL Y is typed, the end of a line (as determined by SET RIGHT=) is reached while the remainder of the line is being copied or while the Hold file is being inserted. In such cases, EDIT/3000 will break that line on a space as close as possible to the end of the line and start on a new line. Thus the command may be used to roughly align the right margin of sections of text after editing.

Example

/ INSERT 25(4) *cr*

Prints line 25 of the Text file with a circumflex directly under the fourth column position in that line to indicate the file pointer. Any characters typed following the circumflex are inserted in the line in front of that point. An additional line number is generated each time *cr* is pressed. Use CTRL Y to terminate INSERT.

/ INSERT 25(4), HOLD *cr*

Operates as above, except that the entire contents of the Hold file are inserted following any insertion from the keyboard when CTRL Y is pressed, and before the remainder of line 25 is inserted. Entering CTRL Y without any copy from the keyboard inserts just the Hold file.

JOIN

The JOIN command adds a complete file to the Text file.

Form:

JOIN [Q] *file name* [{(line number/line number) / (# integer/# integer)}] [TO *line number*] [BY *increment*]

Note: TO and BY may be replaced by commas.

The JOIN command functions like the GATHER command, except that lines from a different file are used rather than lines from the Text file. If the (*line number/line number*) range option follows *file name*, only those lines in *file name* within and including the declared line numbers are joined. If (# integer/# integer) is specified, actual sequence numbers are used as the range. Otherwise, the entire file is joined to the Text file.

When *line number* and *increment* are not stated, the contents of file name are appended to the Text file following the last line using 1 as an increment between lines, unless a different numbering increment has been declared in a SET command.

If *line number* is stated,the operation will start at that line number using either the default or a declared line-numbering increment. As in the GATHER command, EDIT/3000 adjusts the increment to make the insertion fit. If both *line number* and *increment* are stated, the increment called for becomes the temporary increment value, and no adjustment is attempted. In either case, if the file cannot be inserted between existing lines in the Text file without duplicating an existing line number in the Text file, the command fails, and no lines are moved.

As the contents of the file named in the JOIN command are entered into the Text file, each line is listed, with its assigned line number, at the terminal.

This listing is eliminated if the JOINQ form of the command is declared.

Examples

/ JOINQ EDIT02 TO 1000 BY .1 *cr*

The contents of the user file EDIT02 will be inserted into the Text file beginning with line number 1000, continuing with line number 1000.1, 1000.2,...1000.9, 1001.1, etc. The increment value will not decrease by a factor of 10 if the insertion does not all fit, nor will the text of the file be listed as it is inserted.

/ JOIN LIB10 (1000/5000) *cr*

Lines 1000 through 5000 in the file named LIB10 are appended to the end of the Text file.

Note: If the file to be JOINed has been equated to a device (such as a remote card or tape reader) using the MPE/3000 :FILE command, the number of records in the file will not be available in advance. In this situation, the user should use a Utility program to copy the file to disc before entering the JOIN command.

KEEP

The KEEP command saves the contents of the Text file.

Form:

KEEP [Q] *file name* [(range)] [,UNNUMBERED]

If *file name* refers to a new file, a file is opened in the library under that name, and the current contents of the Text file are stored there. When *range* is specified in the command, only those lines of the Text file described by *range* are stored in the user file. (If a column or character position is declared in *range*, the entire line containing that position is saved.)

In interactive mode, if an existing file is named in the command, EDIT/3000 asks the user if he wishes to purge the old contents of the file. A positive response causes the contents of the old file to be purged and the contents of the Text file to be written into the user file. The KEEP operation terminates if a negative response is received.

In batch mode, the contents of an old file are automatically purged when the file is named in a KEEP command.

Records in the Text file are saved as variable length records unless the SET FIXED operation in the SET command was taken, in which case they are saved as fixed length records.

A system error message is issued if access to a file is not authorized. See *Multiprogramming Executive Operating System* for information on file creation and access authorization.

Unless the UNNUMBERED option has been declared, the line numbers in the Text file are placed in the last 8 bytes of the lines in ASCII form unless the SET FRONT option was declared in the SET command. Line numbers in the Text file are lost when the UNNUMBERED option of the command is declared, usually invalidating the line numbers in any previous listings of the contents of the Text file.

The KEEPQ form of the command allows a Text file to be saved in its "edit" format, rather than in the format in which files are normally stored in the MPE/3000 file system. This is accomplished by appending the line directory and other EDIT/3000 variables to the Text file and then changing its name to *file name*. The file is given a special file code by EDIT/3000, and when it is again declared TEXT, EDIT/3000 merely rolls in the directory and other text-related variables.

A *range* cannot be declared with the KEEPQ command. SET options in effect when the file is saved in this manner are lost. The Hold file and the SET options in operation when the file is again declared TEXT take effect. However, the pointer position and other variables relating to the text, such as the last line number, will be restored.

The effect of the KEEPQ command is to suspend editing operations on a file indefinitely without the time-consuming process of converting it to and from a "card" type format.

Examples

/ KEEP EDIT02 *cr*

The contents of the Text file are copied, one line to a record, into a file named EDIT02. The line numbers from the Text file are written in ASCII form into the first or last 8 bytes of the lines, depending on the option taken in the SET command.

/ KEEP EDIT02, UNNUMBERED *cr*

The contents of the Text file will be written into EDIT02 without line numbers.

/ KEEP EDIT02 (1000/2000), UNNUMBERED *cr*

Lines 1000 through 2000 of the Text file are stored in the file named EDIT02.

/ KEEPQ TEMP

The Text file is saved by changing its name to TEMP. The Text file is now empty and can be used for other editing operations.

LIST

The LIST command prints specified lines or all lines of the Text file.

Form:

LIST[Q] [*range*] [,UNNUMBERED] [,OFFLINE] [,TRANSLATE] [,NOTEXT]

When *range* is not specified, the line containing the pointer is listed. If ALL is specified as the range, the entire Text file is listed. When *range* is a single position, only the line specified or the line containing the specified position is listed. When a pair of positions is specified, the second position must follow the first.

All lines within the *range* are listed at the terminal unless the OFFLINE option is declared (in which case the text is printed at the system line printer or is sent to the *listfile* specified in the subsystem calling statement. At termination of the operation the file pointer is left resting at the end of the *range.*

Text can be listed in various ways, depending on the options declared. These options have the following effects.

Option	Effect
UNNUMBERED	The listing is printed without line numbers.
OFFLINE	The listing is sent to the system line printer or to the output file specified in the *listfile* option of the subsystem calling statement.
TRANSLATE	All lowercase alphabetic characters are translated to uppercase use for the listing.
NOTEXT	Listing consists of line numbers only.

When no options are taken, the listing is printed, with line numbers, on the user's terminal. If the LISTQ form of the command is declared, the listing is printed without line numbers, providing the same result as the UNNUMBERED option. The user can combine options in any order but he must separate them with commas. Some combinations, such as UNNUMBERED and NOTEXT are obviously incompatible. TRANSLATE may be required for an output device incapable of producing lowercase characters if they are present in the text.

Examples

/ LIST ALL *cr*
/ LIST FIRST/LAST *cr*

Any of the above commands prints the entire Text file on the user's terminal with line numbers.

/ LIST 110 *cr*
/ LIST 110(10) *cr*

Either command prints just line 110.

LIST * or LIST *cr*

Either command prints the line containing the file pointer; the line number is printed.

/ LIST 100/200 *cr*

Prints all lines between 100 and 200 (inclusive) with line numbers.

/ LIST ALL, UNNUMBERED, OFFLINE *cr*

Prints the entire Text file without line numbers on the line printer or on the auxiliary output device designated in the EDIT/3000 subsystem calling statement.

MODIFY

The MODIFY command permits the editing of characters in lines of the Text file through the use of an editing templet. Characters may be deleted, inserted, and replaced.

Form:

MODIFY[Q] *range list*

Each line within *range list* is printed on the terminal, starting at the beginning of the range. The MODIFY command remains in control until the end of the *range list* is reached or until CTRL Y terminates the process.

EDIT/3000 prints a line from the Text file, then returns the carriage and waits for a templet line to be entered from the keyboard directly below the line of text. Three editing operators are used in the templet line:

Operator	Function
D	Deletes the character directly above it. A consecutive string of characters can be deleted by typing a D below each character or below the first and last character in the string to be deleted.
I	Inserts characters in front of the character directly above the I.
R	Replaces characters starting with the character directly above the R.

Only one type of editing operation may take place in one templet line with one exception: it is possible for the second D of a deletion range to be an I followed by characters to be inserted.

To terminate each templet line or to tell EDIT/3000 that a line is to stand as is, either in its original or edited version, type *cr*.

After a line has been edited and displayed for verification, typing CRTL Y instructs EDIT/3000 to ignore the edit operators and restore the line to its original version. The user can then begin again to modify that line. When CTRL Y is typed immediately after EDIT/3000 prints a line to be edited (and before any edit operators are entered), MODIFY is terminated.

The MODIFY command operates as follows:

1. MODIFY followed by the number of the first line in the range is printed, followed by a carriage return. The line of text is then printed and the carriage is again automatically returned. To retain the line in its original form, press *cr* and this procedure is repeated for the next line in the range. To modify the line, move the print carriage, using the space bar, to a point below the character to be modified, and enter an edit operator. Only one operator will be accepted in each templet line, except for a pair or string of Ds. An I can be substituted for the second D to begin an insertion following the deletion.

2. Terminate the templet line with *cr*. EDIT/3000 will then print the line in its edited form, unless the MODIFYQ form of the command was declared, in which case, the new version is not listed. At this point, another templet can be entered. Each time an editing templet is entered, the line is retyped in its new version, and the process is repeated until the line is accepted. Pressing *cr* without entering any new edit operators instructs EDIT/3000 to store the line in its edited form and print the next line to be edited.

Example

```
/MODIFY 1/4 cr
MODIFY 1
EDIT/3000 IS A TEXT EDITING SUB-SYSTEM OF MPE/3000;
                                        Dcr
EDIT/3000 IS A TEXT EDITING SUBSYSTEM OF MPE/3000;
cr
MODIFY 2
THIS SUBSYSTEM CAN BE USED TO INSERT CHARACTERS.
                                              IONE OR MOREΔcr
THIS SUBSYSTEM CAN BE USED TO INSERT ONE OR MORE CHARACTERS,
cr
MODIFY 3
AND TO DELETE OR REPLACE CHARACTERS.
RORcr
ORD TO DELETE OR REPLACE CHARACTERS.
   Dcr
OR TO DELETE OR REPLACE CHARACTERS.
cr
MODIFY 4
THE MODIFY COMMAND CAN PERFORM SEVERAL EDITING OPERATIONS.
                                        D     ITHREE DIFFERENTcr
THE MODIFY COMMAND CAN PERFORM THREE DIFFERENT EDITING OPERATIONS.
cr
/
```

Q

The Q command prints a message on the terminal.

Form:

Q *character string*

The Q command is used in conjunction with the USE command, which reads commands from a file rather than from the terminal. The *character string*, which must be delimited, is printed at the terminal when the Q command is executed. Typical messages describe the type of input expected. (See the USE command.)

The Q command can also be used in batch or interactive mode. However, the result will be that the message is printed in the batch output listing or at the terminal, depending on the mode.

Example

```
Q "WELCOME TO THE ZERO-DELETE USE FILE"
```

When this command is encountered and executed in a file specified in a USE command, the string in quotes is printed at the user's terminal.

REPLACE

The REPLACE command replaces lines in the Text file with lines entered through the terminal keyboard, or, in batch mode, with object records following the command.

Form:

REPLACE[Q] *range list* [,HOLD[Q] [,NOW]]

Entire lines are replaced by this command. (CHANGE and MODIFY are used to replace characters in a line.) If a column or character position appears in the range list, the entire line containing that position is affected. REPLACE operates only on text within the LEFT and RIGHT margins.

If REPLACEQ has not been specified, the first line to be replaced is listed, the carriage is returned, and only the line number is printed. This allows the user to verify that the correct line is being replaced. CTRL Y will terminate the operation at this point with no modifications made. Declaring REPLACEQ eliminates the listing of the line; the user is simply prompted with the line number. In either case, the text replacement is entered, followed by a *cr*. Characters entered replace the original contents of the line and are stored in the Text file; the next line number is then printed. Entering a *cr* without entering any characters has the effect of erasing the contents of the line (but not of deleting the line). New line numbers are printed each time *cr* is pressed until the end of the range is encountered.

To terminate the REPLACE command before reaching the end of the *range list*, enter CTRL Y. If there are no ranges following in the list, EDIT/3000 requests a new command; otherwise it will prompt for the first line in the next range in the list.

If HOLD is declared, CTRL Y performs the function of replacing the line with the contents of the Hold file. The REPLACE operation terminates at the end of the Hold file. Unless the HOLDQ form is specified, the contents of the Hold file are listed. If NOW is specified, EDIT/3000 automatically enters the replacement from the Hold file without waiting for input from the user or the CTRL Y.

When the REPLACE command is executed, the original contents of the line are destroyed. Lines the user does not wish to be lost can be saved with the KEEP command. Or these can be stored in the Hold file, using the HOLD command and specifying the same range of lines that will be replaced.

Example

/REPLACE 10/20,30/40 *cr*

Line 10 is listed at the terminal and line number 10 is printed below. Any input from the keyboard replaces the original contents of line 10. When *cr* is pressed, the next line is printed for verification, and the same procedure is followed, continuing until line 20 is replaced. The range of lines from 30 to 40 is treated in the same way. After line 40 is replaced, the REPLACE command terminates. If CTRL Y is typed during modification of the 10/20 range, the subsystem will terminate the replacement operation for all lines within that range but will prompt for replacement of lines in the second range beginning with line 30.

The SET command is used to alter certain values normally established by EDIT/3000.

Form:

SET [FROM=*line number*] [,DELTA=*increment*] [,LEFT=*column position*]
[,RIGHT=*column position*] [,LENGTH=*line length*] [, QUIET / DISPLAY] [, SHORT / LONG]
[, BATCH / POLL] [, DEPTH=*unsigned integer*] [,TIME=*unsigned integer*]
[,SIZE=*unsigned integer*] [, FRONT / REAR] [, FIXED / VARIABLE]

When EDIT/3000 is initiated, the options described in the SET command are given default values by the subsystem. Any or all of these values may be altered by specification of the SET command. When more than one option is declared, they may appear in any order, but each option after the first should be separated by commas.

SET FROM=*line number* defines the starting line number of the Text file (if it is UNNUMBERED). The *line number* can contain a decimal point, and there can be up to four decimal digits to the left of the point and three to the right. Subsystem default is 1.

SET DELTA=*increment* defines the interval between lines (in UNNUMBERED Text files), and takes the same form as *line number* in SET FROM=. Default is also 1.

SET LEFT=*column position* and RIGHT=*column position* define the first and last columns in a line that EDIT/3000 examines. Default for LEFT is 1, the first *column position* in the line; default for RIGHT is 72. Both column positions must be unsigned integers.

These options cause most of the EDIT/3000 commands to consider only that portion of text within and inclusive of the values specified for LEFT and RIGHT. For example, if the options are declared as SET LEFT=25, RIGHT=50, when text is displayed (e.g., as a result of a FIND command), the character in column 25 appears in column position one, and only characters in columns 25 through 50 are listed.

When performing ADD or INSERT operations, the first character entered is placed in the column specified by LEFT= and characters to the left of that position are initialized to blanks.

The HOLD and REPLACE commands operate only on text within the specified margins.

On the other hand, the GATHER and JOIN commands operate on the entire line of text, ignoring LEFT and RIGHT settings. The KEEP command saves entire lines.

Commands that can extend existing lines (such as MODIFY, INSERT, and CHANGE) shift characters to the right of the RIGHT margin, although only that text within the left and right margins is actually altered. REPLACE is an exception to this; characters are not shifted to the right of the RIGHT margin. The REPLACE command displays only that text within the margins and only replaces that part of the line. The replacement string is padded with blanks if it is shorter. A warning message is issued if the string is longer than the LEFT/RIGHT range.

When portions of a line are deleted with the DELETE command, any characters to the right of the RIGHT margin are moved to the left. However, a DELETE operation that spans entire lines deletes all text between the positions specified, ignoring the LEFT/RIGHT settings.

These options can be used to produce double column listings, to edit only specific portions of a wide text file, and to create longer lines than are allowed on a 72-character terminal.

SET LENGTH=*line length* defines the maximum length to which a line in the Text file can be extended with nonblank characters as a result of editing operations. An edit operation that increases the length so that nonblank characters in a line extend past the character position set by default (72) or by this command will produce a warning message, although the change will still be made. However, if the length of the edited line exceeds that of the original by more than 50 percent, the change will not be made, and the message "OVERSIZED LINE" will be sent, terminating a batch job.

SET QUIET and SET DISPLAY are mutually exclusive options. If QUIET is set, it is equivalent to using the Q option on all commands where it applies except KEEPQ. DISPLAY is the default.

SET SHORT and SET LONG are also mutually exclusive options. SET SHORT redefines the commands to allow the experienced user to enter as a command the initial character of the command name. For example, "A" would be entered instead of "ADD." SET SHORT also shortens several of the subsystem messages. For instance, when termination of EDIT/3000 is requested, a message is sent asking

IF IT IS OK TO CLEAR RESPOND 'YES'

SET SHORT limits this message to

CLEAR?

When SET LONG, the default, is selected, the subsystem will expect the full command names and print the entire text of all messages.

SET BATCH and SET POLL relate to where the subsystem expects to find edit commands, object records, and where output will be sent. To operate in a session from a Use file in full batch mode, (with object records following commands), SET BATCH is declared. SET POLL, the default, is set to return to interactive mode. These options are useful during operation in Use mode, when the same object records are to be applied to many positions in the text. For example, suppose the user wished to test an edit command file that is to be run in batch mode to change some standard type of Text file. He could use the subsystem in interactive mode to create the edit command file, with all the object records for any ADD, REPLACE, MODIFY, or INSERT commands he used following the command (and followed by/ /). He could then KEEP the command file, enter the SET BATCH command, and name the file in a USE command. Any error messages would then be sent to the terminal, allowing him to issue a SET POLL command, declare the command file Text, and correct the error, repeating the process until the edit command file was free of errors.

SET DEPTH= and SET TIME= set limits on the depth of a recursive edit operation (see "Advanced Features") or on the time spent within an iteration. Default for DEPTH= is 10; for TIME=, the default is 50.

SET SIZE= specifies the initial size of the Text file, in number of records. Declaring SET SIZE=8000 results in a Text file that holds a maximum of 8000 records. The default is SET SIZE=0. If the default is taken, then the Text file will hold 5000 records (when it is created with an ADD command or when text is declared from files such as tape or card files, which do not provide this size). Or it will be made 50 percent larger than any user disc files declared TEXT.

SET FRONT and SET REAR are mutually exclusive options that control the location of line numbers in a user file. SET FRONT indicates that line numbers are placed and are expected to be in the first eight bytes of a line. SET REAR, the default, indicates that line numbers are placed and are expected to be in the last eight bytes of a line.

SET FIXED and SET VARIABLE are also mutually exclusive. They specify whether text is stored by the KEEP command in fixed length or variable length format. Default is VARIABLE.

TEXT

The TEXT command copies the contents of a user file into the Text file for editing.

Form:

$$\text{TEXT } \textit{file name} \left[\begin{Bmatrix} \text{(line number/line number)} \\ \text{(\# integer/\# integer)} \end{Bmatrix} \right] \text{ [,UNNUMBERED]}$$

When the TEXT command is executed, the contents of *file name* are written into the Text file. Anything currently in the Text file is lost. (The KEEP command can be used to save the contents of the Text file before executing a TEXT command.) The *file name* specified must refer to an ASCII file.

(*line number/line number*) indicates that only those lines between and inclusive of the given line numbers are to be declared text.

The range (*#integer/#integer*), where each integer represents an actual sequence number on the file, indicates that only those records between and inclusive of the given integers (prefixed by #) are declared text.

Unless the UNNUMBERED option has been declared, the first or last eight bytes (depending on the SET command option taken) of each line in the user file, which should contain the line numbers stored in ASCII format, are removed and used as line numbers in the Text file. The SET LENGTH= option is then set to the line length of the file less eight columns. If the UNNUMBERED option is specified, the lines are numbered starting with 1 and increasing by 1 unless the SET command has been used to declare a different numbering scheme. Some files may require special access codes, or may be unavailable, either because of restrictions placed on them when created or because they are of the wrong type. If EDIT/3000 cannot locate a file, a message is printed at the terminal.

If there are no line numbers stored in the file named, but the UNNUMBERED option has not been declared, the subsystem will respond with a message. The command must be repeated specifying the UNNUMBERED option.

Examples

```
/TEXT EDIT02, UNNUMBERED cr
```

The contents of the file named EDIT02 are copied into the Text file; any existing contents are lost. Line numbers are assigned in accordance with default or declared values for numbering lines. The contents of the named file are not changed.

```
/TEXT EDIT02 cr
```

The same procedure as just described takes place, except that the line numbers stored in the first or last eight bytes of EDIT02 are used as line numbers in the Text file.

USE

The USE command designates a file to be used by EDIT/3000 as a source of commands.

Form:

USE *file name*

EDIT/3000 reads commands from the named file until it reaches an end-of-file marker. In an interactive session, specifying the USE command changes the mode of operation to use mode. In use mode, commands are read from the named file, but any messages and requests for object records are sent to the user's terminal. Files can be written with Q commands that will prompt the user for object records and any other EDIT/3000 input.

Specifying the USE command in batch mode causes EDIT/3000 to branch to the named file, where commands are read until the end-of-file, at which point the next command after USE is read. No interaction with the user takes place, and any object records required must be within the file unless another branch out of the file is made.

Use files may be "nested" as many times as the DEPTH= option in the SET command allows.

Example

```
/USE EDFIL cr

EDFIL  Q "BEGIN EDFIL TO MODIFY DOUBLE PRECISION TO COMPLEX OR&
       REAL."
       WHILE
           FIND "DOUBLE PRECISION"
           BEGIN
           Q "ENTER YOUR TEMPLET-OTHERWISE HIT RETURN."
           MODIFY *
           END
```

(Continue in interactive mode.)

Assuming EDIT/3000 is initially in interactive mode, when the command USE EDFIL is executed, the subsystem will go into the use mode and begin reading and executing commands from the file named EDFIL. After the first message is printed at the user's terminal, EDIT/3000 begins the operation to find and modify the term "DOUBLE PRECISION" where it occurs in the Text file after the current position of the pointer. (See Section IV, "Advanced Features," for an explanation of the metacommands used in this example.) Whenever the user wishes to change the term, he enters a modification templet; if he does not wish the term changed, he presses *cr*. At the end of the operation, the subsystem returns to interactive mode and continues with commands entered by the user.

VERIFY

The VERIFY command prints the values of options declared in the SET command.

Form:

$$\text{VERIFY} \left[\left\{ \begin{matrix} \text{ALL} \\ \textit{option list} \end{matrix} \right\} \right]$$

Options declared in the SET command are printed at the terminal. If no options were specified, default values are given.

When *option list* contains more than one item, items may be in any order but must be separated by commas. When ALL is specified, all SET options are printed.

If no parameters are given, the position of the pointer is displayed (as in FIND *).

Examples

/VERIFY FROM, DELTA, LENGTH *cr*

Starting line number and increment between lines of the Text file and the maximum column to which nonblank characters can be extended are printed.

/VERIFY ALL *cr*

All options under the control of the SET command are printed.

XPLAIN

The XPLAIN command prints an explanation of selected commands or all commands.

Form:

XPLAIN [{ *command name list* / ALL }] [,OFFLINE]

XPLAIN *command name list* prints a description of each command name stated and a brief explanation of how the command is used. Initial letters may be used instead of the full command name, and if more than one is requested, they may be in any order but must be separated by commas. XPLAIN may be entered as X.

XPLAIN ALL lists and explains all EDIT/3000 commands and prints a brief introduction to the subsystem.

Declaring only "X" or "XPLAIN" with no parameters will explain the XPLAIN command.

OFFLINE sends the listing to the system line printer or to the file designated as the *listfile* in the subsystem calling command.

Example

```
XPLAIN F,M,R
```

Explanations and the correct spelling of FIND, MODIFY, and REPLACE will be printed at the terminal.

Z ::=

The Z ::= command assigns the object record following this command to Z ::= .

Form:

Z ::= *object record*

Whenever Z ::= is found in a edit command, it is replaced by the object record initially assigned to Z ::= . This allows Z ::= to be used as a variable. The object record assigned to Z ::= may be a string or position in the Text file, or any comment. A string must be delimited.

Example

```
Q "THIS SUB-SUBSYSTEM WILL MODIFY ANY STRING."
Q "ENTER STRING TO BE MODIFIED."
Z ::=
WHILE
    FIND Z ::=
    BEGIN
    Q "ENTER MODIFICATION TEMPLET-OR TYPE RETURN."
    MODIFY *
    END
```

This example could be a sub-subsystem in a file called by the USE command to perform a modification operation using any templet entered to change any string assigned to Z ::= . When the user is prompted by the message "ENTER STRING TO BE MODIFIED.", he enters at the terminal the delimited character string he wishes altered, which is then assigned to Z ::= . As soon as the subsystem finds this character string, it requests from the user the modification templet, which is also entered at the terminal. This continues until all occurrences of the string assigned to Z ::= are found and modified. See Section IV, "Advanced Features," for an explanation of the metacommands used in this example.

SECTION IV
Advanced Features

EDIT/3000 has several features designed primarily for the advanced user of the subsystem. These features include the capability of operating in a combined interactive-batch mode, called the *use mode*; of providing the user with a set of metacommands that control interpretation of the basic command set; and of allowing the subsystem to call procedures written in other languages from an outside library.

USE MODE

Use mode is initiated with the USE command and the BATCH/POLL option in the SET command. When SET POLL (the default) has been set, the subsystem reads edit commands from the file designated in the USE command as if operating in batch mode; messages, requests for input, and listings, however, are sent to the terminal. If the *listfile* option was declared in the MPE/3000 log-on command, off-line listings will be sent to that file. SET BATCH can be declared in the Use file so that object records can be included as part of the Use file. The Q command is used to send messages, such as requests for input, to the terminal.

METACOMMANDS

Metacommands differ from the basic commands in the EDIT/3000 command language in that they control execution of the basic commands. Because the subsystem follows the same rules used by most programing languages for interpreting logical expressions, these metacommands can be "programed" to form a sub-subsystem of EDIT/3000 that can either be kept in a user file and called into operation by the USE command or be included as part of a batch job.

When the six metacommands WHILE, BEGIN, END, OR, YES, and NOT are used in a logical sequence with the basic EDIT/3000 commands, they form command blocks with recursive properties. Execution of these blocks is contingent on the text of the commands and on the text being edited.

Input to the subsystem may be considered a logical expression with an assumed logical *AND* between each edit expression, with BEGIN and END commands serving as left and right parentheses, and with OR and NOT commands having their usual functions as logical operators. The basic edit commands resemble logical functions that perform edit operations only if the conditions set by the metacommands permit.

WHILE Command

The WHILE command instructs EDIT/3000 to save the next two edit expressions and repeatedly interpret them until the first one "fails." This failure will be the result of a "soft-fail" condition defined for each command. A *soft-fail* occurs when an otherwise correct edit command cannot be executed because of the configuration of the Text file, the pointer, or edit-option settings. This sort of failure is different from syntactical errors in edit commands which are *hard-fail* conditions because they can cause an operation to end. When the soft-fail condition occurs, EDIT/3000 sets a global logical variable (called the FLAG) to have a value of false. When the FLAG is set false only the OR or YES command is executed.

Example

```
WHILE
FINDQ"this"
LIST *
```

EDIT/3000 scans the Text file from the current pointer position forward to the first occurrence of the string "this." The second expression is then interpreted and that line is listed. Then the first expression is interpreted again, and the next case of the string "this" is found and listed. This process continues until the end of the Text file is found before another "this". When the end of the file is found, the FLAG is set to false and the second expression is not interpreted. The WHILE command then terminates and the FLAG is reset to true.

The WHILE command can also take the form:

```
WHILE [FLAG]
```

When the FLAG option appears in the command, the object of the WHILE command is only one expression, which is iterated until the FLAG is set to false.

BEGIN-END Command

The scope of the WHILE command is defined as the next two edit expressions in the file, an edit expression being either

- An edit command, or
- All edit commands or expressions within a matching BEGIN-END pair.

In addition, expressions may contain nested WHILE commands. Thus, the power of the WHILE command can be considerably extended since one or both of the expressions can be sequences of edit commands or expressions.

BEGIN can take the form BEGINQ. This suppresses all soft-fail error messages until the matching END statement is encountered.

Example

```
WHILE
FINDQ"this"(+3)
    BEGINQ
    FINDQ"program"/*(+20)
    LIST *
    END
```

This WHILE command is interpreted as follows.

The first expression of the WHILE command (FINDQ"this"(+3)) is interpreted and causes a scan forward to the next "this" in the Text file. Then the second expression, which consists of the two commands between BEGIN and END, is interpreted. The first of these commands attempts to FIND the string "program" within 20 nonblank characters of the pointer. (The pointer has been left at the end of the string "this" by the first FIND command.)

If the second FIND succeeds in finding "program" within 20 nonblank characters, the LIST command causes that line to be printed. If the second FIND does not find its object string, the FLAG is set to false, and the LIST command is passed over with no action. When the END command signals the end of the second expression, the FLAG is set to true before the first expression is again interpreted. The iteration continues until the first expression fails because no more cases of "this" are in the Text; only those cases of "this" followed by "program" within 20 nonblank characters are listed.

NOT Command

The NOT command causes the setting of the FLAG to be reversed after the next following expression is interpreted. If the second FIND in the above example were preceded by a NOT command, only the cases of "this" that were *not* followed by "program" would be listed.

OR Command

The OR command allows for alternation and is interpreted when the FLAG is false. If the flag is false an OR command resets it to true; if the FLAG is true, an OR command causes the expression following it to be passed over with no action. An OR command will reset a false FLAG only when it occurs at the same or a higher textual level when compared with the command that set the FLAG false. Any expression between BEGIN and END that is encountered when the FLAG is false is skipped over; consequently, any OR commands within it are not seen.

YES Command

The YES command sets the FLAG to true, like the OR command, but it does not cause the next expression to be skipped if the FLAG already is set to true.

PROCEDURE COMMAND

An additional advanced feature of EDIT/3000, the PROCEDURE command, allows the subsystem to call and use logical procedures written in SPL or FORTRAN and stored in the system, project, or group library.

The calling command takes the form

PROCEDURE *pname*, $\left\{\begin{matrix}G\\P\\S\end{matrix}\right\}$, *range list*

pname is the name of the procedure being called. The second parameter, G, P, or S, indicates the location of the procedure, G denoting the group library, P the project library, and S the system library. *range list* refers to the location in the Text file of the text to be affected by the procedure. If a character or column position is specified, the entire line containing the position will be affected.

For the subsystem to be able to call a procedure, certain conventions must be followed when the procedure is written. The procedure must be logical. The order and type of the parameters used in the procedure are as shown in the following SPL and FORTRAN examples:

A sample SPL logical procedure:

```
LOGICAL PROCEDURE SAMPLE (STRING, SIZE, NUMBER, SPACE)
BYTE ARRAY STRING; NUMBER;
INTEGER SIZE;
ARRAY SPACE;
```

A sample FORTRAN logical procedure:

```
LOGICAL FUNCTION SAMPLE (STRING, SIZE, NUMBER, SPACE)
INTEGER SIZE, SPACE(20)
CHARACTER STRING*80, NUMBER*8
```

The first parameter, STRING, is a byte or character array containing each line to be affected by the procedure. The second parameter, SIZE, is assigned the size of the line of the text, e.g., 72, if that is the column position of the last non-blank character in the line. NUMBER, the third parameter, is assigned the actual number of the line in the Text file in byte or character form. SPACE is an array used as scratch space for the procedure and is initialized to zeros before the procedure is first called. Although the procedure may not use all of these parameters, they must be present and in the correct order.

When a procedure is called by EDIT/3000, the LOADER subsystem of MPE/3000 loads the procedure using a dynamic procedure loading intrinsic. The procedure is then dynamically loaded into the user's processes. EDIT/3000 puts a pointer to an array containing the first line in the first range of the range list and provides the procedure with the size of the line and the actual line number. The procedure can then use this information in performing its particular function, for example, in sending output to the job or session list device. When it finishes with each line in the range list, the procedure ends. Any changes to the line will be entered as a

REPLACE of the old version. If the end of the range affected by the procedure has not been reached, and if the logical value returned is true, the procedure will be called again with the next line in the range. When a false value is returned, the procedure will receive no more lines and the FLAG will be set false as with a soft failing edit command. Only text within the LEFT and RIGHT margins are affected by the procedure.

APPENDIX A
Errors and Error Messages

Two main categories of errors are recognized by EDIT/3000: syntactical and contextual.

Any deviation from the defined grammar of the subsystem command language will cause a syntax error. Misspelling a command name, entering too many or too few parameters, using the wrong symbol to separate items in a list, or entering parameters out of order nearly always cause syntax errors. Because syntax errors normally cause an operation to end, they are referred to as "hard-fail" errors.

When the current configuration of EDIT/3000 makes it impossible to interpret an edit command, a context error will result. For example, if the SET LENGTH=option had been set to 25 and the user entered a command that was syntactically correct to find column position 57, the result would be a context error. Context errors are referred to as "soft-fail" errors.

The effect of either type of error depends upon the mode in which the subsystem is operating.

INTERACTIVE MODE

When operating in interactive mode, EDIT/3000 will reject a command containing a syntax error and print two asterisks (**) and the number of the error message. (Error messages and their corresponding numbers are listed in a table at the end of this appendix.) If the user is not interested in the text of the message, or is already familiar with it, he may type *cr*, and the subsystem will prompt for a new entry. If he wishes to see the text of the message, he may type any character, and the message will be printed at the terminal. He may then try to enter the command again. It is possible to recover from all syntax errors in interactive mode by re-entering the command.

If the command is syntactically correct but contains a context error, it does not cause a terminal error. However, an informative message will be printed. When a nonexistent line number or string is specified, EDIT/3000 searches the Text file, and if the specified line or string cannot be found, it issues an appropriate message.

BATCH MODE

Syntax errors occurring in batch mode cause termination of the processes using the subsystem. The command that caused the error is flagged with the same message printed at the terminal during an interactive session. Commands entered in batch modemay be checked in use mode for freedom from errors. Context errors in batch mode do not cause a process to abort, but simply set the global variable (FLAG) false. This allows the NOT or OR metacommands to be used in such a way that the presence or absence of particular records, strings, or combinations of these may be used to automatically control further editing.

TABLE OF ERROR MESSAGES

Table A-1 is a list of error messages that the user may receive at the terminal or in his batch output. Most messages are self-explanatory. Messages designated as "code" refer to system (MPE/3000) errors. Messages such as number 26, *GETDSEG FAILURE*, and number 30, *'DMOVOUT' ERROR*, refer to that intrinsic in the MPE/3000 file system (i.e., GETDSEG and DMOVOUT). System messages are most likely to occur only once, and the user is advised to reattempt the operation. Refer to *HP 3000 Multiprogramming Executive Operating System (03000-90005)* for help with MPE/3000 error messages.

The user receives a message in the following form:

```
**number**message
```

For example:

```
**1**INVALID COMMAND NAME
```

In interactive mode, only the message number is printed at the terminal. For example:

```
**1
```

If the user wishes to receive the text of the message, he types any character, such as a slash:

```
**1/
```

EDIT/3000 responds then with the entire message:

```
**1/**INVALID COMMAND NAME
```

Table A-1. Error Messages

Number	Message	Type
1	INVALID COMMAND NAME	Hard
2	INVALID OPTION	Hard
3	MISSING PARAMETER	Hard
4	FILE NOT ACCESSIBLE (code)	Hard
5	FILE NOT TYPE ASCII	Hard
6	CHARACTER ADJUSTMENT RUNS OFF OF FILE	Soft
7	READ ERROR ON TEXT FILE (code)	Hard
8	ADJUSTMENT RUNS OFF OF FILE	Soft
9	SYNTAX ERROR IN TEXT POSITION EXPRESSION	Hard
10	ABSOLUTE COLUMN POSITION OUT OF RANGE	Soft
11	POSITION NOT FOUND	Soft
12	OVERSIZED LINE (line number)	Hard
13	TIMEOUT ERROR	Hard
14	INVALID LINE NUMBER	Hard
15	COMMAND WILL NOT REPLACE OR INTERLEAVE LINE	Soft
16	WARNING – LINE nnnn.nnn EXTENDS PAST COLUMN nn	Warning
17	UNMATCHED 'END' COMMAND	Hard
18	INSUFFICIENT LINE NUMBERS FOR 'GATHER' OR 'JOIN'- NOT PERFORMED	Hard
20	UNDELIMITED STRING	Hard
21	STRING NOT FOUND BEFORE LIMIT	Soft
22	LINE DOES NOT EXIST	Soft
23	FAILURE TO OPEN 'TEXT' FILE (code)	Hard
24	NON-EDITABLE FILE TYPE (code)	Hard
25	SCRATCH FILE OPEN FAILURE (code)	Hard
26	GETDSEG FAILURE (seg size)	Hard
27	E.O.D. BEFORE LAST RECORD ON 'TEXT' FILE – WARNING ONLY	Warning
28	SCRATCH FILE READ ERROR	Hard
29	INVALID LINENUMBER ON 'NUMBERED' TEXT FILE – RECORD (number is given)	Hard
30	'DMOVOUT' ERROR	Hard
31	SCRATCH FILE WRITE ERROR (code)	Hard
32	'DMOVIN' ERROR	Hard
33	INVALID COLUMN RANGE (FIRST FOLLOWS SECOND)	Hard
34	WARNING – 'HOLD' FILE IS NULL	Warning

Table A-1. Error Messages *(Continued)*

Number	Message	Type
36	SCRATCH FILE IS FILLED – 'KEEP' AND DECLARE 'TEXT' AGAIN	Hard
37	INVALID INTEGER	Hard
38	INVALID RECORD-LINE PAIR (SUBSYSTEM PROBLEM)	Hard
39	INVALID RANGE (FIRST POSITION FOLLOWS SECOND)	Hard
40	UNDEFINED TEXT	Hard
41	FAILURE TO OPEN 'KEEP' FILE (code)	Hard
42	WRITE ERROR ON 'KEEP' FILE (code)	Hard
43	OUT OF SEQUENCE LINENUMBER IN 'TEXT' FILE	Hard
44	LINE NUMBER ZERO CAN NOT BE ACCESSED	Soft
45	READ ERROR ON COMMAND FILE (code)	Hard
46	COMMAND CONTINUED PAST 256 CHARACTERS	Hard
47	FAILURE TO OPEN USE FILE (code)	Hard
48	FAILURE TO OPEN WHILE FILE (code)	Hard
49	WRITE ERROR ON WHILE FILE	Hard
50	WHILE FILE OVERFLOW	Hard
51	FAILURE TO OPEN JOIN FILE (code)	Soft
52	READ ERROR ON JOIN FILE (code)	Hard
53	TIME-OUT ON WHILE ITERATION	Soft
54	SYNTAX ERROR IN ABSOLUTE COLUMN POSITION EXPRESSION	Hard
55	READ ERROR ON HOLD FILE (code)	Hard
56	READ ERROR ON INPUT (code)	Hard
57	END OF INPUT FILE	Soft
58	FAILURE TO OPEN HOLD FILE (code)	Hard
59	WRITE ERROR ON HOLD FILE (code)	Hard
60	FCLOSE FAILURE (code)	Hard

APPENDIX B
Command Language

Table B-1 allows the user to easily determine the precise syntax requirements of a command, or to quickly refresh his knowledge of the command language. The table is helpful to have, particularly during an interactive session, as a general reference tool. If further clarification is needed, the page number of a complete explanation of each command is provided.

Table B-1. Command Language Reference Chart

Command	Syntax	Use	Example	Remarks	Page Reference
ADD	ADD[Q] [*line number*] [,HOLD[Q] [,NOW]]	To enter lines of text into the Text file from the keyboard or from the Hold file.	ADD 60, HOLD	The contents of the Hold file are entered into the Text file beginning with the line number 60.	3-7
BEGIN	BEGIN[Q]	Used as the first expression in a matching BEGIN-END pair.	WHILE FINDQ"this"(+3) BEGINQ FINDQ"program"/*(+20) LIST* END	This is a subsubsystem using EDIT/3000 metacommands.	4-2
CHANGE	CHANGE[Q] [{*abs. col. pos.*/[*abs. col. pos.*] {*char. string*}] TO *char. string* [IN *range list*]	To replace old text with new text.	CHANGE "RECORD" TO "LINE" IN 40/70	All occurrences of RECORD in lines 40 through 70 will be changed to read LINE.	3-9
DELETE	DELETE[Q] [*range list*]	To delete characters and lines from the Text file.	DELETE 50/75, 150(3)/155(7)	Lines 50 through 75 and characters in the range beginning with the third column in line 150 through the seventh column in line 155 will be deleted.	3-11
END	END	To terminate execution of EDIT/3000 or, when used with a matching BEGIN command, to terminate an iterative operation.	END	Control is returned to MPE/3000.	3-13
FIND	FIND[Q] *range*	To locate a point in the Text file.	FIND "PROGRAM"	The pointer will move to the next occurrence of the string PROGRAM.	3-14
GATHER	GATHER[Q] *range* TO *line number* [BY *increment*]	To move portions of text from one location to another in the Text file and renumber the lines.	GATHER 50/55 TO 5.1 BY .1	Lines 50 through 55 will be moved to begin at line 5.1 and will be renumbered in increments of .1.	3-16
HOLD	HOLD[Q] [*range*] [,APPEND]	To copy text from the Text file into the Hold file.	HOLD 100/150, APPEND	Lines 100 through 150 are copied into the Hold file and are appended to the text that already exists in the Hold file.	3-18
INSERT	INSERT[Q] *position* [BY *increment*] [,HOLD[Q] [,NOW]]	To insert text into the Text file from the terminal and the Hold file.	INSERT 25(4)	Line 25 is printed with the pointer indicated directly under column 4. Any text may then be typed in. The remainder of line 25 is saved and printed following the insertion.	3-19
JOIN	JOIN[Q] *file name* [{(*line number/line number*) {(# *integer*/# *integer*)}] [TO *line number*] [BY *increment*]	To add all or part of a file to the Text file.	JOIN EDIT02 TO 1000 BY .1	The text in EDIT02 is added to the Text file beginning at line 1000. Lines are numbered in increments of .1.	3-20

Table B-1. Command Language Reference Chart *(Continued)*

Command	Syntax	Use	Example	Remarks	Page Reference
KEEP	KEEP[Q] *file name* [(*range*)] [,UNNUMBERED]	To save all or part of the Text file in a user file.	KEEP EDIT02	The contents of the Text file are saved in EDIT02, a user file, and the line numbers are stored in ASCII format in the first or last 8 bytes of each line.	3-22
LIST	LIST[Q] [*range*] [,UNNUMBERED] [,OFFLINE] [,TRANSLATE] [,NOTEXT]	To print out any portion or all of the Text file.	LIST ALL UNNUMBERED, OFFLINE	The contents of the Text file are printed on the file printer or at the device specified in the system log-on command. Line numbers are not printed.	3-24
MODIFY	MODIFY[Q] *range list*	To modify text in the Text file using three operations: delete (D), insert (I), and replace (R).	MODIFY 50/100	Lines 50 through 100 are printed line by line to allow a modification templet to be entered for each line in the range.	3-26
NOT	NOT	Causes setting of FLAG to be reversed after the next following expression is interpreted.	WHILE FINDQ,"this"(+3) BEGINQ NOT FINDQ"program"/*(+20) LIST* END	Occurrences of "this" that are *not* followed by "program" are listed.	4-3
OR	OR	Resets FLAG to true if it is false, or to false if FLAG is true.	BEGINQ FINDQ"this"/*(+40) OR FINDQ"program"/*(+20) END	If "this" is found within 40 nonblank characters of the pointer, the second FIND command is skipped over. If "this" is not found, the FLAG is set to true, and the second FIND is executed.	4-3
PROCEDURE	PROCEDURE *pname*, {G / P / S}, *range list*	Calls logical procedure stored in a library outside the subsystem.	PROCEDURE CARDTEST, G, 100/300	Logical procedure CARDTEST is called from the group library. Records 100 through 300 of the Text file are affected by the procedure.	4-4
Q	Q *character string*	To print a message at the terminal while in Use mode.	Q "HELLO. PLEASE ENTER YOUR IDENTITY CODE."	When this command is executed, the delimited message will be printed at the user's terminal.	3-28
REPLACE	REPLACE[Q] *range list* [,HOLD[Q][,NOW]]	To replace lines in the Text file.	REPLACE 10/20	Each line in the range from 10 through 20 is listed followed by just the line number to allow the user to enter a text replacement.	3-29

Table B-1. Command Language Reference Chart *(Continued)*

Command	Syntax	Use	Example	Remarks	Page Reference
SET	SET[FROM=*line number*] [,DELTA=*increment*] [,LEFT=*column position*] [,RIGHT=*column position*] [,LENGTH=*line length*] [,{QUIET / DISPLAY}] [,{SHORT / LONG}] [,{BATCH / POLL}] [,DEPTH=*unsigned integer*] [,TIME=*unsigned integer*] [,SIZE=*unsigned integer*] [,{FRONT / REAR}] [,{FIXED / VARIABLE}]	To alter options that are normally set by the subsystem and that govern editing operations.	SET FROM=100 DELTA=10, QUIET, SHORT	Lines in the Text file will begin with line number 100 and will be numbered in increments of 10. The position of the pointer will not be displayed each time a location in the Text file is found, and commands that take "Q" operate as if Q had been specified. Commands can be entered using only their initial character. Otherwise, the first and last column position read in a line of the Text file will be 1 and 72, respectively.	3-30
TEXT	TEXT *file name* [{*(line number/line number)* / *(# integer/# integer)*}] [,UNNUMBERED]	To copy contents of a user file into the Text file to be edited.	TEXT EDIT02	Contents of EDIT02 are placed in the Text file and numbered using the line numbers stored in the first or last eight bytes of each line.	3-33
USE	USE *file name*	To instruct the subsystem to read commands from a user file but to send messages to and expect input from the terminal.	USE EDFIL	EDIT/3000 will go to the beginning of EDFIL and execute commands from that file until it finds end-of-file mark.	3-34
VERIFY	VERIFY[{ALL / *option list*}]	To obtain a reminder of the values of options declared in the SET command.	VERIFY DELTA, FROM	The values set up for DELTA and FROM when EDIT/3000 was initialized as listed.	3-35
WHILE	WHILE[FLAG]	Causes EDIT/3000 to iterate the next two edit expressions (or one, if FLAG is declared) until the first one fails.	WHILE FIND "this" LIST *	Each occurrence of "this" following the pointer is found and listed until the end of the Text file is found before another "this" is found. The FLAG is then set to false.	4-2
XPLAIN	XPLAIN[{*com. name list* / ALL}] [,OFFLINE]	To obtain an explanation of the commands.	XPLAIN A,S	Explanations of the ADD and SET commands will be printed at the terminal.	3-36
YES	YES	Sets the FLAG to true			4-3
Z : :=	Z : := *object record*	To assign the value of the object record to Z : :=.	Z : := "DOUBLE PRECISION" / /	The string DOUBLE PRECISION will be substituted for Z : := whenever Z : := occurs in a command outside of a string or filename.	3-37

APPENDIX C
ASCII Character Set

Graphic	Decimal Value	Octal Value	Comments
	0	0	Null
	1	1	Start of heading
	2	2	Start of text
	3	3	End of text
	4	4	End of transmission
	5	5	Enquiry
	6	6	Acknowledge
	7	7	Bell
	8	10	Backspace
	9	11	Horizontal tabulation
	10	12	Line feed
	11	13	Vertical tabulation
	12	14	Form feed
	13	15	Carriage return
	14	16	Shift out
	15	17	Shift in
	16	20	Data link escape
	17	21	Device control 1
	18	22	Device control 2
	19	23	Device control 3
	20	24	Device control 4
	21	25	Negative acknowledge
	22	26	Synchronous idle
	23	27	End of transmission block
	24	30	Cancel
	25	31	End of medium
	26	32	Substitute
	27	33	Escape
	28	34	File separator
	29	35	Group separator
	30	36	Record separator
	31	37	Unit separator
	32	40	Space
!	33	41	Exclamation point
"	34	42	Quotation mark
#	35	43	Number sign
$	36	44	Dollar sign

Graphic	Decimal Value	Octal Value	Comments
%	37	45	Percent sign
&	38	46	Ampersand
'	39	47	Apostrophe
(	40	50	Opening parenthesis
)	41	51	Closing parenthesis
*	42	52	Asterisk
+	43	53	Plus
,	44	54	Comma
-	45	55	Hyphen (Minus)
.	46	56	Period (Decimal)
/	47	57	Slant
0	48	60	Zero
1	49	61	One
2	50	62	Two
3	51	63	Three
4	52	64	Four
5	53	65	Five
6	54	66	Six
7	55	67	Seven
8	56	70	Eight
9	57	71	Nine
:	58	72	Colon
;	59	73	Semi-colon
<	60	74	Less than
=	61	75	Equals
>	62	76	Greater than
?	63	77	Question mark
@	64	100	Commercial at
A	65	101	Uppercase A
B	66	102	Uppercase B
C	67	103	Uppercase C
D	68	104	Uppercase D
E	69	105	Uppercase E
F	70	106	Uppercase F
G	71	107	Uppercase G
H	72	110	Uppercase H
I	73	111	Uppercase I
J	74	112	Uppercase J
K	75	113	Uppercase K
L	76	114	Uppercase L
M	77	115	Uppercase M
N	78	116	Uppercase N
O	79	117	Uppercase O
P	80	120	Uppercase P
Q	81	121	Uppercase Q
R	82	122	Uppercase R
S	83	123	Uppercase S
T	84	124	Uppercase T
U	85	125	Uppercase U
V	86	126	Uppercase V
W	87	127	Uppercase W

Graphic	Decimal Value	Octal Value	Comments
X	88	130	Uppercase X
Y	89	131	Uppercase Y
Z	90	132	Uppercase Z
[	91	133	Opening bracket
\	92	134	Reverse slant
]	93	135	Closing bracket
^	94	136	Circumflex
_	95	137	Underscore
`	96	140	Grave accent
a	97	141	Lowercase a
b	98	142	Lowercase b
c	99	143	Lowercase c
d	100	144	Lowercase d
e	101	145	Lowercase e
f	102	146	Lowercase f
g	103	147	Lowercase g
h	104	150	Lowercase h
i	105	151	Lowercase i
j	106	151	Lowercase j
k	107	152	Lowercase k
l	108	154	Lowercase l
m	109	155	Lowercase m
n	110	156	Lowercase n
o	111	157	Lowercase o
p	112	160	Lowercase p
q	113	161	Lowercase q
r	114	162	Lowercase r
s	115	163	Lowercase s
t	116	164	Lowercase t
u	117	165	Lowercase u
v	118	166	Lowercase v
w	119	167	Lowercase w
x	120	170	Lowercase x
y	121	171	Lowercase y
z	122	172	Lowercase z
{	123	173	Opening (left) brace
|	124	174	Vertical line
}	125	175	Closing (right) brace
~	126	177	Tilde
	127	177	Delete

INDEX