Price: $2.00

TYMSHARE TYMCOM-IX

INSTANT SERIES

EDITOR

TYMSHARE, INC.
CUPERTINO, CALIFORNIA 95014

TYMSHARE®

©1976, TYMSHARE, INC., Litho in U.S.A.

FEBRUARY 1978
VERSION 1

BS5

CONTENTS

Section 1
INTRODUCTION

EDITOR allows the Tymshare user to edit text quickly and easily. The text may be the Gettysburg Address, a computer program written in any language, or data to be read by the computer. The text to be edited may be entered from an existing file on a storage device or typed directly into EDITOR at the terminal.

ABOUT THIS MANUAL

This manual is designed to acquaint the beginning user with the principal features of the EDITOR. Sections describe how to enter text for editing and how to:

- Edit a line of text
- Print a line or range of lines of text
- Delete a line or range of lines from the text
- Insert a line or lines into the text
- Make substitutions throughout the text
- Write the text on a disk storage file

This manual is for the beginning user of EDITOR. Thus, while the entire range of editing possibilities is covered, not all of the more complex features are described. Tymshare's other EDITOR reference manuals describe finer details that allow more intricate text manipulation.

EDITOR is an editing language. A BASIC or FORTRAN program cannot be executed in EDITOR on the TYMCOM-IX. To run a program prepared using EDITOR, the user must write the edited program on a disk file, exit from EDITOR, and then execute the program.

The TYMCOM-IX EDITOR described here is nearly identical to the EDITOR available on Tymshare's TYMCOM-X system, but some procedures, particularly commands used to call or exit from EDITOR, differ. TYMCOM-X users should consult Tymshare's *TYMCOM-X EDITOR Instant Manual* for information on these features.

SYMBOL CONVENTIONS

To indicate what is printed by the computer and what is typed by the user, all user-typed commands in this manual are underlined.

When a carriage return is used to terminate a command or text line, it is shown by the symbol ↵.

When a control character is used, it is shown on the line on which it would be typed. Control characters are shown by a superscript c to the right of an alphabetic character. Control characters are not actually typed by the system, however; they appear in examples in this manual for demonstration only.

Section 2
ENTERING AND LEAVING EDITOR

To gain access to EDITOR, the user must first log in to the system. As soon as the connection to the Tymshare system is made, the system prints:

```
please log in:
```

The user should enter his user name followed by a colon (:) and the system number. A carriage return terminates the line. The system then prompts for the user's password:

```
password:
```

The user should enter his password and a carriage return. The system then prompts the user for an optional project code. The user may enter a carriage return in lieu of a project code, as shown below.

```
PROJ CODE:↩
```

Once the log in procedure is finished, the system prints the Tymshare system number, time, and date; returns the carriage; and prints a dash at the left margin. For example:

```
TYMSHARE    C2  9/15/76   10:27
—
```

The dash prompt (—) indicates that the user is now in the EXECUTIVE and can issue any valid EXECUTIVE command. EDITOR can be called by typing EDITOR followed by a carriage return, as shown below.

```
—EDITOR↩
*
```

An asterisk printed at the left margin indicates EDITOR command level. EDITOR is now ready to accept a command.

To exit from the Tymshare system, the user must first return to the EXECUTIVE. The QUIT command, typed at EDITOR command level, allows the user to exit from EDITOR and return to the EXECUTIVE.

```
*QUIT↩

—
```

The system prints a dash at the left margin, indicating that the user has reentered the EXECUTIVE. The LOG command logs the user off the system.

```
–LOG ↵
```

Following LOG, the system prints one line of user information and exits. The user can then disconnect the line.

Section 3
ENTERING TEXT INTO EDITOR

The APPEND command adds text to the EDITOR text area. With APPEND, the user can enter an entire program into EDITOR, editing as it is being typed. Text entered into EDITOR is not permanently saved on a disk storage device until the user issues the WRITE command.

The WRITE command copies the user's text on a disk file but does not erase any lines from the user's on line text area. This means that text can exist in two places: in the user's on line text area and on a disk file specified by the user. The CLEAR command erases the user's on line text area but not the disk file.

The user should periodically use the WRITE command to save text being entered. This ensures that, in case of system difficulty, the user will not lose all his text. To terminate text entry in EDITOR, the user types a carriage return followed by a control D (D^c). This returns the user to EDITOR command level. To resume text entry, the user types APPEND. The following example demonstrates how to use the APPEND and WRITE commands.

```
-EDITOR↵
*APPEND↵
THIS IS LINE ONE OF THE TEXT↵
ENTRY OF TEXT INTO EDITOR IS↵
TERMINATED WITH A CONTROL D.↵
THE WRITE COMMAND ENSURES THAT↵
THE TEXT IS SAVED. THE USER↵
MUST SUPPLY A NAME FOR THE DISK FILE↵
THAT IS TO CONTAIN THE↵
EDITOR TEXT LINES.↵
D^c
*WRITE↵
TO:MYFILE↵      EDITOR prompts the user for the name of the file on which the text is to be saved.
 NEW FILE↵      The file named MYFILE is a new file. The user confirms the WRITE command by
225 CHARS       entering a carriage return. The system then prints the number of characters in the file.
*APPEND↵
END OF TEXT↵    The user adds a line of text with APPEND.
D^c
*WRITE↵         The user saves the updated file MYFILE.
TO:MYFILE↵
 OLD FILE↵      MYFILE already exists. The user types a carriage return to replace the old contents of
237 CHARS       MYFILE with the updated copy.
*               MYFILE now contains 237 characters.
```

Each time the user issues the WRITE command, the message NEW FILE or OLD FILE is printed. OLD FILE indicates that the file already exists and that the current contents of the on line area would replace the text now in the disk storage area. A carriage return confirms that the old file should be written over. If the user does not wish to have the old copy destroyed, he can cancel the WRITE command by typing NO in response to the prompt. He can then enter the WRITE command again, this time specifying a different file name.

The user can also copy text from a file into the on line text area with the READ command. The READ command does not erase the file contents. Also, reading more text into EDITOR does not erase the text already there. The following example demonstrates the use of the READ command.

```
*READ↵
FROM:MYFILE↵      EDITOR prompts for the name of the file.
237  CHARS        EDITOR places a copy of MYFILE in the text area. MYFILE contains 237 characters.
*
```

The user may edit the file read as if the text had been entered at the terminal. All EDITOR commands and control characters can be used after the READ command.

Section 4
EDITING TEXT

This section describes how to use control characters to correct errors, how to use terminal tab stops, how to use the EDIT command, and how to print text lines while in EDITOR.

EDITING WHILE ENTERING TEXT

Errors made while typing text can be corrected using control characters. A control character is an alphabetic character typed while the "CONTROL" key is depressed. Two of the most often used control characters are control A (A^c) and control Q (Q^c). The user can also use control I (I^c) to space text at defined tab stops.

Deleting a Character with Control A

Control A, typed after a character entered in error, deletes the character. When the user types a control A, the system prints an underscore (_) or backarrow (←). The user can then enter a replacement character.

The user can delete more than one character by entering multiple control A's. A control A typed when there are no more characters left on the line is ignored. The following example illustrates the use of control A.

```
*APPEND↵
THIS IS LNA^c_INE ONE↵
A CNFTA^c_A^c_A^c_ONTROL A↵
CAN DELETE CHARACTERS↵
D^c
*
```

The control A is not actually printed at the terminal. It is shown here only to indicate where it was entered on the line.

The EDITOR text area now contains the following text lines.

```
THIS IS LINE ONE
A CONTROL A
CAN DELETE CHARACTERS
```

Deleting a Line with Control Q

Control Q deletes the contents of the current line. If the current line has no contents, control Q deletes the previous line. When a control Q is entered, the system prints a caret (^). EDITOR then returns the carriage and waits for entry of the text that is to take the place of the deleted line. The following example demonstrates the use of control Q.

```
*APPEND ↵
THIS IS LINE ONE ↵
CONTRFL Q DOETYQ^c ^              The user entered a control Q to delete the line.
CONTROL Q DELETES A LINE ↵        The user types the new line.
D^c
*
```

Setting Tabs with Control I

The user can space text on a line with the control I (I^c) tab key. EDITOR sets tab stops at print positions 8, 16, 32, 40, and at every fifth position from print position 40. Each time control I is entered, EDITOR spaces to the next tab stop. The following example demonstrates how to use control I.

```
*APPEND ↵
110I^c GO TO 1 ↵                  Control I does not actually print on the terminal. It is
I^c    PRINT AVR(A) ↵             shown here only to indicate where it was entered on
I^c    END ↵                      the line.
D^c
*
```

EDITING TEXT ALREADY ENTERED

To change a line of text after it has been entered into EDITOR, the user must first address the line. The simplest way to address a line is by its line number. For example, to edit the second line of text in EDITOR, the user types:

```
*2 EDIT ↵
```

EDITOR then prints the second line and returns the carriage. The user can now type a new line or use control characters to edit the line. The user terminates corrections with a carriage return. The new line takes the place of the old line. The user can edit a number of lines by addressing the range of lines to be edited and then entering the EDIT command. EDITOR prints the first line and then returns the carriage. After the user makes changes to the first line, the next line is printed. This continues until all the specified lines have been edited. The following example demonstrates line address editing with the EDIT command.

```
*1,2 EDIT ↵                       A comma separates the line numbers.
THIS IS LINE ONE
THIS IS LINE ONE OF THE TEXT ↵    The user enters the replacement line.
THIS IS LINE TWO                  EDITOR prints the next line.
THIS IS THE END ↵                 The new line is entered.
*                                 Lines 1 and 2 have now been edited.
```

Listing Text Lines

To list a text line, the user addresses the line to be printed and types a slash (/) character. Typing just a slash without specifying a line instructs EDITOR to type the entire contents of the text area, beginning with the first line. A group of lines can be printed by addressing the range and then typing the slash character. Once the line or lines have been printed, EDITOR returns the carriage and prints an asterisk. The following example demonstrates the above concepts.

```
*1,2/                          Commas separate the line numbers.
THIS IS LINE ONE               Lines 1 and 2 are printed.
THIS IS LINE TWO
*2/
THIS IS LINE TWO               Line 2 is printed.
*
```

Editing with Control Characters

After the EDIT command prints the old line and returns the carriage, the user is ready to edit. The control characters described below allow quick and easy changes to a text line. As used below, the *old line* is the line that is to be edited, and the *new line* is the new line of text that is to replace the old line.

Copying a Line with Control D

Control D is used to copy characters in the old line to the new line. Once the user has edited part of a line, he can use a control D to copy the remainder of the line. For example,

```
*1EDIT↵                    The user wishes to edit the first line of text.
THIS BEU LINE ONE
THIS WASD^c LINE ONE       The user retypes the line until no more corrections are needed. He then
*1/                        enters a control D. The remaining part of the old line is copied into the
THIS WAS LINE ONE          new line.
*                          The user displays the corrected line.
```

Characters entered under EDIT replace the corresponding characters in the old line. In the above example, THIS WAS replaces THIS BEU. Control D copies the rest of the old line to the new line.

Typing a carriage return during EDIT terminates the command and deletes the remainder of the old line. For example, if the user had typed a carriage return instead of a control D in the above example, the following would have occurred:

```
*1EDIT↵
THIS BEU LINE ONE
THIS WAS↵                  The user enters a carriage return instead of a control D.
*1/
THIS WAS                   The remaining part of the old line is lost.
*
```

Restarting an Edit with Control Q

Control Q is used to restart an edit begun with the EDIT command. The user can erase the entire contents of the *new line*, retaining the contents of the *old line*. The effect is as if the user had not started a new line. The following example demonstrates the use of control Q.

```
*1EDIT ↵
THIS WAS LINE ONE
THIS IS LINE DDHQ^c ^
THIS IS LINE ONE ↵
*
```

The user enters a control Q to restart the edit.

Copying Characters from the Old Line

The user can copy characters from the *old line* into the *new line* with control Z. To avoid the need to retype correct characters from the old line, the user types control Z followed by the last correct character in the old line. All characters in the old line up to and including the character entered are placed into the new line. The following examples demonstrate how to use control Z.

```
*15EDIT ↵
100 PRINT "GWSOLINE MILEAGE"
Z^cG 100 PRINT "GAD^cSOLINE MILEAGE"
*15/
100 PRINT "GASOLINE MILEAGE"
*
```

The user copies up to and including the G in GWSOLINE with control Z. He then enters an A to take the place of the W. A control D copies the rest of the old line *into the* new line.

```
*1EDIT ↵
C THIS PRGEMRA COMPUTES HE
Z^cRC THIS PROGRAMZ^cS COMPUTES THE ↵
*1/
C THIS PROGRAM COMPUTES THE
*
```

The user copies up to the R in PRGEMRA with control Z. He then types OGRAM. Another control Z copies all characters up to and including the S in COMPUTES. The user then enters the correct characters for the end of the line.

Deleting Characters from the Old Line

The user can delete characters from the *old line* with control S. This control character can be used with any of the other control characters. When control S is entered, a percent sign (%) is printed and the next character in the old line is bypassed. The new line remains unchanged. The following example demonstrates the use of control S.

```
*4EDIT ↵
320 PRINT "TTYPE 2."
Z^c"320 PRINT "S^c%D^cTYPE 2."
*
```

Control S deletes the first T of TTYPE.

Inserting Characters into the New Line

Control E inserts characters into the *new line.* To use control E, the user must first locate the place in the new line for the new characters. He may use any of the control characters discussed to edit or copy characters from the old line to the new line before using control E. When the user enters a control E, the system prints a left angle bracket (<) to indicate the beginning of the insert. The user may enter the new characters at this time. To terminate the insert, the user types control E. The system prints a right angle bracket (>) to indicate the end of the insert. The following example shows how to use control E.

***45EDIT ↵**
888 TYPE "END OF ARG ONE"
Z^cG888 TYPE "END OF ARGE^c<UMENTE^c>D^cONE"
*****

The user copies up to and including the G in ARG. He then types a control E, inserts UMENT, and types control D to copy the remainder of the line.

Addressing a Line

In EDITOR, a text line may be addressed in any of four ways. One method, addressing a line by its line number, has been used in all the preceding examples. Three other methods are described below.

Line Addressing Using Text

The user can address any line in the EDITOR text area by specifying a portion of the text on that line. The user first decides what part of the line is to be used as a search string. It should be unique to the one particular line which is to be edited. He then types the character string enclosed within single quotes, double quotes, or square brackets. EDITOR searches the contents of the text area for the first instance of the character string enclosed within the delimiters. EDITOR types the line in which the text occurs and prints an asterisk. The user can now find out the line number by typing a period followed by an equals sign. For example,

***'the entire'/**
the entire line contents of

The user locates the line in which the character string enclosed by quotes occurs.

***.=52**
*****

The user wants to know the line number. He types a period and an equal sign (.=) without entering a carriage return. EDITOR prints the line number to the right of the equal sign.

Line Addressing Using Line Labels

Line labels are characters that begin in print position 1 and end at the first blank space to the right. A line label cannot begin with a space; only a line with a nonblank character in print position 1 can be addressed. To address a line containing a line label, the user enters a colon (:), the line label, and another colon. A slash character or other command terminates the line. The following example demonstrates this method of line addressing.

```
*:11:/
11      FORMAT(2F11.5)
*
```

The user enters a line label enclosed by colons. EDITOR searches for the line label and prints it on the terminal.

Line Addressing Using a Line Number and Text

Line numbers can be used to begin a search for a text string if the user types the line number, single quote, a unique string, and then a final single quote. The search begins at the line number specified by the user. The following examples demonstrate this method of line addressing. The contents of the EDITOR text area are displayed below.

```
C THIS SUBROUTINE COMPUTES THE SQUARE
C ROOT OF THE SUM OF SQUARES
10      ACCEPT 11,X,Y
11      FORMAT(2F11.5)
        Z=SQRT(X**2+Y**2)
        TYPE 12,X,Y,Z
12      FORMAT(2X$X=$F11.5,2X$Y=$F11.5,2X$Z=$F11.5,/)
        END
```

```
*2'ACC'/
10      ACCEPT 11,X,Y
*4'X**'/
        Z=SQRT(X**2+Y**2)
```

The user begins the search at line 2, knowing that the ACCEPT statement is located soon after line 2.

ERROR MESSAGES

There are several ways in which EDITOR responds to errors due to misspellings or incorrect responses to prompts. EDITOR prints a question mark (?) when a text line is not found. The question mark is also printed when the user misspells a command name, addresses a nonexistent line, or attempts to do something that is impossible, such as to delete the tenth line when there are only eight lines of text.

If the user is typing a command name and makes an error, he can correct the error with control characters. The command can be edited using control characters until the user enters a carriage return and terminates the line. Control Q is frequently used for this purpose. Using a control Q to edit a command line is identical to using it during APPEND. Control Q echoes as a caret (^) on the terminal. EDITOR deletes the line, returns the carriage, and permits another line to be typed.

EDITOR notes errors in typing control characters by ringing the terminal bell. Whenever an inappropriate control character has been entered, EDITOR rings the bell and disregards the control character. The terminal bell rings, for example, if the user enters a control Z followed by a character which does not appear in the rest of the line.

If the user types something other than N or Y in response to an EDITOR prompt, EDITOR prompts WHAT? at the terminal. The user must then respond by typing either N or Y.

Section 5
ADDITIONAL EDITING COMMANDS

The EDIT command and control characters are only a few of the features available in EDITOR. Additional editing commands are described in this section.

In this section, all examples refer to the following sample program located in the user's on line text area.

```
C  THIS PROGRAM CALCULATES
C  THE FLOW OF WATER THROUGH A PIPE
       WRITE 1,4
       READ 0,6,D12,DI1,HT
       AREA1=DI1**2*PI/4
       AREA2=DI2**2*PI/4
       QUANTITY=0.98*AREA1*AREA2*SQRT
      1(2*32.2*HT*13.47/(AREA**2-
      2AREA2**2))
       WRITE 1,5,QUANTITY
5      FORMAT(//$ FLOW IN CFS=$,1X,
      1F17.4/)
4      FORMAT($DIAM OF VENTURI,PIPES$/$
      1AND MANOMETER READING IN FEET
      2$/)
6      FORMAT(3F17.0)
       END
```

THE DELETE COMMAND

The DELETE command removes a line or a range of lines in the EDITOR text area. Any line addressing method can be used to specify line(s) that are to be deleted from the text area. The following example demonstrates one method.

*'<u>1.5'DELETE</u> ↩	*The user deletes the line(s) containing the number 1.5. This number appeared in the tenth line of the text.*
*	

In this example, the user deletes a range of lines.

```
*1,2DELETE↵        The first two lines are deleted.
*/                 The remaining text area contents are displayed with the / command.
      WRITE 1,4
      READ 0,6,DI2,DI1,HT
      AREA1=DI1**2*PI/4
      AREA2=DI2**2*PI/4
      QUANTITY=0.98*AREA1*AREA2*SQRT
     1(2*32.2*HT*13.47/(AREA1**2-
     2AREA**2))
      WRITE 1,5,QUANTITY
5     FORMAT(//$ FLOW IN CFS=$,1X,
     1F17.4/)
4     FORMAT($DIAM OF VENTURI,PIPES$/$
     1AND MANOMETER READING IN FEET
     2$/)
6     FORMAT(3F17.0)
      END
*
```

Each time the DELETE command is issued, EDITOR reassigns text line numbers. EDITOR maintains line numbers as an unbroken series of integer numbers (i.e., 1,2,3,4. . .).

Thus, in the first example, the line containing the '1.5', line 10, was deleted. Once line 10 was deleted, the following line became line 10, line 12 became line 11, and so on. In the second example, the first two lines were deleted. Line number 3 became line number 1, etc.

THE INSERT COMMAND

The INSERT command places new text lines in the text area. Unlike APPEND, which only enters lines at the end of the text area, INSERT places text lines anywhere in the text area. Any of the line address methods except line range addressing can be used to specify where the lines are to be placed. Additional lines are entered in the same manner as with the APPEND command. The user terminates text entry with a control D (D^c).

In addressing lines, the user specifies the line *before* which the new text is to be inserted. For example, to insert text between the first and second line of text, the user specifies line 2.

The following examples demonstrate how to use the INSERT command.

```
*1INSERT↵                              The user inserts two lines before line 1.
C  THIS PROGRAM CALCULATES THE↵
C  FLOW OF OIL THROUGH A PIPE↵
Dc
*'1=D'INSERT↵                          The line "PI=3.14159" is inserted before the line
      PI=3.14159↵                      containing "1=D".
Dc
*/
C  THIS PROGRAM CALCULATES THE         The user displays the text area with inserted lines.
C  FLOW OF OIL THROUGH A PIPE
      WRITE 1,4
```

```
          READ 0.6,DI2,DI1,HT
          PI=3.14159
          AREA1=DI1**2*PI/4
          AREA2=DI2**2*PI/4
          QUANTITY=0.98*AREA1*AREA2*SQRT
         1(2*32.2*HT*13.47/(AREA1**2-
         2AREA2**2))
          WRITE 1,5,QUANTITY
5         FORMAT(//$ FLOW IN CFS=$,1X,
         1F17.4/)
4         FORMAT($DIAM OF VENTURI,PIPES$/$
         1AND MANOMETER READING IN FEET
         2$/)
6         FORMAT(3F17.0)
          END
*
```

THE SUBSTITUTE COMMAND

The SUBSTITUTE command is useful when a user wishes to replace characters or words that occur frequently in the text area. The user first types the command SUBSTITUTE followed by a carriage return. EDITOR then prints a double quote (") on the next line. The user enters the new character or string of characters followed by a control D. EDITOR encloses the new text in double quotes and prints the word FOR followed by another double quote. The user enters the character or string of characters to be replaced, then types control D. EDITOR next prints WAIT? and does not return the carriage. If the user wishes to examine individual cases before making the substitution, he should type Y. After a user-typed Y, EDITOR prints the first instance, and each succeeding instance, prompting OK? after each. Only after the user approves a change by typing Y is the change made. If the user types N in response to the WAIT? prompt, all changes are made without further prompting.

After all SUBSTITUTE changes have been made, EDITOR prints the total number of replacements and returns the carriage. The following example demonstrates the use of the SUBSTITUTE command.

```
*SUBSTITUTE ↵
"HEIGHTD^c" FOR "HTD^c"
WAIT? N
2
*
```

The user substitutes the word "HEIGHT" for the abbreviation "HT". Each string is terminated with a control D.

EDITOR made two substitutions.

In the following example, the user confirms each substitution.

```
*SUBSTITUTE ↵
"DIAD^c" FOR "DID^c"
WAIT? Y
         READ 0,6,DI2,DI1,HEIGHT
OK? Y
2,DI1,HEIGHT
OK? Y
         AREA1=DI1**2*PI/4
```

This line contains two instances where substitutions can be made. Each one must be verified separately.
The user approves the first substitution.
The remainder of the line is printed and EDITOR asks if it is to be changed.

```
OK? Y
        AREA2=DI2**2*PI/4
OK? Y
4       FORMAT($DIAM OF VENTURI,PIPES$/$
OK? N
       1AND MANOMETER READING IN FEET
OK? N
4
*
```

The user does not wish this line changed and replies N.

This line is not changed either. The characters DI appeared in both lines that were not changed.

The EDITOR made four changes in 3 lines.

THE CLEAR COMMAND

If the user desires to make changes to a number of programs or data sets during one terminal session, he must clear the EDITOR text area before entering the next program or data set. This ensures that new text is not added to the old text with the next APPEND or READ command. The user issues the WRITE command when all work is finished, updating the file on disk. Once this is done, he can enter the CLEAR command. EDITOR then prints ALL? and waits for the user to confirm the command. The user should type Y (for yes). Typing N aborts the command. The next example illustrates the use of the CLEAR command.

```
*WRITE ↵
TO:MYFILE ↵
 OLD FILE ↵
485 CHARS
*CLEAR ↵
ALL? Y
*
```

The user issues the WRITE command to save the contents of the updated text area.

The user issues the CLEAR command to erase the lines from the on line text area.

The user is now ready to read or enter more text.

Appendix A
CONTROL CHARACTER AND COMMAND SUMMARY

Control Character	Symbol Printed On Terminal	Function
D^c		1) Terminates APPEND. 2) Copies remainder of old line to the new line; terminates EDIT. 3) Concludes entry of characters using SUBSTITUTE. 4) Terminates INSERT.
A^c	_ (underscore) or ← (backarrow)	Deletes the preceding character.
Q^c	^ (caret)	Deletes the current line. The line can be a command or a line of text entered using APPEND, INSERT, or EDIT.
I^c		Spaces the terminal print head to the next tab stop.
Z^c *n*		Copies up to and including the character *n* from the old line to the new line.
S^c	% (percent sign)	Skips the next character in the old line.
E^c *text* E^c	< > (left and right brackets)	Inserts text into the new line. The first E^c prints <; the second E^c prints >.

Command	Function
APPEND	Enters text into the user's on line text area.
CLEAR	Erases all text from the text area.
DELETE	Deletes a line or range of lines from the text.
EDIT	Prints line(s) and allows line-by-line editing.
INSERT	Inserts additional text immediately before the line addressed in the command.
QUIT	Returns the user to EXECUTIVE command level.
READ	Copies text from a file into the text area. READ does not erase the file.
SUBSTITUTE	Substitutes a character or string of characters typed at the terminal in one or several lines in the text area.
WRITE	Copies the contents of the text area on a file. WRITE does not erase any lines from the text area.