VAX EDT
Reference Manual

Order Number: AA–Z300A–TE

September 1984

This manual serves as a reference source for the EDT interactive text editor. The manual is intended for all users of EDT.

Revision/Update Information:	This manual supersedes the EDT Editor Manual, Order No. AA–J726A–TC.
Software Version:	VAX/VMS Version 4.0

digital equipment corporation
maynard, massachusetts

September 1984

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital

ZK–2303

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

Contents

PART II EDT COMMANDS

FIGURES

Preface

Intended Audience

This manual is intended for all levels of users.

Structure of This Document

This manual is organized into two parts. Part I includes a brief summary of EDT features. It discusses the three modes (keypad, line, nokeypad), the journal facility, startup command files, key definitions, and macros. Part II lists and describes all the available EDT commands by mode.

Associated Documents

To learn how to use the EDT editor, see the *Guide to Text Processing on VAX/VMS*.

Conventions Used in This Document

Convention	Meaning
RET	A symbol with a one- to three-character abbreviation indicates that you press a key on the terminal, for example, RET .
CTRL/x	The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example, CTRL/C, CTRL/Y, CTRL/O.
$ SHOW TIME 05-JUN-1985 11:55:22	Command examples show all output lines or prompting characters that the system prints or displays in black letters. All user-entered commands are shown in red letters. VT100 keys are shown in red, LK201 keys are shown in blue, and VT52 keys are shown in black.

Convention	Meaning
$ TYPE MYFILE.DAT . . .	Vertical series of periods, or ellipsis, mean either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown.
file-spec,...	Horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

Summary of Technical Changes

Technical Changes

- EDT's original line number facility has been replaced by new line numbering features.
- In keypad mode, the VT100 editing keys GOLD + ← and GOLD + → are no longer predefined to SHL. (shift left) and SHR. (shift right) respectively. However, you can add these definitions to your editing session if you need them.
- You must supply an input file specification with the INCLUDE command.
- You must supply an output file specification with the PRINT and WRITE commands. If you include a file specification with your EXIT command but do not specify a file type, EDT uses the file type of the input file.
- When your output file is a disk file and there is a system interruption before EDT has had a chance to complete the processing of one of these commands—EXIT, PRINT, WRITE—the incomplete copy appears in a file with the file type .TMP (temporary). If a writing operation is completed without interruption, the output file is renamed to have the file type you specified.
- When you use the RESEQUENCE command in line mode, EDT displays a message indicating the exact number of lines that were actually renumbered. The message no longer simply indicates the number of lines that you specified in your RESEQUENCE command.
- The CLEAR command deletes both the contents of a buffer and its location. After you use CLEAR, the buffer you deleted no longer appears in the SHOW BUFFER list, except in the case of MAIN or PASTE. With the MAIN and PASTE buffers, CLEAR only deletes the contents of the buffer; the names remain in the SHOW BUFFER list. When you use the CLEAR command to delete the current buffer, EDT transfers you to the MAIN buffer. If you include the CLEAR command in a key definition that locates a buffer and then uses the INCLUDE command to add an external file to that buffer, be sure to put the CLEAR command before the FIND command (for example, DEFINE KEY GOLD F AS "EXT CLEAR BUFF1; FIND =BUFF1; INCLUDE FILE.X"). If the CLEAR command follows the FIND command, EDT will automatically transfer you to the MAIN buffer and include the external file in MAIN. If you have defined a buffer to be a macro, the CLEAR command not only deletes that buffer, but also removes the macro name from the list of valid line mode commands.
- When you edit a file in a directory other than your current directory, EDT now puts the journal file in the current directory, not the input or output directory. You must use the /JOURNAL qualifier to have the journal file created in a different directory.
- EDT now records a CTRL/C in a journal file. You can use the journal /recovery facility to restore an editing session in which you used CTRL/C to abort an EDT command.
- EDT has an upper limit of 32767 for repeat counts, entity counts, and the /DUPLICATE line mode qualifier.

- When EDT starts up in change mode on a VT100-type terminal, EDT sets your screen width to the width recorded in the operating system. When you exit from EDT, EDT returns the screen to this width.
- EDT now contains much more error checking. Consequently, illegal commands now produce error messages instead of undocumented effects. The most visible result of this change is in the DEFINE KEY command. You can no longer define keys by using keypad key numbers greater than 21; you must use the syntax described in this manual.
- EDT is not supported on IAS systems.

New Features

- The line numbering restrictions for EDT have been changed. In most cases, the maximum number of lines is now subject to disk space restrictions, rather than EDT limits. For EDT, the new maximum line number is 2,814,749,767.
- There are 12 new nokeypad commands in EDT. Many of these commands have been added so that EDT can emulate the DECmail and WPS (word processing) environments. The new commands follow:

  BELL (sound the terminal bell)
  CHGL (change case lower)
  CHGU (change case upper)
  CLSS (clear search string)
  DATE (insert date)
  DESEL (deactivate select range)
  DLWC (default lowercase)
  DMOV (default move)
  DUPC (default uppercase)
  SSEL (search and select)
  TGSEL (toggle select)
  XLATE (pass parameter to a running program)

  The following new SET and SHOW commands have been added:

  SET [NO]AUTOREPEAT
  SET COMMAND
  SET [NO]FNF
  SET HELP
  SET PARAGRAPH [NO]WPS
  SET PROMPT
  SET [NO]REPEAT
  SET SEARCH WPS, CASE INSENSITIVE, DIACRITICAL INSENSITIVE
  SET [NO]SUMMARY
  SET TERMINAL [NO]SCROLL, [NO]EIGHTBIT, [NO]EDIT
  SET TEXT END/PAGE
  SET WORD [NO]DELIMITER

  SHOW AUTOREPEAT
  SHOW COMMAND
  SHOW FILES
  SHOW FNF
  SHOW HELP
  SHOW KEYPAD
  SHOW LINES
  SHOW MODE
  SHOW NUMBERS

SHOW PARAGRAPH
SHOW PROMPT
SHOW QUIET
SHOW REPEAT
SHOW SUMMARY
SHOW TAB
SHOW TEXT END/PAGE
SHOW TRUNCATE
SHOW VERIFY
SHOW WORD
SHOW WRAP

- You can redefine both the GOLD keypad key and the GOLD/GOLD key sequence with the line mode DEFINE KEY command. Use the number 20 to designate the GOLD key.
- EDT can now be called on VAX/VMS systems by programs. The calling program can be written in any VAX–11 language that generates calls based on the VAX–11 calling standard. See the *VAX/VMS Utility Routines Reference Manual* manual for information about callable EDT.
- EDT now supports the DEC Multinational Character Set.
- EDT can be used with terminals that have LK201 keyboards.

PART I

1 Introduction to EDT

1.1 Summary of EDT

EDT is an interactive text editor that has three distinct editing modes: keypad, line, and nokeypad. Both the keypad and nokeypad modes are screen editors for use on VT100-type terminals, VT52 terminals, and VT200 Series terminals. Line mode can be used on any type of terminal—hardcopy or screen.

Note: Please note that 132 column mode is not supported on VT100 terminals without AVO.

With the EDT editor you can create and edit almost all types of text files. When you are editing a file, you can add or delete text, move or copy text from one place to another, save or discard your editing work.

To invoke the EDT editor, you must use the system command EDIT. (If EDT is not the standard editor on your operating system, you must include the /EDT qualifier with the EDIT command.)

Include the name of the file you want to edit on the EDIT command line.

```
$ EDIT LETTER.DAT
```

If a file named LETTER.DAT exists in the current directory, EDT puts a copy of that file into the MAIN buffer in your EDT session. Then EDT displays the first line from the file on your screen or paper, followed by the line mode asterisk prompt (*). You are now ready to begin your EDT session.

```
$ EDIT LETTER.DAT
    1       June 4, 1985
*
```

If you are using EDT to create a new file called LETTER.DAT, your session starts off like this:

```
$ EDIT LETTER.DAT
Input file does not exist
[EOB]
*
```

The message **Input file does not exist** tells you that no text was copied to the MAIN buffer. The end of buffer mark ([EOB]) is printed in place of the first line, since there is no text to display. The next step is to insert some text in the buffer, using one of the three editing modes.

When you start your editing session, EDT's default mode is line mode. To shift to keypad mode, use the line mode CHANGE command. To shift back to line mode from keypad mode, use CTRL/Z.

To go from line mode to nokeypad mode, you must first give the SET NOKEYPAD command and then the CHANGE command. The nokeypad EX command shifts EDT from nokeypad editing back to line mode.

You can use EDT's online HELP facility any time during your editing session. The line mode HELP command supplies general information on EDT as well as details on line mode and nokeypad mode commands. The HELP command by itself provides information on using the HELP facility and a list of topics that you can get help on. For help on a specific topic, type the HELP command followed by the topic name, for example, HELP EXIT.

When you are in keypad mode, press the HELP key (PF2 on VT100-type terminals, red on VT52s). EDT displays a diagram of the keypad, a list of other keypad editing keys, and tells you to press the key you want help on.

You must use the line mode HELP command to get information on nokeypad editing. The relevant topics are HELP CHANGE, HELP CHANGE SCREEN, HELP CHANGE ENTITIES, and HELP CHANGE SUBCOMMANDS. Nokeypad command descriptions are found under the SUBCOMMANDS topic. Thus, if you want information on the BELL command, you must type HELP CHANGE SUBCOMMANDS BELL.

1.1.1 Keypad Mode

This example uses keypad mode to insert the sample text.

```
$ EDIT LETTER.DAT
*CHANGE
      ----------------------------
      |
      | [EOB]
      |

      ----------------------------
      |
      |  June 18, 1985
      |  Mr. Charles R. Thurman
      |  Director of Marketing
      |  Energy Systems Inc.
      |      .
      |      .
      |      .
      |

CTRL/Z

*EXIT
DISK$USER:[SMITH]LETTER.DAT 50 lines
```

1.1.2 Line Mode

This example uses line mode to insert the sample text.

```
$ EDIT/EDT LETTER.DAT
*INSERT
            June 18, 1985
            Mr. Charles R. Thurman
            Director of Marketing
            Energy Systems Inc.
               .
               .
               .
^Z                    (Press CTRL/Z)
*EXIT
DISK$USER:[SMITH]LETTER.DAT 50 lines
```

1.1.3 Nokeypad Mode

This example uses nokeypad mode to insert the sample text.

```
$ EDIT/EDT LETTER.DAT
*SET NOKEYPAD
*CHANGE
      ------------------------------
      |
      |  [EOB]
I  RET
      ------------------------------
      |
      |  June 18, 1985
      |  Mr. Charles R. Thurman
      |  Director of Marketing
      |  Energy Systems Inc.
      |     .
      |     .
      |     .

CTRL/Z

EX
*EXIT
DISK$USER:[SMITH]LETTER.DAT 50 lines
```

1.2 Keypad Mode

Keypad mode uses the numeric keypad located to the right of the main keyboard on VT100-type and VT52 terminals. On terminals with LK201 keyboards, there are two keypads to the right of the main keyboard. The numeric keypad at the right edge of the keyboard corresponds to the VT100 keypad.

The arrow keys on VT100-type terminals are considered to be part of the keypad. On terminals with LK201 keyboards, the arrow keys are located at the bottom of the second keypad.

Four keyboard keys have special keypad editing functions: BACKSPACE, DELETE, LINEFEED, and TAB. On the LK201 keyboard, the DELETE key is labelled <X] . The BACKSPACE and LINEFEED keys are located in the row of function keys across the top of the keyboard. (BACKSPACE is key F12; LINEFEED is key F13.)

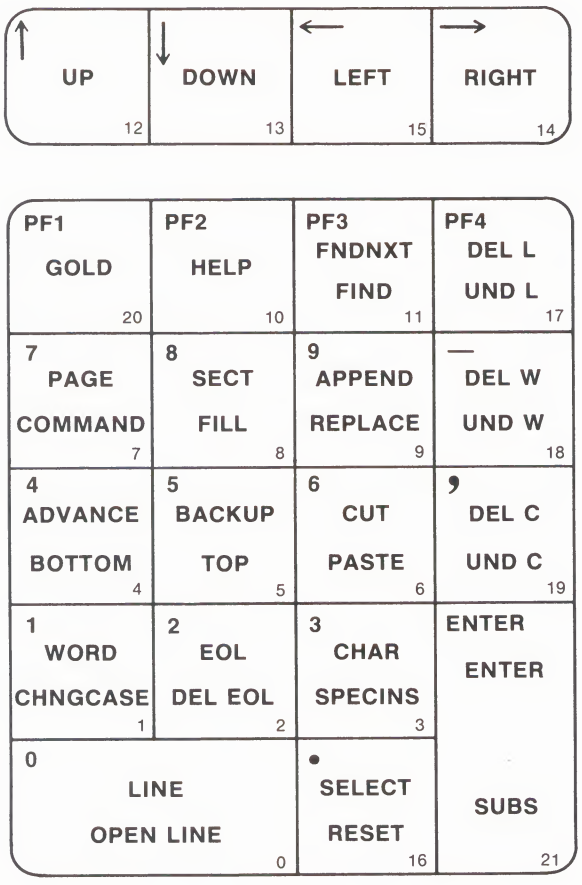
In addition, EDT has preset functions for several control key sequences. (The letters marked with asterisks have preset functions with GOLD as well as with CTRL.)

A*	I	R*
C	J	T*
D*	K	U*
E*	L	W*
F (VT52 only)	M	Z*
H		

Most keypad keys have two editing functions associated with them. To use the upper (primary) function (for example, FNDNXT) simply press the indicated key. To use the lower (alternate) function (for example, FIND), first press the GOLD key (PF1 on VT100; blue on VT52) and then the keypad

key. Figure 1–1 shows the keypad editing keys for VT100-type terminals. Figure 1–2 shows the keypad editing keys for VT52 terminals.

Figure 1–1 Keypad Editing Keys—VT100-Type Terminals



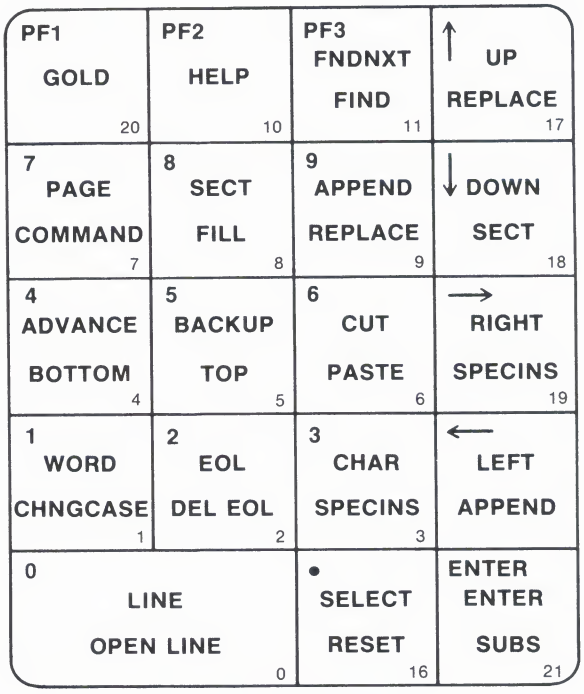
VT100

ZK-1767-84

For terminals that have the LK201 keyboard, EDT has preset keypad mode definitions for the remaining keys on the second keypad as well as for some function keys on the top row of the keyboard. Figure 1–3 shows the editing keypad keys and lists the other preset function keys. The arrow keys are located at the bottom of the editing keypad.

F12 = BACKSPACE
F13 = LINEFEED
<X = DELETE
TAB = TAB

Figure 1–2 Keypad Editing Keys—VT52 Terminals



VT52

ZK-1768-84

1.3 Line Mode

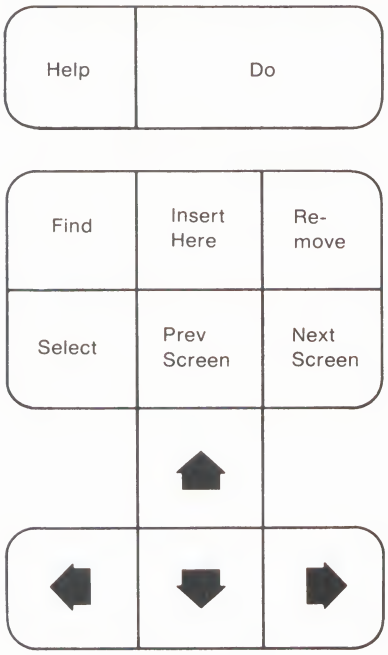
Line editing can be used on any interactive terminal. This mode focuses on the line as the unit of text. Whenever you see the line mode asterisk prompt (*), you can type a line mode command. The underlined letters in the line mode syntax statements that follow indicate the minimum abbreviation for both command words and qualifiers.

Line mode commands use qualifiers and specifiers in addition to command words. Qualifiers, which modify the way EDT processes the command, are always optional. You must precede a qualifier with a slash (for example, /QUERY). The line mode qualifiers are /BRIEF, /DUPLICATE, /NOTYPE, /QUERY, /SAVE, /SEQUENCE, and /STAY.

Specifiers tell EDT which part of the text to act on. Optional specifiers are enclosed in square brackets (for example, [=buffer]). The main specifier is **range**, which references the line or lines affected by the command. See the range specifier description for more information.

The SET and SHOW commands are line mode commands. SET commands modify the way EDT behaves during your editing session. They have no effect on the text you are editing. You can use SET commands to customize your editing session. SHOW commands tell which SET commands are in

Figure 1–3 Additional Keypad Editing Keys—LK201 Keyboards



VT200 Series

effect. Four SHOW commands (SHOW BUFFER, SHOW FILES, SHOW KEY, and SHOW VERSION) have no corresponding SET commands.

1.4 Nokeypad Mode

Nokeypad mode is a screen editor for use with VT100-type and VT52 terminals. Text appears on the upper lines of the screen. As you type commands, they are displayed at the bottom of the screen. When you press RETURN, EDT processes the commands.

Nokeypad commands are the basis for keypad mode key definitions. Nokeypad commands consist of English words and abbreviations. A number of commands take the **entity** specifier to determine which part of the text will be affected by the command. For more information see the entity specifier description.

Nokeypad commands cannot have spaces. For example, to delete two paragraphs and put the text in a buffer named EXTRA, type CUT2PAR=EXTRA. You can put several nokeypad commands on the same line. Spaces between commands are allowed, but not required. You can repeat a series of commands by preceding the commands with the repeat count and enclosing them in parentheses 3(V D+EL).

1.5 The Journal Facility

EDT's journal facility keeps track of each keystroke you make during your EDT session in a special file called the journal file. When you finish your editing session by typing either EXIT or QUIT, EDT normally discards the journal file. If your editing session ends abruptly because of a system interruption, the journal file is saved, even though your editing work has been lost. Using the journal file, EDT can restore almost all of your editing work. Sometimes the last few commands you typed or the last few keypad editing keys you pressed are not recorded in the journal file at the time the interruption occurs.

The journal file is normally stored in the current directory and has **.JOU** for the file type. The file name is the same as the name of the file you were editing when the interruption took place. For example, if you are editing the file LETTER.DAT, the journal file name is LETTER.JOU. (It is possible to instruct EDT to use a different name or file type for the journal file. You do this when you type the EDIT/EDT command line to start your session. Information on including a journal file name in the EDIT/EDT command line appears in the *VAX/VMS DCL Dictionary*.

Include the /RECOVER qualifier in your EDIT/EDT command line in order to have EDT use the journal file to restore your editing work. Even when you have used EDT to create a new file, you can use the /RECOVER qualifier to have EDT restore your editing work after a system interruption.

```
$ EDIT/EDT/RECOVER LETTER.DAT
```

As soon as EDT has processed all the command information and keystrokes stored in the journal file, it continues to use that journal file to store the new information that results from any further editing work you do during the recovery session. If you are able to end your editing session with EXIT or QUIT, EDT discards the journal file from the directory. However, if you find journal files in your directory that you do not plan to use, simply delete them.

You can use the /SAVE qualifier with the line mode EXIT or QUIT command to save the journal file even when there is no system interruption.

1.6 Startup Command Files

Startup command files establish various settis and key definitions at the start of your EDT session. These files can contain only line mode commands. SET and DEFINE KEY commands are the ones most frequently found in startup command files.

Generally, EDT reads a system-wide startup command file at the beginning of your editing session. If no system-wide startup command file exists on your system, EDT looks for a file named EDTINI.EDT in your default directory and processes the commands in that file.

If you want EDT to use a startup command file with another name, you must include that command file specification in the EDIT/EDT command line. See the *VAX/VMS DCL Dictionary* for more information.

You can use EDT to create a startup command file. There are very few restrictions to organizing such a file. If you need to use the carriage return character (CTRL/M) in a SET or DEFINE KEY command, you must enter it from change mode.

Here are some typical commands that you might put in a startup command file:

```
SET QUIET
SET WRAP 60
SET SEARCH BOUNDED
SET TAB 5
SET MODE CHANGE
DEFINE KEY GOLD P AS "PAR."
```

You can include all the necessary commands to set up an EDT macro in your startup command file. The sample macro, called EXACT, resets the search parameter to **exact** when you type the macro name as a line mode command. The following commands can be put in the startup command file to create the EXACT macro:

```
DEFINE MACRO EXACT
FIND =EXACT
INSERT; SET SEARCH EXACT
FIND =MAIN
```

1.7 Defining Keys

You can redefine or relocate any preset keypad editing key. You can also create definitions for editing keys that have none. These key definitions enable you to perform a variety of editing tasks in keypad mode.

The following table shows the five types of keys or key sequences that can be defined as well as any specific keys you cannot define:

Key/Key Sequence	Cannot Define
Keypad keys	
Control keys GOLD control key	C, O, P, Q, S, X, Y, and [
GOLD keyboard key	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, - (When you use !, %, ', or " in the DEFINE KEY command, you must enclose them in quotation marks—for example: "!".)
DELETE GOLD DELETE	
FUNCTION key GOLD FUNCTION key	LK201 keyboard only. F1, F2, F3, F4, and F5 with or without GOLD.

In EDT the following special keyboard keys and control keys are linked. When you redefine one, you have redefined the other.

VT100/VT52				LK201 [1]
BACKSPACE	<--->	CTRL/H	<--->	F12
TAB	<--->	CTRL/I	<--->	Tab
LINEFEED	<--->	CTRL/J	<--->	F13
RETURN	<--->	CTRL/M	<--->	Return

[1] The LK201 F12 and F13 keys are linked to CTRL/H and CTRL/J when the terminal is in VT100 compatibility mode. When the terminal is operating in VT200 mode, F12 and F13 are not linked to CTRL/H and CTRL/J. They have the same preset definitions, but can be defined independently, just as CTRL/A and GOLD/A can be defined independently.

Keypad key definitions use nokeypad command syntax. Nokeypad mode enables you to put several commands on a single line. Thus, you can create key definitions to do several commands in succession. If you want EDT to process the command as soon as you press the key or key sequence, place a period at the end of the definition. This corresponds to pressing the ENTER key in keypad mode.

You can use the SHOW KEY command to find the existing definitions for all EDT editing keys. However, you must use special terms to tell EDT which key or key sequence you want information about. Figure 1–4 gives the special keypad key numbers to use with the SHOW KEY command for VT100 and VT52 terminals. For control keys, type SHOW KEY CONTROL character (for example, **SHOW KEY CONTROL A**). To find the definition of a GOLD key sequence, type SHOW KEY GOLD keyboard key or SHOW KEY GOLD CONTROL keyboard key (for example, **SHOW KEY GOLD *** or **SHOW KEY GOLD CONTROL]**). To see the definition for the DELETE key, spell out the word (for example, **SHOW KEY GOLD DELETE**). To see the definition for a key on the second keypad or in the function key row of the LK201 keyboard, spell out the word FUNCTION and use the EDT FUNCTION number designation, not the F number that appears on the keyboard (for example, **SHOW KEY FUNCTION 24** for the F12 key).

By enclosing all parts of the definition except the ending period in parentheses, you can use the keypad GOLD repeat feature to process the command(s) several times in succession.

There are two ways to define/redefine keypad editing keys. From keypad mode you can use the CTRL/K function. The line mode DEFINE key command can be accessed from any mode.

Using CTRL/K, you can press the key(s) you are defining. Whenever you want to include a preset keypad editing key in your definition, you can press that key as part of the definition process. EDT prompts you for both the key you are defining and the definition. For example, suppose you want to define CTRL/P to move a paragraph at a time:

CTRL + K

```
Press the key you wish to define
```

CTRL + P

```
Now enter the definition terminated by ENTER
PAR.
```

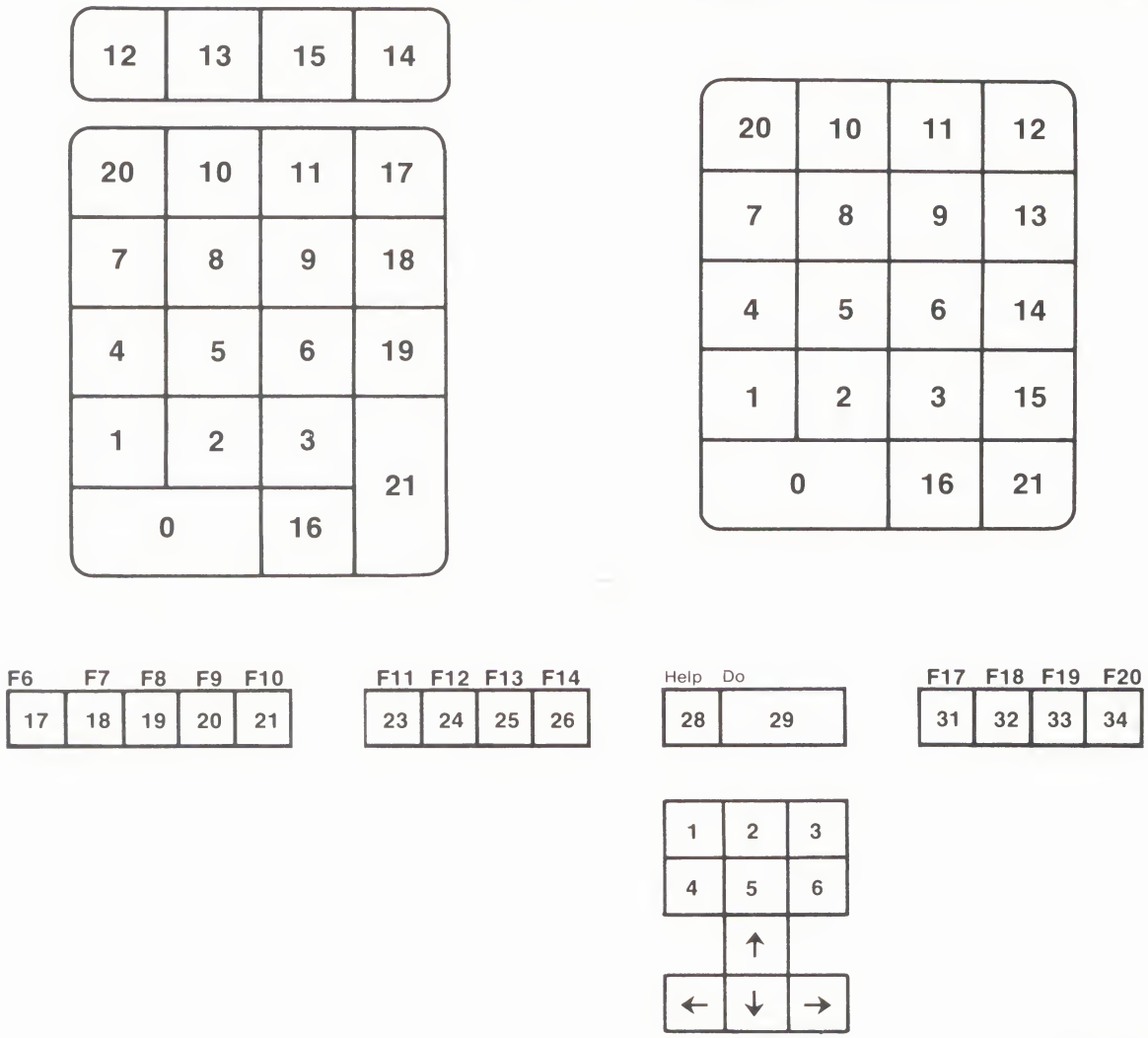
ENTER
SUBS

The DEFINE KEY command in line mode uses the typed out version of the key name and requires that the definition be enclosed in quotation marks. The same definition using line mode is:

```
DEFINE KEY CONTROL P AS "PAR."
```

To redefine keypad keys with the DEFINE KEY command, you must know their special EDT numbers. Figure 1–4 shows these numbers for both VT100-type and VT52 terminals, as well as the FUNCTION numbers for the LK201 keyboard.

Figure 1–4 Keypad Key Numbers



ZK-1770-84

To redefine the PAGE function to be "(-W)." , type the following:

```
DEFINE KEY 7 AS "(-W)."
```

To redefine GOLD Left Arrow to be "SHL." , type the following:

```
DEFINE KEY GOLD 15 AS "SHL."
```

Here are some sample DEFINE KEY commands:

```
DEFINE KEY GOLD CONTROL B AS "CHGUW."
DEFINE KEY GOLD DELETE AS "-DW."
DEFINE KEY CONTROL V AS "(+V D-CI |^Z)."
DEFINE KEY CONTROL N AS "?'Nokeypad Command: '."
DEFINE KEY GOLD 21 AS "S^@?'Find: '^@?' Substitute: '^@."
DEFINE KEY FUNCTION 17 AS "IWashington, DC^Z."
```

1.8 EDT Macros

EDT macros allow you to extend the line mode command language. A macro is a group of one or more line mode commands that is processed when you type the macro name as a line mode command word. For instance, you can create a macro called **XT** to be the command SET SEARCH EXACT, and another macro called **GEN** to be the command SET SEARCH GENERAL.

To create a macro, use a line mode command to move to a new buffer. That buffer must have the same name as the macro. Using EDT, enter the line mode command or commands that you want to be processed when you type the macro name. Then issue the command:

```
DEFINE MACRO macro-name
```

EDT now adds the macro name to its list of valid line mode commands. You are ready to use that command any time during your editing session. If you plan to use the macro in repeated EDT sessions, you can put it in a startup command file or simply store it in an external file to be included in later EDT sessions.

If you put a DEFINE MACRO command in a startup command file, you must include commands to enter the macro text into a buffer with the same name as the macro. If you store a macro in an external file, use the INCLUDE command to put the macro in a buffer with the same name and issue the DEFINE MACRO command to establish the name as a line mode command.

This example creates a macro called SINCY, which inserts the closing of a letter above the current line when you type the newly created line mode command SINCY.

```
*FIND =SINCY
*INSERT
            INSERT ;Sincerely yours,
            INSERT ;
            INSERT ;
            INSERT ;
            INSERT ;
            INSERT ;
            INSERT ;James T. Cortland
            INSERT ;President

CTRL/Z

*DEFINE MACRO SINCY
*FIND =MAIN
```

To close a letter from Mr. Cortland, type SINCY as a line mode command.

```
*SINCY
*TYPE "sincerely" THRU END
  132       Sincerely yours,
  133
  134
  135
  136
  137
  138       James T. Cortland
  139       President
[EOB]
```

PART II EDT Commands

This section contains descriptions of the EDT commands, qualifiers, and specifiers. Entries for each command, qualifier, and specifier described in the rest of the section are arranged by editing mode.

At the top outer corner of the page is the entry name and below that the mode from which the entry is used. For line and nokeypad commands, the next item is the syntax statement. Keypad functions show the keypad designations and the keypad keys used for both VT100 and VT52 terminals. The VT100 keys are shown first, in red; the VT52 keys are shown in blue. The additional preset keypad mode function keys that are available on the LK201 keyboard are shown where appropriate.

There is a description of the purpose and use of each item. Information on special restrictions, if any, is also included. Related commands, qualifiers, and specifiers are printed in bold to indicate that these items are themselves entries in this section.

Where appropriate, one or more examples are provided. Occasionally, in keypad mode, you press different keys to access the function on VT100 and VT52 terminals. In these instances VT100 user input appears in red; VT52 user input appears in blue.

The last element for many entries is Related Commands, which lists commands or keypad functions that are similar to that entry. Related commands are limited to those that perform almost the identical function in a different mode. If no related command is listed, it means that none exists. For example, there are no related commands for any of the SET or SHOW commands.

ADVANCE Function—Keypad

KEY SEQUENCE

VT100: [ADVANCE BOTTOM] [4]

VT52: [ADVANCE BOTTOM] [4]

DESCRIPTION

Pressing ADVANCE sets the direction for subsequent editing work to forward (to the right of the cursor and down toward the end of the buffer). **ADV.** is the nokeypad definition for ADVANCE.

ADVANCE is the default direction and remains in effect until you press **BACKUP**.

You can also use the ADVANCE key to set the direction of and process the **FIND** function. Press the GOLD and FIND keys, enter the string you want to locate, then press ADVANCE to move the cursor forward to find the string.

EXAMPLES

example 1:

Sets EDT's direction to forward and moves the cursor three words to the right.

```
In regard to your memo of January 15th,
```

[ADVANCE BOTTOM] + [WORD CHNGCASE] + [WORD CHNGCASE] + [WORD CHNGCASE]

```
In regard to your memo of January 15th,
```

example 2:

Uses ADVANCE to be sure that EDT looks for the string **town** to the right of the cursor.

```
Acton, MA
Bedford, MA
Charlestown, MA
```

[GOLD] + [FNDNXT FIND]

```
Search for:  town
```

[ADVANCE BOTTOM]

```
Acton, MA
Bedford, MA
Charlestown, MA
```

related commands

— Nokeypad—ADV (advance)

APPEND Function—Keypad

KEY SEQUENCE

VT100: [APPEND REPLACE] [9]

VT52: [GOLD] + [LEFT APPEND] [BLUE COLOR] + [-]

DESCRIPTION

Pressing APPEND deletes the select range from the current buffer and adds it to the end of the PASTE buffer. The previous contents of the PASTE buffer are not deleted. **APPENDSR.** is the nokeypad definition for APPEND.

EXAMPLE

Alphabetizes the list of names using SELECT, CUT, APPEND, and PASTE.

```
Joe Spitzer
Trudi Schutz
Becky Santerre
Kathy Waldbauer
```

VT100: [SELECT RESET] + [LINE OPENLINE] + [CUT PASTE]

(Move the cursor to the **T** in **Trudi**.)

VT100: [SELECT RESET] + [LINE OPENLINE] + [APPEND REPLACE]

(Move the cursor to the **J** in **Joe**.)

VT100: [GOLD] + [CUT PASTE]

VT52: [SELECT RESET] + [LINE OPENLINE] + [CUT PASTE]

(Move the cursor to the **T** in **Trudi**.)

VT52: [SELECT RESET] + [LINE OPENLINE] + [GOLD] + [LEFT APPEND]

(Move the cursor to the **J** in **Joe**.)

VT52: [GOLD] + [CUT PASTE]

```
Becky Santerre
Trudi Schutz
Joe Spitzer
Kathy Waldbauer
```

related commands

– Nokeypad—APPEND

BACKSPACE Function CTRL/H—Keypad

KEY SEQUENCE

VT100:	BACKSPACE	BACK SPACE
VT52:	BACKSPACE	BACK SPACE
LK201:	BACKSPACE	F12
All:	CTRL/H	CTRL + H

DESCRIPTION

Pressing BACKSPACE causes the cursor to move to the beginning of the current line. If the cursor is already at the beginning of a line, pressing BACKSPACE moves it to the beginning of the previous line. **BL.** is the nokeypad definition for BACKSPACE.

The BACKSPACE key and CTRL/H always have the same preset function in EDT. When you redefine the BACKSPACE key, you redefine CTRL/H (except for terminals with LK201 keyboards when they are operating in VT200 mode). To redefine the BACKSPACE key using the line mode DEFINE KEY command, type **DEFINE KEY** CONTROL H. To find the definition of the BACKSPACE key, type **SHOW KEY** CONTROL H. For terminals with LK201 keyboards, use DEFINE KEY FUNCTION 24 and SHOW KEY FUNCTION 24 for the F12 key.

EXAMPLE

Moves the cursor to the beginning of the third line and then to the beginning of the second line.

```
All employees wishing to leave early on
December 24th must obtain permission from
their department supervisors._
```

BACK SPACE + BACK SPACE

```
All employees wishing to leave early on
December 24th must obtain permission from
their department supervisors.
```

related commands

— Nokeypad—BL (beginning of line)

BACKUP Function—Keypad

KEY SEQUENCE

VT100: BACKUP TOP 5

VT52: BACKUP TOP 5

DESCRIPTION

Pressing BACKUP sets the direction for subsequent editing work to backward (to the left of the cursor toward the beginning of the buffer). **BACK.** is the nokeypad definition for BACKUP.

You can use the BACKUP key to set the direction of and process the **FIND** function. Press the GOLD and FIND keys, enter the string you want to locate, and then press BACKUP to move the cursor backward to find the string.

To change EDT's direction to forward, use **ADVANCE**. The **RESET** function also sets EDT's direction to forward.

EXAMPLES

example 1:

Sets the direction to backward and moves the cursor three words to the left.

```
In regard to your memo of January 15th,
```

BACKUP TOP + WORD CHNGCASE + WORD CHNGCASE + WORD CHNGCASE

```
In regard to your memo of January 15th,
```

example 2:

Ensures that EDT searches for the string **Ana** toward the top of the buffer.

```
Anaheim, CA
Bakersfield, CA
Claremont, CA
```

GOLD + FNDNXT FIND

```
Search for:  Ana
```

BACKUP TOP

```
Anaheim, CA
Bakersfield, CA
Claremont, CA
```

related commands

— Nokeypad—BACK (backup)

BOTTOM Function—Keypad

KEY SEQUENCE

VT100: GOLD + ADVANCE BOTTOM PF1 + 4

VT52: GOLD + ADVANCE BOTTOM BLUE COLOR + 4

DESCRIPTION

Pressing BOTTOM moves the cursor to the end of the buffer, after the last character position in the buffer. The cursor is now positioned at the end of buffer ([EOB]) mark. **ER.** is the nokeypad definition for BOTTOM.

EXAMPLE

Moves the cursor to the bottom of the buffer.

```
Dear Sir:
    .
    .
    .
Sincerely yours,
.BLANK 5
David Foster
Manager
[EOB]
```

GOLD + ADVANCE BOTTOM

```
Dear Sir:
    .
    .
    .
Sincerely yours,
.BLANK 5
David Foster
Manager
[EOB]
```

related commands

- Line—TYPE END
- Nokeypad—ER (end of range)

CHAR (Character) Function—Keypad

KEY SEQUENCE

VT100: CHAR SPECINS 3

DESCRIPTION

Pressing CHAR (character) moves the cursor one character in the current direction (forward or backward, depending on whether **ADVANCE** or **BACKUP** is in effect). **C.** is the nokeypad definition for CHAR.

This key is not on the VT52 keypad. However, the left and right arrow keys can be used to move the cursor one character position.

EXAMPLE

Sets the direction to backward and moves the cursor three characters to the left.

```
This product performs perfectly.
```

BACKUP TOP + CHAR SPECINS + CHAR SPECINS + CHAR SPECINS

```
This product performs perfectly.
```

related commands

— Nokeypad—C (character)

CHNGCASE (Change Case) Function—Keypad

KEY SEQUENCE

VT100: GOLD + WORD CHNGCASE PF1 + 1

VT52: GOLD + WORD CHNGCASE BLUE COLOR + 1

DESCRIPTION

Pressing CHNGCASE (change case) changes the case of letters in your text. Uppercase letters become lowercase; lowercase letters become uppercase. The number of letters affected by this function depends on several factors: active select range, cursor location, **SET SEARCH** parameter, and repeat count. The following chart shows what happens when you use CHNGCASE under various conditions:

Conditions	Results
SELECT RANGE ACTIVE	Changes the case of every letter in the select range. All lowercase letters become uppercase; all uppercase letters become lowercase.
NO SELECT RANGE ACTIVE	
1a. SET SEARCH BEGIN in effect. Cursor on first character of current search string. Repeat count = 0 or 1.	Changes the case of every letter in the search string. All lowercase letters become uppercase; all upperletters become lowercase. Move)the cursor to the character to the right of the search string.
1b. SET SEARCH END in effect. Cursor to the right of current search string. Repeat count = 0 or 1.	Changes the case of every letter in the search string. All lowercase letters become uppercase; all uppercase letters become lowercase. Moves the cursor to the first character of the search string.
2a. Current direction is forward. *Cursor not at active end of current search string. Repeat count = 0 or 1.	Changes the case of the letter that the cursor is on. Moves the cursor one column to the right.
2b. Current direction is backward. *Cursor not at active end of current search string. Repeat count = 0 or 1.	Changes the case of the letter to the left of the cursor. The cursor remains on the altered letter.
3a. Repeat count greater than 1. Current direction is forward.	Changes the case of the number of letters given in the repeat count. Moves the cursor to the right of the last altered character.
3b. Repeat count greater than 1. Current direction is backward.	Changes the case of the number of letters given in the repeat count. Moves the cursor to the leftmost altered letter.

```
* Active end of select range:
     SET SEARCH BEGIN          Cursor on first character of current
                               search string.
     SET SEARCH END            Cursor one character to the right of
                               current search string.
```

CHGCSR. is the nokeypad definition of CHNGCASE.

EXAMPLES

example 1:

First changes the case of the **M** in **Maynard** to lowercase. Then changes the case of the entire word to uppercase.

```
Maynard, MA
```

GOLD + WORD CHNGCASE

```
maynard, MA
```

(Move the cursor back to the **m** in **maynard**.)

SELECT RESET + WORD CHNGCASE + GOLD + WORD CHNGCASE

```
MAYNARD, MA
```

example 2:

Using FIND and FNDNXT, locates and changes the case of the string **back** in lines 2 and 3.

```
BACK
backspace
backup
```

GOLD + FNDNXT FIND

```
Search for:  back
```

ENTER SUBS

GOLD + WORD CHNGCASE + FNDNXT FIND + GOLD + WORD CHNGCASE

```
BACK
BACKspace
BACKup
```

related commands

- Nokeypad—CHGC (change case)
- CHGL (change case lower)
- CHGU (change case upper)

COMMAND Function—Keypad

KEY SEQUENCE

VT100: GOLD + PAGE COMMAND PF1 + 7

VT52: GOLD + PAGE COMMAND BLUE COLOR + 7

DESCRIPTION

Pressing COMMAND enables you to enter a line mode command while EDT is still in keypad mode. When you press GOLD and COMMAND, EDT prompts you with **Command:** at the bottom of the screen. Type the line mode command you want to use, for example **EXIT**, then press the **ENTER** key on the keypad to send the command to EDT.

Use **CTRL/Z** to shift from keypad to line editing if you want to issue several line mode commands in a row.

You can issue two or more line mode commands on the same line by separating the commands with semicolons (;). If you want to put nokeypad commands after the line mode command, use the **CHANGE ;nokeypad-command(s)** form. You can use EDT macros with the COMMAND function just as you would any other line mode command.

EXT ?'Command: '. is the nokeypad definition for COMMAND.

EXAMPLES

example 1:

Uses the line mode SHOW BUFFER command while still in keypad mode.

GOLD + PAGE COMMAND

```
Command:  SHOW BUFFER
```

ENTER SUBS

```
 BUF1   2       lines
=BUF2   5       lines
 MAIN   237     lines
 PASTE  1       lines
Press return to continue
```

example 2:

Uses the line mode WRITE command to put a copy of the remaining lines in the current buffer into an external file called SAVE.DAT. EDT remains in keypad mode.

GOLD + PAGE COMMAND

```
Command:  WRITE SAVE.DAT REST
```


```
Command:  WRITE SAVE.DAT REST
DISK$USER:[SMITH]SAVE.DAT 36 lines
```

related commands

— Nokeypad—EXT (extend)

CTRL/A (Control A) Function—Keypad GOLD/A

KEY SEQUENCE

VT100:	CTRL/A	CTRL + A
VT52:	CTRL/A	CTRL + A
VT100:	GOLD/A	GOLD + A
VT52:	GOLD/A	GOLD + A

DESCRIPTION

Pressing CTRL/A establishes the tab position at the present cursor position and resets the indentation level count to be the quotient of the cursor position divided by the **SET TAB** value. To use CTRL/A, the current cursor position must be a multiple of the SET TAB value. For example, if the SET TAB value is 5, you can use CTRL/A for cursor locations at positions 5, 10, 15, 20, and so on. If the cursor is at some other column, for example 13, and you press CTRL/A, EDT prints an error message.

CTRL/A does not move text. You must use the **TAB** function to indent a line. CTRL/A works only if SET TAB is in effect. EDT's default is SET NOTAB. **TC.** is the nokeypad definition for both CTRL/A and GOLD/A.

EXAMPLE

Using a SET TAB value of 5, sets the indentation level count to 3 to move the first line of text 15 columns to the right. Uses SHOW TAB to verify the tab size and level count.

```
This is the first line.
This is the second line.
```

GOLD + PAGE COMMAND

```
Command: SET TAB 5
```

ENTER SUBS

(Move the cursor 15 spaces to the right.)

```
This is the first line.
```

CTRL/A

(Now move the cursor back to the beginning of the line you want to indent.)

TAB

```
                This is the first line.
This is the second line.
```

GOLD + PAGE COMMAND

```
Command:  SHOW TAB
```

ENTER SUBS

```
Command: SHOW TAB
tab size 5; tab level 3
```

---

related commands

— Nokeypad—TC (tab compute)

CTRL/C (Control C) Function—Keypad

KEY SEQUENCE

VT100: CTRL/C or CTRL + C

VT52: CTRL/C or CTRL + C

DESCRIPTION

Pressing CTRL/C interrupts certain operations before EDT finishes processing them. You can use CTRL/C to stop a runaway search through a long file or to stop a long repeat count. CTRL/C halts certain EDT operations. For example, you can use CTRL/C to stop EDT from displaying a whole buffer when you have used the line mode **TYPE** command to move to another buffer.

When EDT aborts the operation, it prints the message **Aborted by CTRL/C**. If EDT cannot stop a particular process, it prints the message **CTRL/C ignored**.

Note: You cannot redefine the CTRL/C function.

EXAMPLE

Stops EDT from printing the entire contents of the buffer DARCY_LET.

GOLD + PAGE COMMAND

```
Command:  SHOW BUFFER
```

ENTER SUBS

```
Command: SHOW BUFFER
 DARCY_LET      40      lines
=MAIN   12      lines
 PASTE  No      lines
Press return to continue
```

RETURN

```
*TYPE =DARCY__LET

1       August 20, 1983
2
3       Mr. Charles R. Darcy
4       Production Manager
5       Midland Manufacturing Corporation
6       East St. Louis, IL  62202
```

CTRL/C

```
Aborted by CTRL/C
```

related commands

- Line—CTRL/C
- Nokeypad—CTRL/C

CTRL/D (Control D) Function—Keypad GOLD/D

KEY SEQUENCE

VT100:	CTRL/D	CTRL + D
VT52:	CTRL/D	CTRL + D
VT100:	GOLD/D	GOLD + D
VT52:	GOLD/D	GOLD + D

DESCRIPTION

Pressing CTRL/D decreases the TAB level count one tab setting. The tab level count is the multiple of the **SET TAB** value that determines the tab indentation level. Suppose the SET TAB value is 5, the tab level count is 3, and the current indentation level is 15. If you press CTRL/D and then the TAB key, the subsequent text will be moved over 10 columns. The SET TAB value is still 5, but the tab level count is now 2 and the current indentation level is 10.

CTRL/D does not move text. You must use the **TAB** function to indent a line. CTRL/D only works if SET TAB is in effect. EDT's default is SET NOTAB. **TD.** is the nokeypad definition for both CTRL/D and GOLD/D.

EXAMPLE

First uses SHOW TAB to find out the current tab size and level count. Then decreases the level count by 1 and uses TAB to move the second line four columns to the right.

```
I.  Main Topic
A.  Major Subtopic
1.  Minor Subtopic
2.  Minor Subtopic
```

GOLD + PAGE COMMAND

```
Command:  SHOW TAB
```

ENTER SUBS

```
Command: SHOW TAB
tab size 4; tab level 2
```

CTRL/D + TAB

```
I.  Main Topic
    A.  Major Subtopic
1.  Minor Subtopic
2.  Minor Subtopic
```

---

related commands

— Nokeypad—TD (tab decrement)

CTRL/E (Control E) Function—Keypad GOLD/E

KEY SEQUENCE

VT100:	CTRL E	CTRL + E
VT52:	CTRL E	CTRL + E
VT100:	GOLD E	GOLD + E
VT52:	GOLD E	GOLD + E

DESCRIPTION

Pressing CTRL/E increments the tab level count by 1. The tab level count is the multiple of the **SET TAB** value that determines the tab indentation level. Suppose the SET TAB value is 5, the tab level count is 2, and the current indentation level is 10. If you press CTRL/E and then the TAB key, the subsequent text will be moved over 15 columns. The SET TAB value is still 5, but the tab level count is now 3 and the current indentation level is 15.

CTRL/E does not move text. You must use the **TAB** function to indent a line. CTRL/E only works if SET TAB is in effect. EDT's default is SET NOTAB. **TI.** is the nokeypad definition for both CTRL/E and GOLD/E.

EXAMPLE

First uses SHOW TAB to find out the current tab size and level count. Then increments the level count by 1 and uses TAB to move the third line eight columns to the right.

```
I.  Main Topic
    A.  Major Subtopic
1.  Minor Subtopic
2.  Minor Subtopic
```

GOLD + PAGE COMMAND

```
Command:  SHOW TAB
```

ENTER SUBS

```
Command: SHOW TAB
tab size 4; tab level 1
```

CTRL/E + TAB

```
I.  Main Topic
    A.  Major Subtopic
        1.  Minor Subtopic
2.  Minor Subtopic
```

---

related commands

— Nokeypad—TI (tab increment)

CTRL/K (Control K) Function—Keypad

KEY SEQUENCE

VT100: CTRL/K or CTRL + K

VT52: CTRL/K or CTRL + K

DESCRIPTION

Pressing CTRL/K starts the key definition process in keypad mode. You can create key definitions using both nokeypad commands and predefined function keys. You can define a key using only nokeypad commands, by pressing predefined function keys, or by combining nokeypad commands with pressing predefined function keys. **DEFK.** is the nokeypad definition for CTRL/K.

Five types of keys can be defined or redefined:

- **A keypad key with or without GOLD**

  All keypad keys can be redefined.

  If the current definition of a key is **GOLD**, you must use the line mode **DEFINE KEY** command to redefine that key. **GOLD** is the default definition for the top left-hand keypad key (PF1 on VT100; blue on VT52). If you are defining a key to have the RESET function, you must type the word RESET in the definition line.

- **CONTROL with a keyboard character, with or without GOLD**

  EDT does not allow you to redefine CTRL/C. Some CONTROL character combinations are system commands and cannot be redefined for that reason. These include O, P, Q, S, X, Y, and [.

  Note that you cannot press CTRL/U to enter its definition in the definition line.

- **GOLD with a keyboard character**

  GOLD can be used with any keyboard character except the digits 0 through 9 and the minus sign.

- **The DELETE key with or without GOLD**

  The DELETE key can be redefined by itself or with GOLD.

  You cannot press DELETE of <X] to enter its definition in the definition line.

- **FUNCTION keys on the LK201 keyboard**

  These include the six keys located above the arrow keys on the terminal's "editing" keypad as well as keys F6 through F20 on the function key row across the top of the keyboard.

To define a key in keypad mode, first press CTRL/K. EDT displays the message:

```
Press the key you wish to define
```

Press the key or key sequence you want to define. EDT then displays the message:

```
Now enter the definition and press ENTER
```

Type your definition and/or press existing keypad function keys and then press the ENTER key as instructed. (The DO key on LK201 keyboards does not process a CTRL/K key definition.) The process is now complete. If you want to review the definition of any key, use the line mode **SHOW KEY** command.

Any nokeypad command can be used in a key definition. The same entity specifiers that are available in nokeypad mode are available for key definitions. Most preset EDT function keys have nokeypad definitions. (GOLD and RESET are exceptions.) Use the SHOW KEY command to see these definitions.

When you look at the definitions for most preset functions, you notice that they end with a period (.). This period is the definition for the **ENTER** function. When you complete your definition with a period, EDT processes the command as soon as you press the key or key sequence. If the period is omitted, EDT stores the command and does not show the results until you press RETURN, or ENTER, or another function key with the period at the end of its definition. For example, if you define GOLD/P to be **2DL.**, when you press the GOLD key and then the P key, two lines will disappear from your screen. If the definition is **2DL**, no change appears on the screen after pressing GOLD and then P. But as soon as you press another function key, the two lines vanish. Be sure to use the period key on the main keyboard, not the one on the keypad, to complete your definitions.

EXAMPLES

example 1:

Defines the key sequence GOLD/D to insert **Dr.[SP]** to the left of the cursor.

```
James T. Roberts
June C. Rumpole
Andrew R. Schaefer
Cynthia Sears
```

CTRL/K

```
Press the key you wish to define
```

GOLD + D

```
Now enter the definition terminated by ENTER.
```

IDr.[SP] CTRL/Z + >

ENTER
SUBS

(Move the cursor to the lines you want to modify and press GOLD/D.)

```
Dr. James T. Roberts
June C. Rumpole
Andrew R. Schaefer
Dr. Cynthia Sears
```

---

example 2:

Defines the key sequence CTRL/P to remove characters from the end of a line. Using the GOLD/repeat feature, you can process all five lines at once.

```
************************
TW184739-47     $5.76
TW184741-38     $3.49
TW184742-84     $7.98
TW184746-64     $9.85
TW184747-38     $2.66
```

CTRL/K

```
Press the key you wish to define.
```

CTRL + P

```
Now enter the definition terminated by ENTER.

(+V D+EL).
```

ENTER
SUBS

```
(Position the cursor directly above the space after the 47.)
```

GOLD + 5 + CTRL + P

```
************************
TW184739-47
TW184741-38
TW184742-84
TW184746-64
TW184747-38
```

---

related commands

- Line—DEFINE KEY
- [Nokeypad—DEFK]

CTRL/L (Control L) Function—Keypad

KEY SEQUENCE

VT100: CTRL/L or CTRL + L

VT52: CTRL/L or CTRL + L

DESCRIPTION

Pressing CTRL/L inserts a form feed character (<FF>) into your text. You can also use CTRL/L to enter a form feed in search strings and SET commands.

^L. is the nokeypad definition of CTRL/L.

EXAMPLES

example 1:

Shows how to enter the default boundary list for the SET ENTITY WORD command. The first **Command:** line shows the text you type up to the point where the boundary limits are entered. The second **Command:** line shows what EDT displays on the screen after you have pressed the spacebar and the five control key sequences. The SHOW ENTITY WORD command displays the control characters differently.

GOLD + PAGE COMMAND

```
Command: SET ENTITY WORD "
```

SPACEBAR + CTRL/I + CTRL/J + CTRL/K + CTRL/L + CTRL/M + "

```
Command: SET ENTITY WORD " ^I^J^K^L^M"
```

ENTER SUBS

GOLD + PAGE COMMAND

```
Command: SHOW ENTITY WORD
```

ENTER SUBS

```
Command: SHOW ENTITY WORD
        <LF><VT><FF><CR>
```

example 2:

Shows how to enter the form feed character in your text.

```
Page 37
```

CTRL/L

```
<FF>Page 37
```

related commands

None.

CTRL/M (Control M) Function—Keypad

KEY SEQUENCE

VT100: CTRL/M or CTRL + M

VT52: CTRL/M or CTRL + M

DESCRIPTION

Pressing CTRL/M inserts a carriage return character (<CR>) into your text. You can also use CTRL/M to enter a carriage return character (<CR> or ^M) in strings and **SET** commands. CTRL/M is not identical to an EDT line terminator. However, in keypad mode, you can use CTRL/M to mean a line terminator in search and substitute strings.

^M. is the nokeypad definition for CTRL/M. When you redefine the CTRL/M key sequence, you also automatically redefine the **RETURN** key. It is recommended that you do not alter the preset definition of CTRL/M for that reason.

EXAMPLES

example 1:

Shows how to enter the default boundary list for the SET ENTITY WORD command. The first **Command:** line shows the text you type up to the point where the boundary limits are entered. The second **Command:** line shows what EDT displays on the screen after you have pressed the spacebar and the five control key sequences. The SHOW ENTITY WORD command displays the control characters differently.

GOLD + PAGE COMMAND

```
Command: SET ENTITY WORD "
```

SPACEBAR + CTRL/I + CTRL/J + CTRL/K + CTRL/L + CTRL/M + -

```
Command: SET ENTITY WORD " _^I_^J_^K_^L_^M"
```

ENTER SUBS

GOLD + PAGE COMMAND

```
Command: SHOW ENTITY WORD
```

ENTER SUBS

```
Command:  SHOW ENTITY WORD
                <LF><VT><FF><CR>
```

example 2:

Uses CTRL/M in the search string to locate the string at the end of the line.

```
Model Number     Price      Sale Price
Model 38567X     $38.99     $33.99
Model 37580Z     $47.99     $41.99
```

GOLD + PAGE COMMAND

```
Search for:
```

9 + 9 + CTRL/M

```
Search for:  99^M
```

ENTER SUBS

```
Model Number     Price      Sale Price
Model 38567X     $38.99     $33.99
Model 37580Z     $47.99     $41.99
```

related commands

None.

CTRL/R (Control R) Function—Keypad GOLD/R

KEY SEQUENCE

VT100:	CTRL/R	CTRL +	R
VT52:	CTRL/R	CTRL +	R
VT100:	GOLD/R	GOLD +	R
VT52:	GOLD/R	GOLD +	R

DESCRIPTION

Pressing CTRL/R (in keypad mode) refreshes the screen display. This function has no effect on the text you are editing. It simply clears and redraws the screen, eliminating any extraneous characters or messages that might have appeared on the screen but are not part of the current text you are editing. Note that CTRL/R performs the same function as CTRL/W in keypad mode. **REF.** is the nokeypad definition for both CTRL/R and GOLD /R.

EXAMPLE

Refreshes the screen to eliminate the notification of new mail on the fourth line.

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1985 in the Glen
Room.  All members are expected to attend.
New mail from XXXXXX::BROWNe served.
```

CTRL/R

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1985 in the Glen
Room.  All members are expected to attend.
Coffee and doughnuts will be served.
```

related commands

- Line—CTRL/R
- Nokeypad—REF (refresh)

CTRL/T (Control T) Function—Keypad GOLD/T

KEY SEQUENCE

VT100:	CTRL/T	CTRL + T
VT52:	CTRL/T	CTRL + T
VT100:	GOLD/T	GOLD + T
VT52:	GOLD/T	GOLD + T

DESCRIPTION

Pressing CTRL/T indents the lines in a select range which must contain only whole lines. After creating the select range, press CTRL/T to move the select range lines over one tab stop to the right. Use a repeat count to indent the lines more than one tab stop. To move the text one tab stop to the left, press GOLD and then the minus sign (–) before CTRL/T. You can move the lines several tab stops to the left by using both the minus sign and a repeat count.

CTRL/T works only if **SET TAB** is in effect. EDT's default is SET NOTAB. To determine the current SET TAB value, use the **SHOW TAB** command. Note that CTRL/T is not affected by the tab level count, nor does that count have any effect on how far text is indented.

TADJSR. is the nokeypad definition for both CTRL/T and GOLD/T.

When the DCL command SET CONTROL=T is in effect, you cannot use CTRL/T in EDT to perform tabbing. If DCL is set to NOCONTROL=T (the default), CTRL/T will perform tabbing in EDT. GOLD/T always performs its tabbing function in EDT, unless you have redefined the key sequence.

EXAMPLE

Sets the tab value to 5. Using a select range for lines 2 and 3, indents them one tab level (5 columns). Using a select range for lines 4 and 5, and a repeat count of 2, indents them two tab levels (10 columns).

```
I.  Main Topic
A.  Major Subtopic
B.  Major Subtopic
1.  Minor Subtopic
2.  Minor Subtopic
```

GOLD + PAGE COMMAND

```
Command:  SET TAB 5
```

ENTER SUBS

EDT Commands

CTRL/T (Control T) Function

SELECT RESET + LINE OPENLINE + LINE OPENLINE + CTRL/T

SELECT RESET + LINE OPENLINE + LINE OPENLINE + GOLD + @ 2 + CTRL/T

```
I.  Main Topic
    A.  Major Subtopic
    B.  Major Subtopic
        1.  Minor Subtopic
        2.  Minor Subtopic
I.  Second Main Topic
```

---

related commands

- Line—TAB ADJUST
- Nokeypad—TADJ (tab adjust)

CTRL/U (Control U) Function—Keypad GOLD/U

KEY SEQUENCE

VT100:	CTRL/U	CTRL + U
VT52:	CTRL/U	CTRL + U
VT100:	GOLD/U	GOLD + U
VT52:	GOLD/U	GOLD + U

DESCRIPTION

Pressing CTRL/U deletes everything from the character to the left of the cursor to the beginning of the line. If the cursor is in the middle or at the end of the line and CTRL/U is pressed, EDT deletes the characters between the cursor and the beginning of that line. If the cursor is at the beginning of a line when CTRL/U is pressed, the line above the cursor is deleted. Text deleted by CTRL/U is stored in the delete line buffer. Use **UND L** to insert or restore the deleted text.

CTRL/U can be used to cancel a **COMMAND**, **FIND**, or **CTRL/K** operation. For example, if you have pressed GOLD/COMMAND and have started to type a line mode command, you can press CTRL/U to return the cursor to the text. If you have pressed GOLD/FIND and have started to type a search string, you can also press CTRL/U to return the cursor to the text. The string in the search buffer remains the same as it was before you pressed GOLD /FIND. Similarly, if you are in the process of creating a key definition with CTRL/K, you can press CTRL/U to cancel the definition process.

DBL. is the nokeypad definition for both CTRL/U and GOLD/U. CTRL/X always performs the same function as CTRL/U, regardless of the definition assigned to CTRL/X.

EXAMPLE

Deletes the first part of the second line.

```
Before signing up for the course, check with Fran Pelletier.
Or you can call the registrar yourself.  Be sure that you have
```

CTRL/U

```
Before signing up for the course, check with Fran Pelletier.
Be sure that you have
```

related commands

— Nokeypad—DBL (delete to beginning of line)

CTRL/W (Control W) Function—Keypad GOLD/W

KEY SEQUENCE

VT100:	CTRL/W	CTRL + W
VT52:	CTRL/W	CTRL + W
VT100:	GOLD/W	GOLD + W
VT52:	GOLD/W	GOLD + W

DESCRIPTION

Pressing CTRL/W or GOLD/W refreshes the screen display. This function has no effect on the text you are editing; it simply clears and redraws the screen, eliminating any extraneous characters or messages that have appeared on the screen but are not part of the current text you are editing. Note that CTRL/W performs the same function as **CTRL/R** in keypad mode. **REF.** is the nokeypad definition for both CTRL/W and GOLD/W.

EXAMPLE

Refreshes the screen to eliminate the notification of new mail on the fourth line.

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1985 in the Glen
Room.  All members are expected to attend.
New mail from ZZZZZZ::BLACKe served.
```

CTRL/W

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1985 in the Glen
Room.  All members are expected to attend.
Coffee and doughnuts will be served.
```

related commands

— Nokeypad—REF (refresh)

CTRL/Z (Control Z) Function—Keypad GOLD/Z

KEY SEQUENCE

VT100:	CTRL/Z	CTRL +	Z
VT52:	CTRL/Z	CTRL +	Z
VT100:	GOLD/Z	GOLD +	Z
VT52:	GOLD/Z	GOLD +	Z

DESCRIPTION

Pressing CTRL/Z shifts EDT from keypad mode to line mode. After you have pressed CTRL/Z, the line mode asterisk prompt (*) appears indicating that EDT is ready to accept line mode commands. To resume keypad editing, use the line mode **CHANGE** command.

EX. is the nokeypad definition for CTRL/Z and GOLD/Z.

EXAMPLE

Shifts from keypad mode to line mode.

```
Customer called Repair Service 5/25/85.
One of his telephones was not working.
He was told to disconnect his phone and
take it to his nearest phone store.
    .
    .
    .

CTRL/Z

*TYPE .
    2       One of his telephones was not working.
```

related commands

- Line—CHANGE
- Nokeypad—EX (exit to line mode)

CUT Function—Keypad

KEY SEQUENCE

VT100:	CUT PASTE	6
VT52:	CUT PASTE	3
LK201:	CUT PASTE	Re-move [1]

DESCRIPTION

Pressing CUT removes the active select range from the current buffer and stores it in the PASTE buffer. You can use CUT to delete large or small sections of text. When you use CUT in conjunction with the **PASTE** function, you can move or copy text from one place in the current buffer to another place in that buffer.

When you use CUT to delete only part of a line, EDT adds a line terminator at the end of the text being stored in the PASTE buffer. The line terminator is necessary because EDT cannot store partial lines in the PASTE buffer. When you use the PASTE function, EDT removes the added line terminator. Thus, when you insert the text, you do not have an extra line terminator.

The steps for moving and copying text are described under the keypad PASTE function. **CUTSR.** is the nokeypad definition of CUT.

EXAMPLE

Using a select range, moves the first line to the PASTE buffer and then uses PASTE to insert that line in the proper chronological sequence between Adams and Madison.

```
Thomas Jefferson
George Washington
John Adams
James Madison
```

SELECT RESET + LINE OPENLINE + CUT PASTE

(Move the cursor to the **J** in **James Madison**.)

GOLD + CUT PASTE

```
George Washington
John Adams
Thomas Jefferson
James Madison
```

[1] On LK201 keyboards, both the 6 keypad key and the **Remove** key have the same preset function.

related commands

– Nokeypad—CUT

DEL C (Delete Character) Function—Keypad

KEY SEQUENCE

VT100: DEL C / UND C ,

VT52: DEL C / UND C 6

DESCRIPTION

Pressing DEL C (delete character) deletes the character on which the cursor is positioned. The cursor stays in the same position, but the remaining characters on the line shift one position to the left. **D+C.** is the nokeypad definition for DEL C.

The deleted character is stored in the delete character buffer. Only one character at a time can occupy that buffer. Each time you delete a character with DEL C or the **DELETE** function, the contents of the delete character buffer are overwritten. Remember that the delete character buffer is inaccessible to you and that its name does not appear in the list displayed by the **SHOW BUFFER** command.

- Use **UND C** to restore or insert the contents of the delete character buffer into your text.

DEL C deletes the character the cursor is on; the DELETE function always deletes the character to the left of the cursor. (**D-C.** is the nokeypad definition for DELETE.)

EXAMPLES

example 1:

Deletes the second period after **Dr.**

```
Dr._ William W. Williams
```

DEL C / UND C

```
D William W. Williams
```

example 2:

Reverses the order of the mistyped letters, using UND C to insert the deleted letter in its proper place.

```
teh wrong turn
```

DEL C / UND C

(Move the cursor one position to the left.)

GOLD + DEL C / UND C

```
the wrong turn
```

related commands

- Line—DELETE
- Nokeypad—DC

DEL EOL (Delete to End of Line) Function—Keypad

KEY SEQUENCE

VT100: GOLD + EOL (DEL EOL) PF1 + 2

VT52: GOLD + EOL (DEL EOL) BLUE (COLOR) + 2

DESCRIPTION

Pressing DEL EOL (delete to end of line) deletes everything on a line from the character the cursor is on up to, but not including, the line terminator. The cursor remains in the same position as it was before DEL EOL was pressed. If the cursor is on a line terminator, DEL EOL deletes that line terminator and all the text up to the next line terminator.

The characters deleted from the line are placed in the delete line buffer. Each time DEL EOL, **DEL L**, or **CTRL/U** is used, the contents of that buffer are overwritten. Use **UND L** to restore or insert the contents of the buffer in your text.

When you use DEL EOL, EDT deletes the characters up to the line terminator to the right of the cursor. DEL L deletes those same characters, but also deletes the line terminator and positions the cursor on the first character of the next line. **D+EL.** is the nokeypad definition for DEL EOL. **D+NL.** is the nokeypad definition for DEL L. CTRL/U (**DBL.** is its nokeypad definition) deletes the text from the character to the left of the cursor to the beginning of the line.

EXAMPLES

example 1:

Deletes the location information after the name.

```
Bob Jamison_ MK01-2/E37
```

GOLD + EOL (DEL EOL)

```
Bob Jamison_
```

example 2:

Deletes the phone extension number on the second line, but does not delete the line terminator.

```
Al Kerr
x3946
_
```

GOLD + EOL (DEL EOL)

DEL EOL (Delete to End of Line) Function

Al Kerr

-

related commands

- Line—DELETE
- Nokeypad—DEL (delete to end of line)

DELETE Function—Keypad

KEY SEQUENCE

VT100:	DELETE	DELETE
VT52:	DELETE	DELETE
LK201:	DELETE	<X

DESCRIPTION

Pressing the DELETE key deletes the character to the left of the cursor. If the cursor is at the beginning of a line, pressing DELETE deletes the preceding line terminator.

When a character is deleted using the DELETE key, that character is placed in the delete character buffer. The contents of the buffer are overwritten each time a character is deleted either by the DELETE function or by **DEL C**. Use **UND C** to restore or insert the contents of the delete character buffer into the text you are editing.

Use the DELETE key to edit the text you type in response to EDT prompts such as **Search for:** or **Command:**. These deleted characters are not stored in the delete character buffer.

The difference between DELETE and DEL C is that DEL C deletes the character that the cursor is on; DELETE deletes the character to the left of the cursor. **D-C.** is the nokeypad definition for DELETE. **D+C.** is the nokeypad definition of DEL C.

EXAMPLES

example 1:

Deletes the **c** from the misspelled name.

```
John Smitch
```

DELETE

```
John Smith
```

example 2:

Deletes the line terminator that is separating the first and second lines.

```
This is the first line.
This is the second line.
This is the third line.
```

DELETE

```
This is the first line.This is the second line.
This is the third line.
```

related commands

- Line—DELETE
- Nokeypad—D-C (delete character to the left)

DEL L (Delete Line) Function—Keypad

KEY SEQUENCE

VT100: DEL L / UND L PF4

VT52: DEL L / UND L GRAY COLOR

DESCRIPTION

Pressing DEL L (delete line) deletes everything on a line starting with the character that the cursor is on up to and including the line terminator. The cursor position remains unchanged on the screen.

If the cursor is on the first character of the line, the entire line is deleted. The cursor is now positioned on the first character of the following line.

The characters deleted by DEL L, **DEL EOL**, or **CTRL/U** are stored in the delete line buffer. Each time a line or piece of line is deleted, the contents of the delete line buffer are overwritten. Use **UND L** to restore or insert the contents of the delete line buffer into the text you are editing.

DEL L always deletes the line terminator to the right of the cursor; DEL EOL deletes only the characters up to that line terminator. **D+NL.** is the nokeypad definition for DEL L. **D+EL.** is the nokeypad definition for DEL EOL. CTRL/U (**DBL.** is its nokeypad definition) deletes the text from the character to the left of the cursor to the beginning of the line.

EXAMPLES

example 1:

Deletes the characters from the current cursor position to the end of the line, including the line terminator.

```
The committee will be composed of George, Mary, Larry,
Bill, and Peter.
```

DEL L / UND L

```
The committee will be composed of George, Mary, Bill, and Peter.
```

example 2:

Alphabetizes the list by deleting the third line and using UND L to insert it in its proper location.

```
Anderson, Richard
Andrews, Harold
Andorsky, Thomas
Anthony, James
```

DEL L / UND L

(Move cursor to the **A** in **Andrews**.)

GOLD + DEL L / UND L

```
Anderson, Richard
Andorsky, Thomas
Andrews, Harold
Anthony, James
```

---

related commands

- Line—DELETE
- Nokeypad—D+NL (delete to next line)

DEL W (Delete Word) Function—Keypad

KEY SEQUENCE

VT100: DEL W / UND W –

VT52: DEL W / UND W 9

DESCRIPTION

Pressing DEL W (delete word) deletes words or parts of words. When the cursor is at the beginning of the word, the entire word and the space following it are deleted. If the cursor is in the middle of the word, only the character that the cursor is on and those characters to the right of the cursor, up to and including the following space(s), are deleted. The characters to the left of the cursor in that word remain in the text. If the word being deleted is at the end of a line, all characters up to, but not including, the line terminator are deleted.

The characters deleted by DEL W and **LINEFEED** (F13 on LK201 keyboards) are stored in the delete word buffer. Each time DEL W or LINEFEED is used, the contents of the delete word buffer are overwritten. Use **UND W** to restore or insert the contents of the delete word buffer into the text you are editing.

DEL W always deletes the cursor character and the remaining characters in the word to the right of the cursor. LINEFEED deletes the word or part of the word to the left of the cursor. **DEW.** is the nokeypad definition for DEL W. **DBW.** is the nokeypad definition for LINEFEED.

EXAMPLES

example 1:

Deletes the word **Major** from the General's title.

```
Major General George H. Mitchell
```

DEL W / UND W

```
General George H. Mitchell
```

example 2:

Changes the order of the names by deleting the last name and then using UND W to insert it after the first name.

```
Smith, Brian
```

DEL W
UND W

(Move the cursor to the end of the line and add a space after **Brian**.)

GOLD + DEL W / UND W

```
Brian Smith,
```

related commands

- Line—DELETE
- Nokeypad—DEW (delete to end of word)

DO Function—Keypad (LK201 only)

KEY SEQUENCE

LK201: DO Do

DESCRIPTION

Pressing DO processes searches and line editing commands in keypad mode. The period (.) is the nokeypad definition for DO. Although DO has the same definition as **ENTER**, you cannot use the DO key to enter a key definition with **CTRL/K**.

When you receive a prompt from EDT in keypad mode, you can use DO to send EDT the information you type in response to the prompt. The two preset EDT functions that have prompts are **COMMAND** and **FIND**.

To use COMMAND, press the GOLD and COMMAND keys. When EDT displays the **Command:** prompt, type the line mode command. Then press DO to send the command to EDT for processing.

To use FIND, press either the LK201 **Find** key or the GOLD and FIND keys on the numeric keypad. When EDT displays the **Search for:** prompt, type the search string. Then press DO to send the string to EDT so it can perform the search.

EXAMPLES

example 1:

Processes the FIND function using the LK201 Find key.

Find

```
Search for:  Chicago
```

DO

```
meeting to be held in Chicago on April 9, 1985
```

example 2:

Processes the COMMAND function.

```
meeting to be held in Chicago on April 9, 1985
```

GOLD + PAGE COMMAND

```
Command:  SUBSTITUTE/Chicago/Boston/
```

DO

```
meeting to be held in Boston on April 9, 1985
```

related commands

- Line— RETURN
- Nokeypad— RETURN

Down Arrow—Keypad

KEY SEQUENCE

VT100: ↓ ↓

VT52: DOWN SECT ↓

DESCRIPTION

Pressing the Down Arrow key moves the cursor down one line toward the bottom of the buffer regardless of EDT's direction. +V. is the nokeypad definition for Down Arrow.

When you use the Down Arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of that line. If you continue to use the Down Arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for Down Arrow and resets it the next time you use the function.

EXAMPLE

Moves the cursor from the end of the first line to the end of the last line.

```
Asuncion, Paraguay_
Bogota, Colombia
Brasilia, Brazil
Buenos Aires, Argentina
Caracas, Venezuela
Georgetown, Guyana
```

↓

```
Asuncion, Paraguay
Bogota, Colombia_
```

↓ + ↓

```
Bogota, Colombia
Brasilia, Brazil
Buenos Aires, Argentina
```

↓ + ↓

```
Buenos Aires, Argentina
Caracas, Venezuela
Georgetown, Guyana_
```

related commands

— Nokeypad—Down Arrow

ENTER Function—Keypad

KEY SEQUENCE

VT100: ENTER SUBS ENTER

VT52: ENTER SUBS ENTER

DESCRIPTION

Pressing ENTER processes searches, line editing commands, and key definitions in keypad mode. EDT generally uses the ENTER function to process keypad editing functions. The period (.) is the nokeypad definition for ENTER.

When you receive a prompt from EDT in keypad mode, use ENTER to send EDT the information you type in response to the prompt. The two preset EDT functions that have prompts are **COMMAND** and **FIND**.

To use COMMAND, press the GOLD and COMMAND keys. When EDT displays the **Command:** prompt, type the line mode command. Then press ENTER to send the command to EDT for processing.

To use FIND, press the GOLD and FIND keys. When EDT displays the **Search for:** prompt, type the search string. Then press ENTER to send the string to EDT so it can perform the search.

You are asked to press the ENTER key when you complete a keypad definition using **CTRL/K** in keypad mode. When the message **Now enter the definition terminated by ENTER** appears, type the definition and then press the ENTER key.

EXAMPLES

example 1:

Processes the FIND function.

GOLD + FNDNXT FIND

```
Search for:  Chicago
```

ENTER SUBS

```
meeting to be held in Chicago on April 9, 1985
```

example 2:

Processes the COMMAND function.

```
meeting to be held in Chicago on April 9, 1985
```

GOLD + PAGE COMMAND

```
Command:  SUBSTITUTE/Chicago/Boston/
```

ENTER
SUBS

```
meeting to be held in Boston_on April 9, 1985
```

---

example 3:

Completes the processing of the CTRL/K key definition operation. The key definition shown in this example creates a keypad substitute function that prompts for both the search and the substitute strings.

CTRL/K

```
Press the key you wish to define
```

GOLD + S

```
Now enter the definition terminated by ENTER
S/?'Find: '/?' Substitute: '/.
```

ENTER
SUBS

---

example 4:

Using the new substitute function created in Example 3, sends the strings to EDT after you type them.

```
the line entity.
```

GOLD + S

```
Find: line
```

ENTER
SUBS

```
Find: line Substitute: paragraph
```

ENTER
SUBS

```
the paragraph_entity.
```

---

related commands

- Line—RETURN
- Nokeypad—RETURN

EOL (End of Line) Function—Keypad

KEY SEQUENCE

VT100: EOL DEL EOL | 2

VT52: EOL DEL EOL | 2

DESCRIPTION

Pressing EOL (end of line) moves the cursor to the end of the current line if the direction is forward. If the current direction is backward, the cursor moves to the end of the previous line. If the cursor is already at the end of a line, EOL moves it to the end of the next or previous line, depending on the current direction. Use **BACKSPACE** (F12 on LK201 keyboards) to move the cursor to the beginning of a line. **EL.** is the nokeypad definition of EOL.

EXAMPLES

example 1:

Moves the cursor to the right to the nearest line terminator.

```
The meeting is scheduled for Friday afternoon.
```

EOL DEL EOL

```
The meeting is scheduled for Friday afternoon._
```

example 2:

Moves the cursor to the left to the nearest line terminator, after pressing BACKUP.

```
3 typewriters
2 typing stands
```

BACKUP TOP + EOL DEL EOL

```
3 typewriters
2 typing stands
```

related commands

— Nokeypad—EL (end of line)

FILL Function (VT100)—Keypad CTRL/F Function (VT52)

KEY SEQUENCE

VT100: GOLD + SECT FILL PF1 + 8

VT52: CTRL/F CTRL + F

DESCRIPTION

Pressing FILL takes a select range of lines and reorganizes the text so that the maximum number of whole words can fit within the current line width. The default line width for EDT is the terminal width that the operating system passes to EDT. Use the line mode **SHOW SCREEN** command to find the current screen/line width. The valid screen width values for screen mode editing are 80 and 132. (132 is only valid for VT100-type terminals with AVO—advanced video option.) If your screen width is set to 80, EDT will fill lines to column 79; if your screen width is 132, EDT will fill lines to column 131.

If you want to use a line length other than 80 or 132 for filling text, you must use the line mode **SET WRAP** command. The SET WRAP command also affects the line length that EDT uses for inserting text in keypad mode. EDT uses the SET SCREEN value to determine the line length for filling text only if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses the wrap value, regardless of the SET SCREEN width. You can use the **SHOW WRAP** command to find out the current wrap value or setting.

The filling process considers a blank line to be a break between paragraphs. Even if there are spaces on the blank line, EDT fills the text up to the blank line and then resumes filling with the next line that contains text.

FILLSR. is the nokeypad definition for FILL on VT100-type terminals and for CTRL/F on VT52 terminals.

The FILL function is available on all VT100-type terminals. You must use CTRL/F on VT52 terminals to perform the FILL function.

EXAMPLES

example 1:

Creates a select range of four lines and then reformats it using the current SET SCREEN width—80 characters.

```
When the last line of the file is reached,
the computer reads the end of block
indicator
and displays the system prompt character.
```

VT100: SELECT RESET + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE

VT100: GOLD + SECT FILL

VT52: SELECT RESET + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE

VT52: CTRL/F

```
When the last line of the file is reached, the computer reads the end of block
indicator and displays the system prompt character.

-
```

---

example 2:

First uses the line mode SET WRAP command to limit the line length to 30 characters. Then creates a select range of four lines and reformats the text.

```
When the last line of the file is reached,
the computer reads the end of block
indicator
and displays the system prompt character.
```

VT100: GOLD + PAGE COMMAND

```
Command: SET WRAP 30
```

VT100: ENTER SUBS

VT100: SELECT RESET + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE

VT100: GOLD + SECT FILL

VT52: GOLD + PAGE COMMAND

```
Command: SET WRAP 30
```

VT52: ENTER SUBS

VT52: SELECT RESET + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE + LINE OPENLINE

VT52: CTRL + F

```
When the last line of the file
is reached, the computer reads
the end of block indicator and
displays the system prompt
character.

-
```

---

related commands

- Line—FILL
- Nokeypad—FILL

FIND Function—Keypad

KEY SEQUENCE

VT100: GOLD + FNDNXT (FIND) — PF1 + PF3

VT52: GOLD + FNDNXT (FIND) — BLUE COLOR + 8

LK201: GOLD + FNDNXT (FIND) — Find [1]

DESCRIPTION

Pressing FIND sets up a search procedure. When you press GOLD and then FIND, EDT displays the prompt **Search for:** at the bottom of the screen. Type the string you want to locate. Then push **ENTER** to process the search in the current direction.

After you have typed in your search string, you can press **ADVANCE** instead of ENTER to search toward the end of the buffer or you can press **BACKUP** to search backward toward the top. The direction you use to process FIND becomes EDT's current direction.

EDT can perform searches in several ways. The default ways are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search in the portion of the buffer between the cursor position and the beginning or end of the buffer, depending on the direction of the search. Use the **SET SEARCH** command to change the way EDT performs searches. The **SHOW SEARCH** command tells you which search parameters are currently in effect.

^@?'Search for: '^@. is the nokeypad definition for FIND.

EXAMPLES

example 1:

Searches for the next occurrence of **Walker** in the current direction.

```
Wes Chandler
Hershel Walker
Wesley Walker
```

GOLD + FNDNXT (FIND)

```
Search for:  Walker
```

ENTER (SUBS)

[1] On LK201 keyboards, both the PF1/PF3 key sequence and the Find key have the same preset function.

```
Wes Chandler
Hershel Walker
Wesley Walker
```

example 2:

Using the same list of names with the cursor in its new position, searches for **Wes** in the backward direction.

GOLD + FNDNXT FIND

```
Search for:  Wes
```

BACKUP TOP

```
Wes Chandler
Hershel Walker
Wesley Walker
```

example 3:

Using the same list of names with the cursor in its new position, searches for **Wes** in the forward direction.

GOLD + FNDNXT FIND

```
Search for:  Wes
```

ADVANCE BOTTOM

```
Wes Chandler
Hershel Walker
Wesley Walker
```

example 4:

Using the same list of names with the cursor in its new position, searches for the next occurrence of **Walker** in the current direction.

GOLD + FNDNXT FIND

```
Search for:  Walker
```

ENTER SUBS

```
Wes Chandler
Hershel Walker
Wesley Walker
```

related commands

- Line—FIND
- Nokeypad—"string"

FNDNXT (Find Next) Function—Keypad

KEY SEQUENCE

VT100: [FNDNXT] [PF3]

VT52: [FNDNXT FIND] [8]

DESCRIPTION

After a search string has been established by **FIND**, you can use FNDNXT (find next) to locate the next occurrence of that string. The direction for FNDNXT is always the current EDT direction. You can change directions without affecting the search string.

The search string established by FIND remains in effect until you use FIND again or use some other EDT function that overwrites the contents of the search buffer.

EDT can perform searches in several ways. The default ways are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search on the portion of the buffer between the cursor position and the beginning or end of the buffer, depending on the direction of the search. Use the **SET SEARCH** command to change the way EDT performs searches. The **SHOW SEARCH** command tells you which search parameters are currently in effect.

"". is the nokeypad definition for FNDNXT.

EXAMPLES

example 1:

First uses FIND to load the search buffer. Then es FNDNXT to locate the fourth occurrence of the string **wi** in the list.

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

[GOLD] + [FNDNXT FIND]

```
Search for:  wi
```

[ENTER SUBS]

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

FNDNXT FIND + FNDNXT FIND + FNDNXT FIND

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

---

example 2:

With the current search string still **wi**, but the direction changed to backward, locates the string at the beginning of the third line.

BACKUP TOP
FNDNXT FIND

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

---

related commands

- Line—FIND ""
- Nokeypad ""

GOLD Function—Keypad

KEY SEQUENCE

VT100: GOLD PF1

VT52: GOLD BLUE COLOR

DESCRIPTION

Pressing GOLD together with other keypad and keyboard keys performs various editing functions. GOLD is like the SHIFT key in that it does nothing by itself.

When used with a keypad key, GOLD causes EDT to perform that key's alternate function. For example, to use the COMMAND function, you must first press GOLD and then the 7 key on the keypad. If you do not press GOLD, EDT performs the PAGE function. Using EDT's key definition facility, you can redefine any GOLD/keypad sequence to perform a different function during your EDT session.

The define key feature allows you to designate a GOLD/keyboard key sequence to perform a keypad editing function for the duration of your editing session. You can also use GOLD in combination with a CTRL/character sequence and with the DELETE key to define new keypad mode functions. GOLD/FUNCTION key sequences can be defined on terminals that have LK201 keyboards.

GOLD can be used with keyboard number keys to designate the number of times for EDT to repeat a keypad editing function. First press GOLD, next the keyboard number key(s), and then the keypad function key(s) that you want EDT to repeat. When EDT's direction is set to forward, you can use GOLD followed by a minus sign (–) to change EDT's direction to backward temporarily. This feature allows you to process an EDT function in the opposite direction, without having to reset EDT's direction. For example, you can use GOLD/–2 with **WORD** to have the cursor back up two words without changing EDT's direction. The maximum number of times you can repeat a function with the GOLD/repeat feature is 32767.

When you use the **SPECINS** function, you first press GOLD, then the keyboard digits for the decimal equivalent value of the character you want to insert. Then press GOLD again—this time to access the alternate function on the keypad function key—and finally the SPECINS key.

GOLD is the nokeypad definition for GOLD. Note that there is no period at the end of the definition because GOLD is not a nokeypad command. You must use the line mode DEFINE KEY command to redefine a key that has GOLD as its definition.

EXAMPLES

example 1:

Causes EDT to process FIND, not FNDNXT.

GOLD + FNDNXT / FIND

example 2:

Causes EDT to perform the PASTE function, not CUT.

GOLD + CUT / PASTE

example 3:

Using CTRL/K, defines the key sequence GOLD/E to locate a semicolon (;) and delete to the end of line.

CTRL/K

```
Press the key you wish to define.
```

GOLD + E

```
Now enter the definition terminated by ENTER.
";"D+EL.
```

ENTER / SUBS

example 4:

Causes EDT to repeat the period (.) 60 times, thus creating a line of 60 dots.

GOLD + 6 + 0 + .

```
.............................................................
```

example 5:

Using the minus sign (–) as well as a repeat count of two, causes EDT to change the case of the two letters preceding the cursor.

```
New vt100 terminals are now available.
```

GOLD + – + 2 + GOLD + WORD / CHNGCASE

```
New VT100 terminals are now available.
```

example 6:

With SPECINS, inserts the form feed character into the text.

```
as of the last day in August.

The next item on the agenda will be to clarify the
```

VT100: GOLD + 1 + 2 + GOLD + CHAR / SPECINS

VT52: GOLD + 1 + 2 + GOLD + RIGHT / SPECINS

```
as of the last day in August.
<FF>
The next item on the agenda will be to clarify the
```

related commands

None.

HELP Function—Keypad

KEY SEQUENCE

VT100:	HELP	PF2
VT52:	HELP	RED COLOR
LK201:	HELP	Help [1]

DESCRIPTION

Pressing HELP provides information on EDT's preset keypad and control functions. Using HELP puts you in touch with EDT's HELP facility; it has no effect on your editing session. When you exit from HELP, the screen is redrawn exactly as it was before you pressed HELP and the cursor is in the same position as before.

When you press HELP, EDT displays a diagram of the keypad functions and a list of preset control key functions. For help on a particular keypad function key, press the appropriate keypad key. For information on a GOLD/keypad sequence, press only the keypad key. Information for both the primary and alternate functions of that key will be displayed. For information on a control key sequence, press both the CTRL and keyboard keys after you are in the keypad HELP facility. For help on a GOLD/keyboard key sequence, press only the keyboard key; do not press GOLD.

To exit from HELP, press the spacebar.

If you have access to more than one HELP file, use the **SET HELP** command to change HELP files. The **SHOW HELP** command prints the name of the HELP file that is currently available for your editing session.

To define another key to perform the HELP function, use the nokeypad **HELP** command. **HELP.** is the nokeypad definition for HELP.

related commands

— Line—HELP

— [Nokeypad—HELP]

[1] On LK201 keyboards, both the PF2 key and the Help key have the same preset function.

Left Arrow—Keypad

KEY SEQUENCE

VT100: [←] [←]

VT52: [LEFT APPEND] [←]

DESCRIPTION

Pressing the Left Arrow moves the cursor one character to the left, regardless of EDT's direction. **-C.** is the nokeypad definition for Left Arrow.

If the cursor is at the first character position of a line, pressing Left Arrow moves the cursor to the line terminator of the previous line.

EXAMPLE

Moves the cursor to the left, first to the beginning of the second line and then to the end of the first line.

```
July 1, 1985
October 1, 1985
```

[←] + [←] + [←]

```
July 1, 1985
October 1, 1985
```

[←]

```
July 1, 1985
October 1, 1985
```

related commands

— Nokeypad—Left Arrow

LINE Function—Keypad

KEY SEQUENCE

VT100: [LINE OPEN LINE] [0]

VT52: [LINE OPEN LINE] [0]

DESCRIPTION

Pressing LINE moves the cursor to the beginning of the next line if the direction is forward or to the beginning of the current line if the direction is backward. If the cursor is at the beginning of a line and the direction is backward, the cursor moves to the beginning of the previous line. **L.** is the nokeypad definition for LINE.

EXAMPLES

example 1:

Moves the cursor from the middle of the second line to the beginning of the third line.

```
Software Documentation
Software Development
Production Groups
```

[LINE OPEN LINE]

```
Software Documentation
Software Development
Production Groups
```

example 2:

Using the same text and original cursor position, reverses the direction and moves the cursor to the beginning of the first line.

[BACKUP TOP] + [LINE OPEN LINE] + [LINE OPEN LINE]

```
Software Documentation
Software Development
Production Groups
```

related commands

— Nokeypad—L (line)

LINEFEED Function—Keypad CTRL/J

KEY SEQUENCE

VT100:	LINEFEED	LINE FEED
VT52:	LINEFEED	LINE FEED
LK201:	LINEFEED	F13
All:	CTRL/J	CTRL + J

DESCRIPTION

Pressing LINEFEED deletes the word or characters in a word to the left of the cursor up to the beginning of the previous word. It is similar to **DEL W**, which deletes the word or characters in a word to the right of the cursor up to the beginning of the next word. **DBW.** is the nokeypad definition for LINEFEED.

If the cursor is on a space when LINEFEED is pressed, the word preceding the space is deleted, usually leaving two spaces in a row. If the cursor is at the end or in the middle of a word, all characters in that word to the left of the cursor are deleted. The letter that the cursor is on remains in the text.

When the cursor is at the beginning of a word, the preceding word and space are deleted by LINEFEED. If the cursor is at the beginning of a line, LINEFEED deletes the preceding line terminator.

All characters deleted by LINEFEED are stored in the delete word buffer. Each time DEL W or LINEFEED is used, the contents of the delete word buffer are overwritten. Use **UND W** to insert or restore the contents of the delete word buffer in your text.

The LINEFEED key and CTRL/J always have the same preset function in EDT. When you redefine the LINEFEED key, you redefine CTRL/J (except for terminals with LK201 keyboards when they are operating in VT200 mode.) To redefine the LINEFEED key using the line mode **DEFINE KEY** command, type DEFINE KEY CONTROL J. To find out what the definition of the LINEFEED key is, type **SHOW KEY** CONTROL J. For terminals with LK201 keyboards, use DEFINE KEY FUNCTION 25 and SHOW KEY FUNCTION 25 for the F13 key.

EXAMPLE

Deletes the characters to the left of the cursor up to the next word boundary.

```
This guide describes the VT100 terminal_
```

LINEFEED

```
This guide describes the VT100 _.
```

related commands

— Nokeypad—DBW (delete to beginning of word)

OPEN LINE Function—Keypad

KEY SEQUENCE

VT100: GOLD + LINE (OPEN LINE) PF1 + 0

VT52: GOLD + LINE (OPEN LINE) BLUE (COLOR) + 0

DESCRIPTION

Pressing OPEN LINE inserts a line terminator in the text you are editing at the current cursor position and makes the line terminator the new cursor character. If the cursor is initially at the beginning of a line, the text on that line is moved down so that the cursor is on the blank line.

When the cursor is in the middle of a line, the text to the right of the cursor and the cursor character itself move to a new line. The cursor is now on the line terminator that OPEN LINE inserts. When the cursor is at the end of a line, a line terminator is added, creating a blank line below the current line.

RETURN and **CTRL/M** also insert line terminators in your text. However, neither of these functions moves the cursor to the inserted line terminator. The cursor remains on the same character.

(^M-C). is the nokeypad definition for OPEN LINE. **^M** is the definition of RETURN.

EXAMPLES

example 1:

Inserts a blank line between the first and second line.

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

GOLD + LINE (OPEN LINE)

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

example 2:

Using the same text, inserts a blank line above the last line.

(Move the cursor to the beginning of the last line.)

GOLD + LINE OPEN LINE

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.

The presentation was very good.
```

example 3:

Using the same text, inserts a line terminator that moves the words **The topics covered** to a new line.

(Move the cursor to the **T** in **The** in the middle of the second line.)

GOLD + LINE OPEN LINE

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.
The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

related commands

- Line—INSERT RETURN
- Nokeypad—I RETURN

PAGE Function—Keypad

KEY SEQUENCE

VT100: PAGE COMMAND 7

VT52: PAGE COMMAND 7

DESCRIPTION

Pressing PAGE moves the cursor to a position at the right of the next page marker in your text. The cursor is always located after the page marker, but the direction that EDT moves to find the page marker depends on the current direction. In order to use PAGE, the text you are editing must have PAGE boundary markers. The default page marker is the form feed character (CTRL/L, decimal value 12, displayed by EDT as <FF> .)

PAGETOP. is the nokeypad definition for PAGE. This means that EDT moves the page marker line to the top of the screen if there are more than 22 lines between the page marker and the end of the buffer.

If you have no page markers in your buffer, the PAGE entity is the same as the whole buffer. When EDT's direction is forward, PAGE moves the cursor to the end of buffer ([EOB]) mark. If the current direction is backward, PAGE moves the cursor to the beginning of the buffer.

You can use the **SET ENTITY** PAGE command to define any string of characters as the page marker for the duration of your editing session. The marker can be either a single character that you insert in the text, such as an exclamation point (!), or a series of characters, such as a RUNOFF header level (.HL1).

If you are using the default page marker, you can use **SET TEXT** PAGE to have EDT display some other text in place of the <FF> page marker for the duration of your EDT session.

EXAMPLES

example 1:

Moves the cursor to the next page marker in the text.

```
and will be forthcoming as soon as we receive your payment.
<FF>
If you are not completely satisfied with this product, please
notify us immediately so that we can arrange for you to return
```

PAGE COMMAND

```
and will be forthcoming as soon as we receive your payment.
<FF>
If you are not completely satisfied with this product, please
notify us immediately so that we can arrange for you to return
```

example 2:

Sets the PAGE entity to be the string **.HL1**. Then moves the cursor to the next occurrence of the new page marker.

```
Now that you understand the basic concepts of word
processing, we can move on to an actual session at the terminal.
.HL1 Starting to Work with your Word Processing System
```

GOLD + PAGE COMMAND

```
Command:  SET ENTITY PAGE ".HL1"
```

ENTER SUBS

PAGE COMMAND

```
Now that you understand the basic concepts of word
processing, we can move on to an actual session at the terminal.
.HL1 Starting to Work with your Word Processing System
```

related commands

— Nokeypad—PAGE

PASTE Function—Keypad

KEY SEQUENCE

VT100:	GOLD + CUT PASTE	PF1 + 6
VT52:	GOLD + CUT PASTE	BLUE COLOR + 3
LK201:	GOLD + CUT PASTE	Insert Here[1]

DESCRIPTION

Pressing PASTE with **CUT** or **APPEND** is used to copy or move text within a buffer. PASTE copies the text currently residing in the PASTE buffer into the current buffer. The PASTE buffer contents are inserted to the left of the cursor regardless of EDT's current direction. PASTE has no effect on the contents of the PASTE buffer. **PASTE.** is the nokeypad definition for PASTE.

To move text from one place in your buffer to another, you need to use **SELECT**, CUT, and PASTE in the following order:

1. Use SELECT to create a select range of the text you want to move.
2. Press CUT to delete the text from the current buffer and put it into the PASTE buffer.
3. Move the cursor to the new location where you want to insert the deleted text.
4. Press PASTE to have EDT copy the PASTE buffer text into your current buffer to the left of the cursor.

You can use SELECT, CUT and PASTE to copy text that exists in one place in your buffer to a second location. Follow the same procedure as for moving text, but add an additional step between the second and third steps:

2a. Press PASTE to have EDT restore the deleted text in its original location.

Each time you use CUT, EDT overwrites the contents of the PASTE buffer. If you want to add more text to the buffer before you insert it in the new location, you can use APPEND. APPEND deletes the select range text from its current location and adds it to the end of the PASTE buffer. When you press PASTE, both the text you deleted with CUT and the text you deleted with APPEND are inserted to the left of the cursor.

It is possible to edit the PASTE buffer. Using the line mode **FIND =buffer** command, you can enter the PASTE buffer, make your changes, and then return to your original buffer. Now, when you use PASTE, the revised buffer contents are inserted at the cursor location.

When you use CUT to put part of a line into a buffer, EDT adds a line terminator at the end of the text since EDT buffers cannot hold partial lines. PASTE removes the added line terminator so that when you insert the text, you do not have an extra line terminator.

[1] On the LK201 keyboard, both the PF1/6 key sequence and the Insert Here key have the same preset function.

You can use the line mode FIND command to move from one buffer to another during your EDT session. Then you can use PASTE to put the contents of the PASTE buffer in that buffer.

EXAMPLES

example 1:

Uses a select range and CUT to move the second element of the list to the PASTE buffer. After moving the cursor, inserts the PASTE buffer contents to reorder the list.

```
Peripheral hardware, also known as input/output devices,
includes terminals, line printers, tapes, and disks.
```

SELECT / RESET + WORD / CHNGCASE + WORD / CHNGCASE

```
Peripheral hardware, also known as input/output devices,
includes terminals, line printers, tapes, and disks.
```

CUT / PASTE

```
Peripheral hardware, also known as input/output devices,
includes terminals, tapes, and disks.
```

(Move the cursor to the **a** of **and disks**.)

GOLD + CUT / PASTE

```
Peripheral hardware, also known as input/output devices,
includes terminals, tapes, line printers, and disks.
```

example 2:

Uses a select range and CUT to move the separating line to the PASTE buffer. Immediately restores the deleted line to its former position. Moves the cursor to another location and copies the separating line there.

```
End of Section 1
***** ***** *****
```

SELECT / RESET + LINE / OPENLINE + CUT / PASTE + GOLD + CUT / PASTE

```
End of Section 1
***** ***** *****

```

(Move the cursor to the next location where you want to put the line.)

```
End of Section 2

```

GOLD + CUT / PASTE

```
End of Section 2
***** ***** *****

```

example 3:

Using a select range and CUT, moves three lines to the PASTE buffer. Then uses the line mode FIND command to enter the PASTE buffer and edit the text. Again, uses the line mode FIND command to return to the MAIN buffer. After finding the new location for the text, inserts the revised contents of the PASTE buffer.

```
You combine the control key with a character key by
pressing both simultaneously, i.e., hold the CTRL
key down while you press the character key.
```

SELECT / RESET + LINE / OPEN LINE + LINE / OPEN LINE + LINE / OPEN LINE + CUT / PASTE

GOLD + PAGE / COMMAND

```
Command:  FIND =PASTE
```

ENTER / SUBS

(Move cursor to the **i** of **i.e.** and delete the next four characters. Now type **that is** to replace **i.e.**.)

GOLD + PAGE / COMMAND

```
Command: FIND =MAIN
```

ENTER / SUBS

(Move the cursor to the new location where you want to insert the text.)

GOLD + CUT / PASTE

```
You combine the control key with a character key by
pressing both simultaneously, that is, hold the CTRL
key down while you press the character key.
_
```

related commands

- Line—COPY, MOVE
- Nokeypad—PASTE

REPLACE Function—Keypad

KEY SEQUENCE

VT100: GOLD + APPEND REPLACE PF1 + 9

VT52: GOLD + UP REPLACE BLUE COLOR + 1

DESCRIPTION

Pressing REPLACE deletes the text in the select range and replaces it with the contents of the PASTE buffer. REPLACE enables you to delete different blocks of text and replace them all with the same text. EDT stores the deleted text in a buffer called DELETE. If the buffer does not exist, EDT creates it. If you have created a buffer called DELETE, EDT overwrites the text you had in that buffer with the newly deleted text. Each time you use REPLACE, EDT overwrites the text in the DELETE buffer. The DELETE buffer can be entered and edited and its name appears on the **SHOW BUFFER** list.

You can use **CUT** to put the replacement text into the PASTE buffer, or you can move to the PASTE buffer with the line mode **FIND** command and insert the text directly there.

CUTSR=DELETE PASTE. is the nokeypad definition for REPLACE.

EXAMPLE

Creates a select range of four lines. Moves the selected text to the DELETE buffer and inserts the contents of the PASTE buffer.

```
Martha Jackson
Purchasing Department
Kitchen Cabinets, Inc.
Post Mills, VT  05058
```

VT100: SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE

VT100: GOLD + APPEND REPLACE

VT52: SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE

VT52: GOLD + UP REPLACE

```
Judy Henning
Purchasing Department
Kitchens Incorporated
Cumberland, RI  02864

_
```

related commands

- Line—REPLACE
- Nokeypad—R (replace)

RESET Function—Keypad

KEY SEQUENCE

VT100: GOLD + SELECT RESET PF1 + .

VT52: GOLD + SELECT RESET BLUE COLOR + .

DESCRIPTION

Pressing RESET changes several conditions of your editing session:

- Cancels an active select range
- Sets EDT's current direction to **ADVANCE**
- Sets EDT to the default **DMOV** state

RESET also has a special function within the **CTRL/K** key definition facility. Namely, you can use RESET to delete the text on the definition line if you want to start your definition over again.

RESET is the nokeypad definition for RESET. Note that there is no period at the end of the definition. This is because RESET is not a nokeypad command.

EXAMPLES

example 1:

Cancels the active select range.

```
The next meeting is scheduled for December 13th in my office.
```

SELECT RESET + WORD CHNGCASE + WORD CHNGCASE

```
The next meeting is scheduled for December 13th in my office.
```

GOLD + SELECT RESET

```
The next meeting is scheduled for December 13th in my office.
```

example 2:

Uses RESET to delete to beginning of line so that you can retype the key definition correctly. Shows how to create a key definition to access nokeypad commands directly from keypad mode.

CTRL/K

```
Press the key you wish to define
```

CTR + N

```
Now enter the definition terminated by ENTER
?'Bijetoad cinn_
```

GOLD + SELECT / RESET

```
Now enter the definition terminated by ENTER
?'Nokeypad command: '._
```

---

related commands

— Nokeypad—DESEL

RETURN Function—Keypad

KEY SEQUENCE

VT100: RETURN RETURN

VT52: RETURN RETURN

DESCRIPTION

Pressing RETURN adds a line terminator to the text you are editing. The new line terminator is inserted to the left of the current cursor position. The cursor remains on the same character where it was before you pressed RETURN. If the cursor is at the beginning of the line, a blank line is created above the current cursor line.

When the cursor is in the middle of a line, RETURN moves the cursor character and all the text to the right of the cursor to a new line. When the cursor is at the end of a line, RETURN adds a line terminator, creating a blank line below the current line. The cursor is now positioned at the beginning of the new blank line. **OPEN LINE** also inserts a line terminator in your text, but it positions the cursor on the new line terminator.

You can redefine the RETURN key, although this is not recommended. When you redefine the RETURN key, you also redefine **CTRL/M**. To find out the definition of the RETURN key, type **SHOW KEY** CONTROL M.

EXAMPLES

example 1:

Inserts a blank line between the first and second lines.

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

RETURN

```
Meeting Report

On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

example 2:

Using the same text, inserts a blank line above the last line.

(Move the cursor to the beginning of the last line.)

RETURN

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

---

example 3:

Using the same text, inserts a line terminator that moves the words **The topics covered** to a new line.

(Move the cursor to the **T** in **The topics covered**.)

RETURN

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.
The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

---

related commands

None.

Right Arrow—Keypad

KEY SEQUENCE

VT100: → or →

VT52: RIGHT SPECINS or →

DESCRIPTION

Pressing the Right Arrow moves the cursor one character to the right, regardless of EDT's direction. +C. is the nokeypad definition for Right Arrow.

If the cursor is on a line terminator, Right Arrow moves the cursor to the first character on the next line.

EXAMPLE

Moves the cursor to the right to the end of first line. Then moves the cursor to the beginning of the second line.

```
January 1, 1985
April 1, 1985
```

→ + → + →

```
January 1, 1985
April 1, 1985
```

→

```
January 1, 1985
April 1, 1985
```

related commands

– Nokeypad—Right Arrow

SECT (Section) Function—Keypad

KEY SEQUENCE

VT100:	SECT FILL	8
VT52:	GOLD + DOWN SECT	BLUE COLOR + ↓
LK201:	SECT FILL	Next Screen [1]
LK201:	BACKUP TOP	Prev Screen [1]

DESCRIPTION

Pressing SECT (section) moves the cursor one section—16 lines—toward the end or beginning of the buffer, depending on EDT's current direction. The cursor is always placed at the beginning of the new current line regardless of its previous position. **(16L).** is the nokeypad definition for SECT.

EXAMPLE

Moves the cursor from its position at the end of Test Line 1 to the beginning of Test Line 17.

```
Test Line  1
Test Line  2
Test Line  3
Test Line  4
Test Line  5
Test Line  6
Test Line  7
Test Line  8
Test Line  9
Test Line 10
Test Line 11
Test Line 12
Test Line 13
Test Line 14
Test Line 15
Test Line 16
Test Line 17
```

VT100: SECT FILL

VT52: GOLD + DOWN SECT

[1] On the LK201 keyboard, the **Next Screen** key moves the cursor 16 lines forward, regardless of EDT's current direction. The **Prev Screen** key moves the cursor 16 lines backward, regardless of EDT's current direction. On all screen mode terminals the SECT function moves the cursor 16 lines in the current direction.

```
Test Line  1
Test Line  2
Test Line  3
Test Line  4
Test Line  5
Test Line  6
Test Line  7
Test Line  8
Test Line  9
Test Line 10
Test Line 11
Test Line 12
Test Line 13
Test Line 14
Test Line 15
Test Line 16
Test Line 17
```

related commands

None.

SELECT Function—Keypad

KEY SEQUENCE

VT100:	SELECT RESET	.
VT52:	SELECT RESET	.
LK201:	SELECT RESET	Select [1]

DESCRIPTION

Pressing SELECT sets up a select range for use with keypad functions such as **APPEND**, **CHNGCASE**, **CUT**, **FILL**, **REPLACE**, **SUBS**, and **CTRL/T**. Start with the cursor at one end of the text you want selected. Next press SELECT to mark that position as the beginning of the select range. Then, using the arrow keys and/or function keys that move the cursor, mark the other end of the text being selected. Now you are ready to press a function key that uses a select range.

The **RESET** function cancels the select range. If you have included more text than you wanted in the select range, you can move the cursor back toward the position initially marked by SELECT, using arrow keys and cursor moving functions, thus reducing the size of the range. Adjusting select ranges on VT100 terminals is easy because EDT shows the text in reverse video. On VT52 terminals, you might find it easier to use RESET to cancel the select range and then start over.

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that the select range contain only whole lines.

SEL. is the nokeypad definition for SELECT.

EXAMPLES

example 1:

Creates a select range of the last line and, using CUT, moves it to the PASTE buffer. Uses PASTE to insert the line in its proper alphabetical order.

```
Introduction to BASIC
Roget's Thesaurus
American Heritage Dictionary
_
```

SELECT RESET + LINE OPEN LINE + CUT PASTE

```
Introduction to BASIC
Roget's Thesaurus
_
```

↑ + ↑ + GOLD + CUT PASTE

[1] On the LK201 keyboard, both the period (.) keypad key and the **Select** key have the same preset function.

```
American Heritage Dictionary
Introduction to BASIC
Roget's Thesaurus
```

example 2:

First uses the COMMAND function to process the line mode SET WRAP 60. Then creates a select range of four lines. Reformats the select range text with the FILL function.

```
In order to process data, there must be
communication between you and the computer.  This communication
is achieved by means of the
two fundamental computer components:  hardware and software.
```

VT100: GOLD + PAGE (COMMAND)

```
Command:  SET WRAP 60
```

VT100: ENTER SUBS

VT100: SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE

VT100: GOLD + SECT (FILL)

VT52: GOLD + PAGE (COMMAND)

```
Command:  SET WRAP 60
```

VT52: ENTER SUBS

VT52: SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE

VT52: CTRL/F

```
In order to process data, there must be communication
between you and the computer.  This communication is
achieved by means of the two fundamental computer
components:  hardware and software.
_
```

related commands

— Nokeypad—SEL (select)

SPECINS (Special Insert) Function—Keypad

KEY SEQUENCE

VT100: GOLD + CHAR SPECINS PF1 + 3

VT52: GOLD + RIGHT SPECINS BLUE COLOR + →

DESCRIPTION

Pressing SPECINS (special insert) enables you to insert any character from the DEC Multinational Character Set into your text, using the character's decimal equivalent value. You can use SPECINS to enter ASCII control characters, such as CTRL/L, or letters with diacritical marks such as the umlaut (¨) or acute accent (´). **ASC.** is the nokeypad definition for SPECINS.

To use SPECINS, first press GOLD. Next, type the decimal equivalent number for the character you want to insert. Use the main keyboard digits to type this number; do not use the keypad number keys. EDT displays the number you typed at the bottom of the screen. You can use the DELETE key to edit the number. Now press GOLD again, this time followed by the SPECINS key. The EDT symbol for the character you inserted appears on the screen to the left of the cursor.

Each time you want to enter a special character, you must repeat the entire procedure. You cannot enter two characters with one SPECINS function, nor can you use the GOLD repeat feature to enter the same character several times in one location.

SPECINS cannot be used if **SET NOREPEAT** is in effect for your editing session.

The maximum decimal character value for SPECINS is 255.

EXAMPLE

Inserts first a <CR>, then an <LF> near the end of the line.

```
for the last time.  Then when you need to get another shipment
```

VT100: GOLD + ! 1 + # 3

VT100: GOLD + CHAR SPECINS

VT100: GOLD + ! 1 + (0

VT100: GOLD + CHAR SPECINS

VT52: GOLD + ! 1 + # 3

VT52: GOLD + RIGHT SPECINS

VT52: GOLD + ! 1 +) 0

VT52: GOLD + RIGHT SPECINS

```
for the last time.  Then when you need to get another <CR><LF>shipment
```

---

related commands

— Nokeypad—ASC, ^ (circumflex)

string specifier—Keypad

SYNTAX

string

DESCRIPTION

The string specifier is generally used either to locate characters in a buffer or to replace the located characters. When a string specifier is used to locate a piece of text, it is referred to as the search string. All three editing modes use search strings. Line mode and nokeypad mode use substitute strings.

Whenever you issue a search string, EDT overwrites the contents of the search buffer. (Similarly, when you issue a substitute string, EDT overwrites the contents of the substitute buffer.)

The search and substitute buffers cannot be edited or entered. Their names never appear in the SHOW BUFFER list. You can use the nokeypad **CLSS** (clear search string) command to delete the contents of the search buffer.

EDT has a number of ways to perform searches. See the discussion of the **SET SEARCH** command for information about the EDT search parameters.

EXAMPLES

example 1:

Moves the cursor to the string **lollypop**.

GOLD + FNDNXT FIND

```
Search for:  lollypop
```

ENTER SUBS

example 2:

Moves the cursor to the next occurrence of the string **lollypop**.

FNDNXT FIND

SUBS (Substitute) Function—Keypad

KEY SEQUENCE

VT100: GOLD + ENTER SUBS or PF1 + ENTER

VT52: GOLD + ENTER SUBS or BLUE COLOR + ENTER

DESCRIPTION

Pressing SUBS (substitute) replaces the current search string with the contents of the PASTE buffer. In order to use SUBS, you must first put the string you want to replace in the search buffer and the new text in the PASTE buffer. All searches and substitutions are made in the current direction.

(CUTSR=DELETE PASTEKS""). is the nokeypad definition for SUBS. This means that the select range—in most cases the current search string—is deleted from the current buffer and placed in a buffer named DELETE. The contents of the PASTE buffer are inserted in the text and the cursor is placed on the last character of the inserted text if EDT's direction is forward. (If EDT's direction is backward, the cursor is positioned on the first character of the inserted text.) Finally, EDT moves to the next occurrence of the current search string.

Using SUBS involves four steps:

1. Put the search string in the search buffer.
2. Put the replacement text in the PASTE buffer.
3. Locate the search string.
4. Press GOLD + SUBS.

1. The easiest way to load the search buffer is with the **FIND** function. You can also use any line mode or nokeypad command that involves a search string. Remember, the search buffer cannot be entered or edited.
2. There are two ways to load the PASTE buffer.
   - You can type the replacement text in your current buffer, make it a select range, and then use CUT to transfer it to the PASTE buffer.
   - Since you can enter the PASTE buffer and edit its contents, you can use the line mode **FIND** command to move to the PASTE buffer and then insert the replacement text there. Use the FIND command to return to the buffer you are editing.
3. If you reverse the order of steps 1 and 2, the cursor will already be at the search string. Otherwise, you must be sure that the cursor is positioned on the first character of the search string before you press SUBS. This is because SUBS performs the substitution first and then moves to the next occurrence of the search string. The order allows you to decide whether you want to perform the substitution on that instance of the search string or go on to the next one. (Use **FNDNXT** to skip the substitution on the current search string match and advance to the next occurrence.)

4 The final step calls for pressing GOLD and then SUBS. If the cursor is not on the search string, EDT prints the message **No select range active**. If there is no other search string match in the remaining portion of your buffer, EDT prints the message **String was not found**.

SUBS is the only substitute function that can use a line terminator in the replacement text.

EXAMPLES

example 1:

First locates the string **languge** in line 2. Then puts the substitute string in the PASTE buffer. Replaces the mistake on line 2 and then the one on line 5.

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a languge that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
languge related to everyday speech.
```

GOLD + FNDNXT FIND

```
Search for:  languge
```

ENTER SUBS

(Now type the string **language** at the current cursor position.)

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a languagelanguge that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
languge related to everyday speech.
```

SELECT RESET + GOLD + 8 + ← + CUT PASTE

GOLD + ENTER SUBS + GOLD + ENTER SUBS

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a language that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
language related to everyday speech.
   .
   .
   .
String was not found
```

example 2:

Using the line terminator in both the search and replacement strings, inserts a dollar sign ($) at the beginning of each line.

```
29.99
35.95
17.00
```

GOLD + FNDNXT FIND

```
Search for:  ^M     (Press RETURN in response to
                     Search for: prompt to put the
                           line terminator in the search buffer.)
```

ENTER SUBS

RETURN + $ 4

(You press RETURN and then $ to put both the line terminator and the dollar sign characters in the PASTE buffer.)

SELECT RESET + ← + ← + CUT PASTE

(Move the cursor back to its original position above the first number.)

GOLD + ENTER SUBS + GOLD + ENTER SUBS + GOLD + ENTER SUBS

```
$29.99
$35.95
$17.00_
```

related commands

- Line—SUBSTITUTE NEXT
- Nokeypad—SN (substitute next)

TAB Function CTRL/I—Keypad

KEY SEQUENCE

VT100:	TAB	TAB
VT52:	TAB	TAB
LK201:	TAB	Tab
All:	CTRL/I	CTRL + I

DESCRIPTION

Pressing TAB (CTRL/I) moves text to the right. The number of column positions that the text moves depends on the cursor position, the **SET TAB** value, if one is in effect, and the indentation level count, if one is in effect. (SET NOTAB is the default.) **TAB.** is the nokeypad definition for TAB.

EDT has preset tab stops every eight characters, regardless of how your terminal is set. If no SET TAB command has been given, pressing TAB moves the cursor character, as well as all the characters on the current line to the right of the cursor, to the nearest preset tab position. Text is always moved to the right, regardless of EDT's current direction.

When a SET TAB value is in effect, TAB moves the entire line to the column designated by the SET TAB value only if the cursor is located in column 1. If the cursor is located anywhere else on the line, TAB moves the text to the nearest preset EDT tab stop.

If a tab indentation level count is in effect and the cursor is located in column one of the line, TAB moves the text to the indentation level position. The indentation level count is determined by three functions: (1) **CTRL/A**, which can be used to compute the indentation level count, (2) **CTRL/D**, which decrements the count, and (3) **CTRL/E**, which increments the count. Use the **SHOW TAB** command to find out the current SET TAB value and the indentation level count.

CTRL/T indents whole lines of text by the SET TAB value.

The TAB key and CTRL/I always have identical functions in EDT. When you redefine the TAB key, you redefine CTRL/I. To redefine the TAB key using the line mode **DEFINE KEY** command, type DEFINE KEY CONTROL I. When you want to find out the definition of the TAB key, type **SHOW KEY** CONTROL I.

EXAMPLE

Using the default EDT tab settings (SET NOTAB in effect), indents the lines of text, each one eight columns farther than the one above.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.
```

(Start with the cursor at the beginning of the second line.)

TAB

(Move the cursor to the beginning of the third line.)

TAB + TAB

(Move the cursor to the beginning of the fourth line.)

TAB + TAB + TAB

```
This is the first line of text.
        This is the second line of text.
                This is the third line of text.
                        This is the fourth line of text.
```

related commands

— Nokeypad—TAB

TOP Function—Keypad

KEY SEQUENCE

VT100: GOLD + BACKUP TOP PF1 + 5

VT52: GOLD + BACKUP TOP BLUE COLOR + 5

DESCRIPTION

Pressing TOP moves the cursor to the first character position at the beginning of the buffer. TOP has no effect on EDT's current direction. **BR.** is the nokeypad definition for TOP.

EXAMPLE

Moves the cursor from the middle of line 3 to the first character position in the buffer.

```
DATE:  June 16, 1985
TO:    Tom Langston
FROM:  Judy Tortini
   .
   .
   .
```

GOLD + BACKUP TOP

```
DATE:  June 16, 1985
TO:    Tom Langston
FROM:  Judy Tortini
   .
   .
   .
```

related commands

— Nokeypad—BR

UND C (Undelete Character) Function—Keypad

KEY SEQUENCE

VT100: GOLD + DEL C / UND C

GOLD + PF1 + ,

VT52: GOLD + DEL C / UND C

GOLD + BLUE / COLOR + 6

DESCRIPTION

Pressing UND C (undelete character) inserts the current contents of the delete character buffer into your text to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is now located on the inserted character if you used DEL C to delete the character. If you used DELETE to delete the character, the cursor is located to the right of the inserted character. **UNDC.** is the nokeypad definition for UND C.

The keypad functions **DEL C** and **DELETE** both place the character they delete in the delete character buffer. Each time you use DEL C or DELETE, the contents of the delete character buffer are overwritten. The buffer contains only the most recently deleted character. When you use a repeat count with DEL C or DELETE, only the last character deleted is in the delete character buffer. If no character has been deleted thus far during the current EDT session, UND C inserts nothing. Note that if you use the DELETE key to delete characters in a command line or prompt line, these characters are *not* stored in the delete character buffer and will not affect the character inserted by UND C.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in your text and you delete it. When you undelete this character, EDT changes the <CR> character into a line terminator and inserts the line terminator in your text.

EXAMPLE

Using DEL C and UND C, reverses the order of the misplaced letters in **poeple** to **people**.

```
Many poeple have been aware of this problem for some time.
```

DEL C / UND C + ←

GOLD + DEL C / UND C

```
Many people have been aware of this problem for some time.
```

related commands

— Nokeypad—UNDC (undelete character)

UND L (Undelete Line) Function—Keypad

KEY SEQUENCE

VT100: GOLD + DEL L / UND L PF1 + PF4

VT52: GOLD + DEL L / UND L BLUE COLOR + GRAY COLOR

DESCRIPTION

Pressing UND L (undelete line) inserts the current contents of the delete line buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to a new line below the current line, if the buffer contents ends with a line terminator. Otherwise, text just moves to the right. The cursor is now located on the first character of the inserted text if you used DEL L or DEL EOL to delete the text. If you used CTRL/U to delete the text, the cursor is located to the right of the inserted text. **UNDL.** is the nokeypad definition for UND L.

The delete line buffer is loaded by using **DEL L**, **DEL EOL**, or **CTRL/U**. Each time one of these three functions is used, the contents of the delete line buffer are overwritten. The current contents of the buffer are the most recently deleted line or line portion. When you use a repeat count with a delete line function, only the last line or line portion that was deleted is in the delete line buffer. If no line has been deleted thus far in your EDT session, UND L inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer.

EXAMPLES

example 1:

Using DEL L and UND L, reorganizes the list so that the cities are in alphabetical order by state.

```
Montgomery, Alabama
Juneau, Alaska
Little Rock, Arkansas
Phoenix, Arizona
```

DEL L / UND L

(Move the cursor to the **L** in **Little Rock**.)

GOLD + DEL L / UND L

```
Montgomery, Alabama
Juneau, Alaska
Phoenix, Arizona
Little Rock, Arkansas
```

---

example 2:

Using DEL L and UND L, inserts the second line of text after **you**.

```
When you give it more thought, please contact me.
have had a chance to
```

DEL L
UND L

(Move the cursor to the **g** in **give**.)

GOLD + DEL L / UND L

```
When you have had a chance to
give it more thought, please contact me.
```

---

example 3:

Uses DEL EOL and UND L to insert several words in the middle of some lines. Note that a leading space is included in the text stored in the delete line buffer.

```
the company.  (Digital Equipment Corporation)
    .
    .
    .
After you have received the proper notification
from the company, you can
```

GOLD + EOL / DEL EOL

(Move the cursor back to the period [.] after the **y** in **company**.)

GOLD + DEL L / UND L

```
the company (Digital Equipment Corporation).
    .
    .
    .
After you have received the proper notification
from the company, you can
```

(Now move the cursor to the comma between **company** and **you**.)

GOLD + DEL L / UND L

```
the company (Digital Equipment Corporation).
    .
    .
    .
After you have received the proper notification
from the company (Digital Equipment Corporation), you can
```

---

related commands

— Nokeypad—UNDL (undelete line)

UND W (Undelete Word) Function—Keypad

KEY SEQUENCE

VT100: GOLD + DEL W / UND W PF1 + –

VT52: GOLD + DEL W / UND W BLUE COLOR + 9

DESCRIPTION

Pressing UND W (undelete word) inserts the current contents of the delete word buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is now located on the first character of the inserted word or word portion if you used DEL W to make the deletion. If you used LINEFEED, the cursor is located to the right of the inserted word or word portion. **UNDW.** is the nokeypad definition for UND W.

The delete word buffer is loaded by using **DEL W** or **LINEFEED** (CTRL/J, F13—LK201). Each time you use DEL W or LINEFEED, the contents of the delete word buffer are overwritten. The current contents of the buffer are the most recently deleted word or word portion. When you use a repeat count with a delete word function, only the last word or word portion deleted is in the delete word buffer. If no word has been deleted thus far in your EDT session, UND W inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer.

EXAMPLE

Using DEL W and UND W, inserts the word **PRINT** at the beginning of lines 1, 2, and 3.

```
LETTER1.MEM
LETTER2.MEM
LETTER3.MEM
PRINT LETTER4.MEM
```

DEL W / UND W

(Move the cursor to the **L** in **LETTER1.MEM**.)

GOLD + DEL W / UND W + ↓ + GOLD + DEL W / UND W + ↓

GOLD + DEL W / UND W + ↓ + GOLD + DEL W / UND W

```
PRINT LETTER1.MEM
PRINT LETTER2.MEM
PRINT LETTER3.MEM
PRINT LETTER4.MEM
_
```

related commands

— Nokeypad—UNDW (undelete word)

Up Arrow—Keypad

KEY SEQUENCE

VT100: [↑] [↑]

VT52: [UP REPLACE] [↑]

DESCRIPTION

Pressing the Up Arrow key moves the cursor up one line toward the top of the buffer regardless of EDT's direction. **-V.** is the nokeypad definition for Up Arrow.

When you use the Up Arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of the short line. If you continue to use the Up Arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for the Up Arrow and resets it the next time you use the function.

EXAMPLE

Moves the cursor from the end of the last line to the end of the first line.

```
La Paz, Bolivia
Lima, Peru
Montevideo, Uruguay
Parimaribo, Surinam
Quito, Ecuador
Santiago, Chile_
```

[↑]

```
Quito, Ecuador_
Santiago, Chile
```

[↑] + [↑]

```
Montevideo, Uruguay
Parimaribo, Surinam
Quito, Ecuador
```

[↑] + [↑]

```
La Paz, Bolivia_
Lima, Peru
Montevideo, Uruguay
```

related commands

— Nokeypad—Up Arrow

WORD Function—Keypad

KEY SEQUENCE

VT100: WORD CHNGCASE 1

VT52: WORD CHNGCASE 1

DESCRIPTION

Pressing WORD moves the cursor to the beginning of the next word in the current direction (forward or backward, depending on whether **ADVANCE** or **BACKUP** is in effect). **W.** is the nokeypad definition for WORD.

An EDT word is any group of characters bounded by a space, horizontal tab, line feed <LF>, vertical tab <VT>, form feed <FF>, or carriage return <CR>. You can establish different word boundaries with the line mode **SET ENTITY** WORD command. Use the line mode **SHOW ENTITY** WORD command to find the current boundary markers for the word entity.

The **SET WORD [NO]DELIMITER** command affects how EDT interprets word boundaries. With SET WORD DELIMITER (the default) in effect, EDT considers all word boundaries, except the space, as words themselves.

EXAMPLE

Moves the cursor four words to the right.

```
More and more businesses are using computers.
```

WORD CHNGCASE + WORD CHNGCASE + WORD CHNGCASE + WORD CHNGCASE

```
More and more businesses are using computers.
```

related commands

— Nokeypad—W (word)

/BRIEF Qualifier—Line

SYNTAX

/BRIEF*[:n]*

DESCRIPTION

The /BRIEF qualifier is used with the line mode **SUBSTITUTE** and **TYPE** commands. All EDT qualifiers must be preceded with slashes.

/BRIEF instructs EDT to display only the first **n** characters of a line, instead of the entire line. The default value for **n** is 10.

EXAMPLES

example 1:

Instructs EDT to display only the first 15 characters of the line in which the substitution took place.

```
3       Mr. Arlen J. Coolidge
4       3829 Gardner Avenue
5       Hollywood, FL  33021

*SUBSTITUTE*J.*R.* /BRIEF:15
    3       Mr. Arlen R. Co
1 substitution
```

example 2:

Instructs EDT to display only the first 10 characters of line 5.

```
3       Mr. Arlen R. Coolidge
4       3829 Gardner Avenue
5       Hollywood, FL  33021

*TYPE 5 /BRIEF
    5       Hollywood,
```

buffer specifier—Line

SYNTAX

buffer name
=buffer name
BUFFER *buffer name*

DESCRIPTION

The buffer specifier consists of two elements: (1) the signal to EDT that the following characters are a buffer name and (2) the buffer name itself. There are three syntax forms. The first syntax form does not have the signal element and can be used only with the **CLEAR** command. You can use the second and third forms— =buffer and BUFFER buffer—in all other line mode commands that can take a buffer specifier.

The specifier **buffer** refers to the name of a location used by EDT to store text during an editing session. When you begin your EDT session, a copy of your file is put in the buffer called MAIN. EDT also creates a buffer called PASTE for use with the **CUT**, **PASTE**, and **APPEND** commands. When you exit from EDT using the **EXIT** command, a copy of the MAIN buffer text is put in an external file. All other buffers created during your editing session disappear. If you use **QUIT** to leave EDT, no copy of the MAIN buffer is made.

You create additional buffers during an EDT session each time you specify a new buffer name in a line or nokeypad command. For example, the command **FIND =LIST** creates a buffer called LIST. There are no keypad functions that create new buffers, but you can use line mode commands while still in keypad mode to create new buffers or move from one buffer to another.

The CLEAR command deletes buffers and their contents from your EDT session. In the case of the MAIN and PASTE buffers, only the contents can be deleted, not the buffers themselves. To delete only the contents of buffers other than MAIN or PASTE, use the line mode **DELETE** command.

Buffer names can have more than 80 alphanumeric characters. Only letters, digits, and the underscore character (_) can be used to create buffer names. The first character in the name must always be a letter.

You can edit any buffer you create, as well as the MAIN and PASTE buffers. Text can be copied or moved from one buffer to another. The **SHOW BUFFER** command lists all the accessible buffers in your editing session and indicates the current buffer by preceding the name with an equal sign (=).

EDT has several storage areas that you cannot access. These include the delete character, delete line, delete word, search, and substitute buffers. The first three are used only in keypad and nokeypad editing. These buffers contain the most recent character, line, or word deleted by the respective delete functions. The search buffer is used by the various find, substitute, and change case functions. The substitute buffer is used only by substitute commands. Storage areas also exist for entity and prompt definitions. Although you have some control over what goes into these various buffers, you cannot enter them or edit them, and their names never appear in the SHOW BUFFER list.

EXAMPLES

1 `*FIND =ADDRESS1`

Moves to the first line of buffer called ADDRESS1. Displays nothing.

2 `*COPY 10 THRU 100 TO =TEMP`

Copies lines 10 through 100 in the current buffer to the buffer named TEMP and moves to TEMP.

3 `*INCLUDE DISTLIST.DAT =DISTLIST`

Copies the external file DISTLIST.DAT into the buffer called DISTLIST and moves to that buffer.

4 `*MOVE =HOLD 1 THRU 32 TO 88`

Deletes lines 1 through 32 in the buffer HOLD and places them above line 88 in the current buffer.

5 `*WRITE LETTER.RNO BUFFER EXTRA`

Copies the entire contents of the buffer EXTRA to the file LETTER.RNO. EDT remains in the current buffer.

6 `*CLEAR JUNK`

Deletes the buffer called JUNK. If JUNK is the current buffer, EDT moves to the MAIN buffer.

CHANGE Command—Line

SYNTAX

CHANGE *[=buffer] [range] [;nokeypad command(s)]*

DESCRIPTION

The CHANGE command transfers your editing session from line mode to keypad, nokeypad, or hardcopy change mode. The buffer and range specifiers determine the cursor position when EDT finishes processing the CHANGE command. The default cursor position is at the beginning of the current line in the current buffer.

When the range specifier is a line number, the cursor appears at the beginning of the line. If you use a string for the range specifier, EDT positions the cursor on the character immediately following the string.

You can include nokeypad commands on the CHANGE command line regardless of which change mode you are accessing. A semicolon (;) separates the command word CHANGE and any location specifiers from the nokeypad command(s).

The default mode with CHANGE is keypad for VT100 and VT52 terminals. To go from line to keypad mode, simply issue the CHANGE command. To use nokeypad mode, you must first give the **SET NOKEYPAD** command, then the CHANGE command. Once SET NOKEYPAD is in effect, you must use the **SET KEYPAD** command with the CHANGE command to enter keypad mode. If your terminal's setting for EDT is hardcopy (HCPY), the CHANGE command shifts EDT to hardcopy change mode.

EXAMPLES

example 1:

Shifts to keypad mode after the EDT session starts off in line mode.

```
$ EDIT /EDT MEMO.RNO
    1       .FILL.JUSTIFY
*CHANGE

.FILL.JUSTIFY
.LEFT MARGIN0
.RIGHT MARGIN70
.PAGE SIZE 58,70
    .
    .
    .
```

example 2:

Shifts to nokeypad mode after the EDT session starts in line mode.

```
$ EDIT /EDT MEMO.RNO
    1       .FILL.JUSTIFY

*SET NOKEYPAD
*CHANGE
```

```
.FILL.JUSTIFY
.LEFT MARGIN0
.RIGHT MARGIN70
.PAGE SIZE 58,70
   .
   .
   .
```

example 3:

Shifts to keypad mode and moves to the buffer TESTING.

```
*CHANGE =TESTING
```

example 4:

Shifts to keypad mode and moves the cursor to the ninth line of the current buffer.

```
*CHANGE 9
```

example 5:

Shifts to keypad mode and moves the cursor to the beginning of the fourth paragraph.

```
*CHANGE ;3PAR
```

example 6:

Shifts to hardcopy change mode. Assumes that the terminal setting is hardcopy (HCPY).

```
*CHANGE
[.]FILL.JUSTIFY
C*
```

related commands

- Keypad—CTRL/Z
- Nokeypad—EX (exit to line mode)

CLEAR Command—Line

SYNTAX

CLEAR *buffer*

DESCRIPTION

The CLEAR command deletes the entire contents of the specified buffer. The buffer name must be supplied, even if it is the current buffer. When CLEAR deletes the buffer, it removes the buffer name from the list of active buffers for your session. The MAIN and PASTE buffers are exceptions. You can delete the contents of these buffers, but not their names and storage locations.

When you use CLEAR to delete the current buffer, EDT shifts to the most recent current line in the MAIN buffer.

See the entry on the buffer specifier for more information on how EDT uses buffers.

EXAMPLE

First deletes the contents and location of the current buffer named DISCARD. Then deletes the contents of the PASTE buffer.

```
*SHOW BUFFER
=DISCARD        8       lines
 EXTRA  10      lines
 MAIN   256     lines
 PASTE  1       lines
*CLEAR DISCARD
*SHOW BUFFER
 EXTRA  10      lines
=MAIN   256     lines
 PASTE  1       lines
*CLEAR PASTE
*SHOW BUFFER
 EXTRA  10      lines
=MAIN   256     lines
 PASTE  No      lines
```

related commands

None.

COPY Command—Line

SYNTAX

COPY *[=buffer-1] [range-1]* **TO** *[=buffer-2] [range-2]* *[/**QUERY**] [/**DUPLICATE**:n]*

DESCRIPTION

The COPY command copies the text specified by location-1 (buffer-1 and/or range-1) and places it directly above the line specified by location-2 (buffer-2 and/or range-2). No text is deleted. You can copy to and from different buffers, creating new buffers as appropriate.

If location-1 is omitted, EDT copies the current line. When location-2 is omitted, EDT inserts a copy of the text above the current line. Whenever you omit the buffer specifier, EDT assumes the current buffer. If buffer-1 is specified without a range-1 specifier, the entire contents of buffer-1 are copied to location-2. If buffer-2 is specified without a range-2 specifier, a copy of the text is inserted at the beginning of buffer-2.

Range-1 can refer to one or more lines in the buffer, but range-2 is limited to a single line in the current or specified buffer. Remember to leave spaces between the buffer name and any range specifier if one is used. Otherwise, EDT might misinterpret the buffer name.

If you do not include a buffer-2 specifier, then EDT completes the COPY command in the same buffer from which you issued the command. When you do specify buffer-2, that buffer becomes the current buffer.

The **/QUERY** qualifier lets you verify each line to be copied to location-2. EDT prompts you with a question mark (?) after displaying a line from location-1 to determine whether you want to copy that line. There are four responses to the question mark prompt: Y (YES), N (NO), A (ALL), and Q (QUIT).

EXAMPLES

1
```
*COPY 11 THRU 14 TO 45
```
Puts a copy of lines 11 through 14 above line 45.

2
```
*COPY =MAIN TO =OTHER
```
Puts a copy of the contents of the MAIN buffer into the buffer called OTHER. If the OTHER buffer does not already exist, EDT creates it.

3
```
*COPY 1 TO 8 /DUPLICATE:3
```
Puts three copies of line 1 above line 8.

4
```
*COPY 4 THRU 7 TO END
```
Puts a copy of lines 4 through 7 in the current buffer after the last line of the current buffer.

5 `*COPY "Date" THRU "Dear Sir:"+1 TO =MAIN "Enclosed"`

Puts a copy of the lines in the current buffer starting with the line containing the word **Date** through the line following the line containing the words **Dear Sir:** immediately above the line containing the word **Enclosed**, located in the MAIN buffer.

6 `*COPY 16 THRU 20 TO 58 /QUERY`

Prints line 16 and then questions you to be sure you want to copy that line. If the answer is Y (YES), line 16 is copied above line 58 and line 17 is displayed. If the answer for line 16 is N (NO), line 16 is not copied; EDT moves to line 17 and displays it. If the answer for line 16 is A (ALL), lines 16 through 20 are copied to the location above line 58 without further prompts. If the answer for line 16 is Q (QUIT), no lines are copied.

7 `*COPY =ADDR 12 THRU 16 TO .`

Puts a copy of lines 12 through 16 from the buffer ADDR into the current buffer immediately above the current line.

8 `*COPY =MAIN 9 THRU 47 TO =BROWN 9`

Puts a copy of lines 9 through 47 from the MAIN buffer into the buffer named BROWN immediately above line 9 in that buffer. The new current line is line 9 in the BROWN buffer.

---

related commands

- Keypad—CUT + PASTE [+ "move"] + PASTE
- Nokeypad—CUT + PASTE [+ "move"] + PASTE

CTRL/C (Control C) Function—Line

KEY SEQUENCE

CTRL/C

DESCRIPTION

CTRL/C interrupts certain operations before EDT finishes processing them. You can use CTRL/C to stop a runaway search through a long file or to stop a long repeat count. CTRL/C halts certain EDT operations. For example, you can use CTRL/C to stop EDT from printing a whole buffer when you have used the line mode **TYPE** command to move to another buffer.

When EDT aborts the operation, it prints the message **Aborted by CTRL/C**. If EDT cannot stop a particular process, it prints the message **CTRL/C ignored**.

EXAMPLE

Stops EDT from printing the entire contents of the buffer DARCY_LET.

```
*SHOW BUFFER
 DARCY_LET      40      lines
=MAIN   12      lines
 PASTE  No      lines
*TYPE =DARCY_LET

    1       August 20, 1985
    2
    3       Mr. Charles R. Darcy
    4       Production Manager
    5       Midland Manufacturing Corporation
    6       East St. Louis, IL  62202

CTRL/C


Aborted by CTRL/C
```

related commands

- Keypad—CTRL/C
- Nokeypad—CTRL/C

CTRL/R (Control R) Function—Line GOLD/R

KEY SEQUENCE

CTRL/R

DESCRIPTION

In line mode, CTRL/R refreshes the line you are currently typing. If you have used the DELETE key several times to make corrections, you can use CTRL/R to see the characters that are presently part of the line. After using CTRL/R, EDT is still positioned on the line you were typing. You can continue typing on that line or press RETURN to send the data to EDT.

EXAMPLE

Refreshes the command line to clear the deletions.

```
*MOVE 1 THRU 10 TO =NAUB
```

DELETE + DELETE + DELETE + DELETE

```
*MOVE 1 THRU 10 TO =NAUB\\\\
```

CTRL/R

```
*MOVE 1 THRU 10 TO =NAUB\\\\
*MOVE 1 THRU 10 TO =
```

related commands

- Keypad—CTRL/R
- Nokeypad—REF (refresh)

CTRL/Z (Control Z) Function—Line

KEY SEQUENCE

CTRL/Z

DESCRIPTION

CTRL/Z performs the same function in line editing as it does in nokeypad editing. (It has a different function in keypad editing.) You use CTRL/Z to exit from the insert state. After you have given the appropriate insert command and typed the text you want to add, use CTRL/Z to complete the insert procedure. CTRL/Z is used with both the **INSERT** and **REPLACE** commands in line mode and the **I** (insert) and **R** (replace) commands in nokeypad mode.

Pressing CTRL/Z three times in succession has the same effect as the QUIT /SAVE command.

EXAMPLE

In line mode, completes the insert operation and returns EDT to the command state. Uses the TYPE command to display the first two lines in the buffer.

```
1       TO:    Kyle Carlson

*INSERT

            DATE:  March 31, 1986

CTRL/Z


          ^Z
    1       TO:    Kyle Carlson

*TYPE 0 THRU 1

    0.1     DATE:  March 31, 1986
    1       TO:    Kyle)Carlson
```

related commands

— Nokeypad—CTRL/Z

DEFINE KEY Command—Line

SYNTAX

DEFINE KEY [GOLD] *keypad-key-number* **AS** *"string[.]"*

DEFINE KEY GOLD *character* **AS** *"string[.]"*

DEFINE KEY [GOLD] CONTROL *character* **AS** *"string[.]"*

DEFINE KEY [GOLD] DELETE AS *"string[.]"*

DEFINE KEY [GOLD] FUNCTION *key-number* **AS** *"string[.]"*

DESCRIPTION

The DEFINE KEY command defines or redefines function keys used in keypad editing. Key definitions are based on nokeypad commands.

Five types of keys can be defined or redefined:

- A keypad key with or without GOLD

  All keypad keys can be defined.

- CONTROL with a keyboard character, with or without GOLD

  EDT does not allow you to redefine CTRL/C. Some CONTROL character combinations are system commands and therefore cannot be redefined. These include O, P, Q, S, X, Y, and [.

- GOLD with a keyboard character

  GOLD can be used with any keyboard character except the digits 0 through 9 and the minus sign.

- The DELETE key with or without GOLD

  The DELETE key can be redefined by itself or with GOLD.

- FUNCTION keys on the LK201 keyboard.

  These include the six keys located above the arrow keys on the terminal's "editing" keypad as well as keys F6 through F20 on the function key row across the top of the keyboard.

When using the DEFINE KEY command you must type the words GOLD and CONTROL in your commands. Pressing the GOLD or CTRL keys has no effect in line mode. If you are redefining the DELETE key, you must type the word DELETE. To redefine a function key on the LK201 keyboard, type the word FUNCTION.

When you want to change the definition for the **BACKSPACE**, **LINEFEED** or **TAB** key, you must redefine its CONTROL equivalent: CONTROL H for BACKSPACE, CONTROL J for LINEFEED, and CONTROL I for TAB. For terminals with LK201 keyboards, you can redefine the F12 (BACKSPACE) and F13 (LINEFEED) keys using their function numbers, FUNCTION 24 and FUNCTION 25.

Keypad-key-number refers to EDT's numerical designations for the keypad keys. For example, the period key on the keypad has number 16, the ENTER key has number 21, and the down arrow key has number 13.

FUNCTION keys are the additional keys on the LK201 keyboard that you can define. These include the six keys located above the arrow keys on the "editing" keypad, as well as the keys on the function key row from F6 through F20. You cannot define keys F1 through F5 on the function key row. EDT uses its own numbers for these keys, ranging from FUNCTION 1 through FUNCTION 99, to encompass the special user defined keys (UDKs) that are allowed on terminals with LK201 keyboards.

String is the key definition. Key definitions are composed of nokeypad commands and specifiers. The definition for a single function key can contain several nokeypad commands. An individual nokeypad command can be separated from another nokeypad command with a space, for example **3W 4C 2DC.** or **+V BL I$_^Z.**. However, you cannot have spaces within a single nokeypad command. (D3W is valid; D 3W is not.)

The optional period determines whether the defined function is automatically processed by simply pressing the key or key sequence. If you omit the period, EDT waits until it receives the **ENTER** signal before processing the function. (The period [.] is the definition of ENTER.) Most preset key definitions end with the period. The exceptions are **GOLD** and **RESET**, which are not nokeypad commands. If you include the period in your definition, be sure to use the keyboard period key, not the keypad one.

If your definition includes more than one nokeypad command, you might want to enclose the definition in parentheses. The parentheses enable you to use the keypad GOLD/repeat feature to process your function more than once. If you try to use a repeat count on a multicommand definition with no parentheses, EDT repeats only the first command. Be sure to put the period outside the parentheses, though. For example, **(BL V W DW).** deletes the second word in a line. You cannot use a repeat count with the nokeypad **EXT** (extend) command.

EXAMPLES

1
```
*DEFINE KEY 7 AS "+PAR."
```
Moves the cursor forward to the beginning of the next paragraph.

2
```
*DEFINE KEY CONTROL E AS "EXT EXIT."
```
Exits from EDT completely in one step, without having to press several keys and type EXIT.

3 `*DEFINE KEY GOLD P AS "CUTSR PASTE."`

Deletes the select range and pastes it back in the same place, while retaining a copy of the select range in the PASTE buffer for inserting elsewhere during your editing session.

4 `*DEFINE KEY CONTROL A AS "CUTSR I^&^Z PASTE I\&^Z."`

Puts RUNOFF underline flags around a select range.

5 `*DEFINE KEY CONTROL V AS "I!!!^Z ^M EXT EXIT."`

Inserts a mark (!!!) in your file and exits from EDT. The mark makes it easy for you to find the place where you were last working.

6 `*DEFINE KEY GOLD 3 AS "?'ENTER NOKEYPAD COMMAND: '."`

Prompts the user with the message in single quotes. When you type the nokeypad command(s) and press ENTER, EDT processes the command(s).

7 `*DEFINE KEY GOLD CONTROL B AS "(+V D-C I|^Z.")`

Instructs EDT to move down one line, delete the character preceding the cursor, and then insert the vertical bar character. This can be used to draw vertical lines in your text.

8 `*DEFINE KEY 8 AS "+D"`

Deletes the entity specified by the subsequent key pressed. For example, if the word key is pressed after 8, the next word is deleted. The absence of the period at the end of the definition means that nothing will happen when the 8 key is pressed by itself. The key pressed *after* the 8 will cause a change in the text to occur.

---

related commands

— Keypad—CTRL/K

— [Nokeypad—DEFK]

DEFINE MACRO Command—Line

SYNTAX

DEFINE MACRO *macro-name*

DESCRIPTION

The DEFINE MACRO command is the part of EDT's macro facility that enables you to create new line mode commands for the duration of your editing session. When you issue the DEFINE MACRO command, EDT adds the macro name to the list of valid line mode commands. The macro name must be the same as the name of the buffer that contains the EDT macro. When you issue the macro name as a line mode command, EDT performs all the line mode commands that constitute the macro.

A macro is a group of one or more line mode commands that EDT performs when you issue the macro name as a line mode command. Creating an EDT macro takes several steps. Start by using the **FIND** commd to move to the buffer with the macro name. Then insert the line mode command(s) that constitute that macro. You can issue the DEFINE MACRO command either before or after you put the macro text into its buffer. Once the macro text has been entered into its buffer and you have issued the DEFINE MACRO command, you are ready to use the macro name as a line mode command.

Macro names can be the same as existing EDT line mode commands, but in those cases, EDT redefines the command to be that of the macro for the remainder of the session or until the macro is deleted with the **CLEAR** command. To delete a macro, simply use the macro name as the buffer specifier with CLEAR.

A macro can be saved in an external file and copied into a buffer during your editing session with the **INCLUDE** command. You can put a DEFINE MACRO command in your startup command file, along with the text of the macro and the necessary commands to create the macro.

EXAMPLES

example 1:

Defines and creates a macro called FORM that activates three SET commands. Then returns to the MAIN buffer.

```
*DEFINE MACRO FORM
*FIND =FORM
*INSERT
          SET TAB 5
          SET ENTITY SENTENCE '.?;'
          SET WRAP 70

CTRL/Z

*FIND =MAIN
```

example 2:

Defines and creates a macro called MNH which inserts the text **Merrimack, NH 03054**. Then returns to the MAIN buffer.

```
*DEFINE MACRO MNH
*FIND =MNH
*INSERT
                INSERT ;Merrimack, NH 03054

CTRL/Z

*FIND =MAIN
```

example 3:

Shows the portion of a startup command file that contains line mode commands to create a macro called MEMO. The MEMO macro inserts the heading for a standard memo. The second FIND command at the end of the example returns EDT to the MAIN buffer so you can begin to type the memo text. Lines beginning with exclamation points are comments in the startup command file. Note that the single-line form of the line mode INSERT command must be used to create the macro text.

```
!
! Creates the macro called MEMO, which contains
!   the heading for a standard memo.
!
FIND =MEMO                      !Moves to the macro buffer.
INSERT; INSERT;                 !Blank line
INSERT; INSERT;                 !Blank line
INSERT; INSERT; DATE:
IERT; INSERT;                 !Blank line
INSERT; INSERT; TO:
INSERT; INSERT;                 !Blank line
INSERT; INSERT; FROM:
INSERT; INSERT;                 !Blank line
INSERT; INSERT; SUBJECT:
DEFINE MACRO MEMO               !MEMO is now a line mode command.
FIND =MAIN                      !Returns to the first line of MAIN.
!
!End of macro
```

related commands

None.

DELETE Command—Line

SYNTAX

DELETE *[=buffer] [range] [/**QUERY**]*

DESCRIPTION

The DELETE command deletes a line or group of lines, depending on the range that you specify. If no buffer or range is specified, EDT deletes the current line. When you use a buffer specifier but omit the range specifier, EDT moves to that buffer and deletes its entire contents. Whenever you include a buffer specifier, that buffer becomes the current buffer. When you include a range specifier with the DELETE command, EDT deletes all the lines included in that range.

The DELETE command always deletes entire lines, even if the range specifier is a string. In that case, DELETE deletes the line(s) containing that string.

The **/QUERY** qualifier allows you to verify deletions line by line. When /QUERY is in effect, EDT prints a line and then the question mark prompt (?) to ask if you want that line deleted. The valid responses to the question mark prompt are: Y (YES), N (NO), A (ALL), and Q (QUIT).

EXAMPLES

example 1:

Deletes the current line.

```
    8       2:30  Meeting with Paul Jeffreys

*DELETE
1 line deleted
    9       3:00 Party for Janice Kay.
```

example 2:

Uses the /QUERY qualifier to delete some, but not all, of the lines in the range 1 through 5.

```
    1       Calendar for Tuesday, Dec. 13
    2       Staff Meeting 9:00 a.m., Glen Room
    3       EDT Lecture, 11:00 a.m., Merrimack Auditorium
    4       Luncheon Seminar, 12:00 noon, Hanover Room
    5
    6       Calendar for Wednesday, Dec. 14

*DELETE 1 THRU 5 /QUERY
    1       Calendar for Tuesday, Dec. 13
? Y
    2       Staff Meeting 9:00 a.m., Glen Room
? Y
    3       EDT Lecture, 11:00 a.m., Merrimack Auditorium
? N
    4       Luncheon Seminar, 12:00 noon, Hanover Room
? A
4 lines deleted
    6       Calendar for Wednesday, Dec. 14
```

example 3:

Moves to the buffer ADDRS and deletes lines 1 through 4 in that buffer.

```
*DELETE =ADDRS 1 THRU 4
4 lines deleted
    5       Dr. Janet Townley
```

related commands

- Keypad—DEL L
- Nokeypad—DL (delete line)

/DUPLICATE Qualifier—Line

SYNTAX

/DUPLICATE:*n*

DESCRIPTION

/DUPLICATE is a qualifier used with the line mode **COPY** command. All EDT qualifiers must be preceded by slashes. The /DUPLICATE qualifier instructs EDT to copy the specified text **n** times above the location-2 specified in the COPY command. You must supply a value for **n** between 1 and 32767.

EXAMPLE

Inserts five copies of line 3 above line 14.

```
*COPY 3 TO 14 /DUPLICATE:5
1 line copied 5 times
*TYPE 13 THRU 14

   13       1 pkg. #2 copper tubing
   13.1     1 pkg. #3 copper tubing
   13.2     1 pkg. #3 copper tubing
   13.3     1 pkg. #3 copper tubing
   13.4     1 pkg. #3 copper tubing
   13.5     1 pkg. #3 copper tubing
   14       1 pkg. #4 copper tubing
```

EXIT Command—Line

SYNTAX

EXIT *[file-specification]*

*[/***SEQUENCE***[:initial[:increment]]] [/***SAVE***]*

DESCRIPTION

The EXIT command ends your editing session. EDT creates an external file and copies the contents of the MAIN buffer into that file. When you type EXIT with no file specification, EDT uses the file specification information supplied in the EDIT/EDT command line. The file specification that you supply with the EXIT command supersedes any output file specification given in the EDIT/EDT command line. If the output file was only partially specified in the EDIT/EDT command line or EXIT command, EDT uses the input file name or file type in place of whichever item is missing.

If a directory name is included in the file specification, that directory must already exist and you must have access to it. When you use the EXIT command, EDT creates a file. However, EDT cannot create a directory.

The **/SEQUENCE** qualifier instructs EDT to retain the EDT line numbers for use in subsequent editing sessions. Only whole numbers (that is, line numbers with no decimals) can be used for the :initial and :increment specifiers. When /SEQUENCE is used with no specifiers, EDT attempts to retain the existing line numbers insofar as it can. But all line numbers with decimal fractions are adjusted to whole numbers.

The **/SAVE** qualifier tells EDT to save the copy of the journal file in the current directory. The default journal file specification has the same file name as the input file. **.JOU** is the default file type. Unless you use the /SAVE qualifier in your EXIT command, EDT automatically deletes the journal file when you issue the EXIT command. The journal file contains all the keystrokes you made during the editing session. You can specify a different journal file name in the EDIT/EDT command line.

The EXIT command tries to apply the attributes of the input file to the output file it creates. If these attributes have been violated by any editing work done during the session (for example, altering the length of a record in a fixed length record file on some systems), EDT might not be able to finish copying the MAIN buffer text into the output file. In these instances, EDT displays the error detected by the appropriate file service and discontinues processing the EXIT command. You can use the WRITE command to copy the text to an external file, but the attributes of the input file are no longer preserved. The WRITE command always creates an external file with EDT's default attributes. (Then use the QUIT command to exit from EDT after you have issued the WRITE command.)

EXAMPLES

example 1:

Shows the beginning and end of an EDT session. The input file is WKREP25.RNO and the output file is WKREP25.RNO;2. The system command DIRECTORY shows that both files exist in the directory after the EXIT command is used.

```
$ EDIT /EDT WKREP25.RNO
    1       .FILL.JUSTIFY
    .
    .
    .
*EXIT
DISK$USER:[SMITH:STATUS]WKREP25.RNO;2  63 lines
$ DIRECTORY WKREP25.RNO
     WKREP25.RNO;1          WKREP25.RNO;2
```

example 2:

Shows the beginning and end of an EDT session. The input file is LETTER.RNO;4 and the output file is SANCHEZ.RNO;1. The system command DIRECTORY shows that both files exist in the directory after the EXIT command is used.

```
$ EDIT /EDT LETTER.RNO;4
    1       .FILL.JUSTIFY
    .
    .
    .
*EXIT SANCHEZ.RNO
DISK$USER:[MARSDEN.LETTERS]SANCHEZ.RNO;1 37 line
$  DIRECTORY
    .
    .
    .
     LETTER.RNO;4
    .
    .
    .
     SANCHEZ.RNO;1
```

example 3:

Shows the beginning and end of an EDT session. The input file is EOL.RNO and the output file is ENDOFLINE.RNO. The /SAVE qualifier causes the journal file to be saved so this editing work can be repeated with other files. Uses the DCL DIRECTORY command to show the journal file.

```
$ EDIT /EDT EOL.RNO
    1       .FILL.JUSTIFY.LEFT MARGIN0.RIGHT MARGIN70
*SUBSTITUTE/GIN0/GIN10/
    1       .FILL.JUSTIFY.LEFT MARGIN10.RIGHT MARGIN70
1 substitution
*SUBSTITUTE/70/80/
    1       .FILL.JUSTIFY.LEFT MARGIN10.RIGHT MARGIN80
1 substitution
*INSERT ;.PAGE
    1       .FILL.JUSTIFY.LEFT MARGIN10.RIGHT MARGIN80
*EXIT ENDOFLINE.RNO /SAVE
DISK$USER:[SMITH.CHAPTER8]ENDOFLINE.RNO;1 54 lines
$ DIRECTORY *.JOU
    EOL.JOU;1
```

related commands

None.

FILL Command—Line

SYNTAX

FILL *[=buffer] [range]*

DESCRIPTION

The FILL command takes a select range of lines and reorganizes the text so that the maximum number of whole words can fit within the current line width. The default line width for EDT is the terminal width that the operating system passes to EDT. Use the line mode **SHOW SCREEN** command to find out the current screen/line width.

You can use either the **SET WRAP** or **SET SCREEN** line mode command to change the line width for filling. SET SCREEN changes the line width for all operations in your editing session. SET WRAP affects only filling. EDT uses the SET SCREEN width for filling if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses that value, regardless of the SET SCREEN width. Both SET SCREEN and SET WRAP take the line length in characters as the command specifier. For example:

```
SET SCREEN 40
SET WRAP 60
```

Use the **SHOW WRAP** command to find the current wrap value or setting.

The fill process considers a blank line to be a break between paragraphs. Even if there are spaces on the blank line, EDT fills the text up to the blank line and then resumes filling with the next line that contains text.

If you do not specify a range or buffer with the FILL command, EDT assumes that a select range has been established in one of the screen modes. If no select range is active, EDT displays an error message.

If your SET SCREEN width is 80, EDT will fill lines to column 79; if your SET SCREEN width is 132, EDT will fill lines to column 131.

EXAMPLE

Uses the SET WRAP command to limit the line length to 40 characters. Then reformats lines 10 through 15.

```
   10       With regard to your letter of January 4th,
   11       I am unable to satisfy your need for terminals
   12       that scroll sideways.  To the best of my knowledge,
   13       no such terminals are available on today's market.
   14       We do, however,
   15       have an editor named EDT with this capability.

*SET WRAP 40
*FILL 10 THRU 15

   10       With regard to your letter of January
   11       4th, I am unable to satisfy your need
   12       for terminals that scroll sideways.  To
   13       the best of my knowledge, no such
   14       terminals are available on today's
   15       market. We do, however, have an editor
   16       named EDT with this capability.
```

related commands

- Keypad—FILL
- Nokeypad—FILL

FIND Command—Line

SYNTAX

FIND *[=buffer] [range]*

DESCRIPTION

The FIND command moves EDT to a new position in the current buffer or in the specified buffer. In line mode, once the specified line has been found, EDT signals with the line mode prompt (*). The new current line is not displayed at the terminal. (Use the **TYPE** command if you want to see the line.)

If you use the FIND command from keypad or nokeypad mode, the cursor moves to the first character of the specified line. If you use a string for range, the cursor moves to the character to the right of the search string.

When FIND is used with a buffer specifier but no range specifier, EDT positions itself at the first character of the specified buffer. When you use both buffer and range specifiers, EDT positions itself at the line you specified as the range in the specified buffer. Use a space to separate the buffer name from the range specifier if the range specifier begins with a letter or digit (for example, =PASTE 1). Otherwise, EDT will interpret the range as part of the buffer name.

With the FIND command, you can search either forward or backward for a string in the current buffer. To search backward toward the beginning of the buffer, you must precede the search string with a minus sign (–). When you use a line number as the range specifier, FIND will locate the line regardless of direction.

If you use a string as the range specifier, you should be aware of how EDT searches for a matching string in the text. EDT performs searches in several ways. The default ways are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search on the portion of the buffer between the cursor position and the beginning or end of the buffer, depending on the direction of the search. Use the **SET SEARCH** command to change the way EDT performs searches. The **SHOW SEARCH** command tells you which search parameters are currently in effect. (BEGIN is applicable only in change mode.)

The FIND command is frequently used to move EDT from one buffer to another. When you type FIND =buffer name, EDT moves to the first line of the specified buffer. If that buffer does not already exist, EDT creates the buffer.

You can use the period range specifier (.) to return to the last line you were working on in a previously used buffer. For example, suppose 254.2 was the last line EDT was at in the MAIN buffer. When you finish your editing in the TEST buffer, type **FIND =MAIN.** to return to the MAIN buffer. EDT positions itself at line 254.2 in the MAIN buffer. (You can place the period immediately after the buffer name without confusing EDT, because periods are not valid characters in buffer names.)

EXAMPLES

example 1:

Moves EDT to line 15, but prints nothing. The new location is correct for the SUBSTITUTE command.

```
   11       In regard to your letter of the 24th, we are planning

*FIND 15
*SUBSTITUTE/December/January/

   15       I will be flying to Denver on January 15th, arriving at
1 substitution
```

example 2:

Moves EDT to the next line containing the string **Bob** but prints nothing. The new location is correct for the SUBSTITUTE command.

```
*FIND "Bob"
*SUBSTITUTE/Bob/Harry/

   21       with Harry at the next Board of Directors Meeting.
1 substitution
```

example 3:

Moves to the end of the buffer so that you can insert a new line at the end of the text.

```
*FIND END
*INSERT; cc:  John C. Fredericks
[EOB]
*
```

example 4:

Moves to the buffer named LIST but prints nothing. The TYPE command simply verifies the new location.

```
*FIND =LIST
*TYPE .

    1       Distribution List for File Copies
```

related commands

- Keypad—FIND
- Nokeypad—"string"

HELP Command—Line

SYNTAX

HELP *[topic [subtopic [subsubtopic]]]*

DESCRIPTION

The HELP command puts you in touch with EDT's HELP facility. When you type HELP by itself, EDT prints general information and instructions on how to get help for a specific topic. To get information on subtopics, you must type the name of the topic and then the name of the subtopic, for example, HELP EXIT /SAVE. Using the HELP facility has no effect on your editing session.

You can access the HELP file any time you see the line mode asterisk prompt (*). You can also access the information directly from keypad mode by using the **COMMAND** function or from nokeypad mode by using the **EXT** (extend) command.

HELP for nokeypad commands is available through the line mode HELP command. Type HELP CHANGE for information on nokeypad syntax and the list of subtopics. To get information on nokeypad commands, type HELP CHANGE SUBCOMMANDS.

If you have access to more than one HELP file, use the **SET HELP** command to change HELP files. The **SHOW HELP** command prints the name of the HELP file that is currently available for your editing session.

EXAMPLES

1 `*HELP JOURNAL`

Prints information on EDT's journal facility.

2 `*HELP CHANGE HARDCOPY`

Prints information on hardcopy change mode.

3 `*HELP DEFINE KEY VT100`

Prints the keypad key number chart for VT100 terminals. These key numbers are used with the DEFINE KEY and SHOW KEY commands.

4 `*HELP RESEQUENCE /SEQUENCE`

Prints information on the /SEQUENCE qualifier when used with the RESEQUENCE command.

5 `*HELP CHANGE SUBCOMMANDS`

Prints general information on nokeypad commands.

6 `*HELP CHANGE SUBCOMMANDS TADJ`

Prints information on the nokeypad command TADJ.

related commands

— Keypad—HELP

INCLUDE Command—Line

SYNTAX

INCLUDE *file-specification [=buffer] [range]*

DESCRIPTION

The INCLUDE command copies external files into an EDT text buffer. In line mode, EDT displays its asterisk prompt (*) when the INCLUDE command has finished copying the file. If you use the INCLUDE command from keypad or nokeypad mode, the included text appears on the screen.

The file specification is the name of the file you want to include in the specified text buffer. If the file name is omitted, EDT assumes the same name as the input file. Similarly, if the file type is omitted, EDT assumes the same file type as the input file. However, you must include some element of the file specification in the command line.

If no buffer is specified, the file is added to the current buffer. The range specifier refers to a single line. EDT puts the new text above the range line. If you omit the range specifier, EDT puts the text above the current line or at the top of the specified buffer. EDT assigns appropriate numbers to the new lines that are added to the buffer. If there are any sequence numbers stored with the included file, EDT ignores them.

EXAMPLES

1
```
*INCLUDE LIST.DAT
```
EDT places a copy of the file LIST.DAT in the current buffer immediately above the current line.

2
```
*INCLUDE ADDR.DAT =ADDRESS
```
EDT places a copy of the file ADDR.DAT at the top of the buffer ADDRESS. If no buffer named ADDRESS exists, EDT creates one.

3
```
*INCLUDE [JACKSON]MEMO35.RNO 12
```
EDT places a copy of the file MEMO35.RNO from the directory JACKSON in the current buffer immediately above line 12.

4
```
*INCLUDE EXTRAPAR.X =MAIN 58
```
EDT places a copy of the file EXTRAPAR.X in the MAIN buffer immediately above line 58.

related commands

None.

INSERT Command—Line

SYNTAX

INSERT *[=buffer] [range] ;line-to-be-inserted*

INSERT *[=buffer] [range]* RETURN *text* CTRL/Z

DESCRIPTION

The INSERT command is used to add text to the current buffer or to another buffer, if one is specified. Text is always inserted above the line specified by range, above the current line if you omit the range specifier, or at the top of the specified buffer if only a buffer name is supplied.

To insert a single line of text, you can type INSERT, followed optionally by a buffer and/or range specifier, then a semicolon (;), and finally the text to be inserted. You must press RETURN to process the one-line insert. If the semicolon is omitted, EDT prints an error message. To see the line you have just inserted you can use the **TYPE** command with –1 for the range specifier.

To insert one or more lines, type INSERT, followed optionally by a buffer and /or range specifier, and press RETURN. EDT is now in the insert state. You can type as many lines as you want. EDT assigns ascending line numbers to the new lines. When you have finished typing the text, press **CTRL/Z** to exit from the insert state.

If you use a buffer specifier with your INSERT command, EDT moves to the named buffer and remains there after you have finished inserting the text. You must use the **FIND** command to return to the buffer where you were previously editing.

Only the **INSERT ;line-to-be-entered** form of the command can be used in startup command files or EDT macros. To insert several lines of text in a startup command file or EDT macro, put **INSERT[SP];** before each text line.

EXAMPLES

example 1:

Adds the telephone number line after the last line of the address. Uses PE to show the newly inserted line.

```
    2       South Bend, IN  46628
*INSERT 3 ;(219) 555-1234
    3
*TYPE -1
    2.1     (219) 555-1234
```

example 2:

Moves to the buffer ADDRESS and inserts an address. Then uses FIND to return to the MAIN buffer.

```
*INSERT =ADDRESS RETURN
            Dr. James Grogan
            Charlotte Professional Building
            Suite 12A
            Charlotte, NC  28202

CTRL/Z

[EOB]
*FIND =MAIN .
```

related commands

— Nokeypad—I

MOVE Command—Line

SYNTAX

MOVE *[=buffer-1] [range-1]* **TO** *[=buffer-2] [range-2] [/**QUERY**]*

DESCRIPTION

The MOVE command moves the line(s) specified by location-1 (buffer-1 and/or range-1) to location-2 (buffer-2 and/or range-2). The lines specified in location-1 are deleted from their original position and inserted at the new location. Use the **COPY** command if you want to have the text appear in both places.

If location-1 is omitted, EDT moves the current line. When location-2 is omitted, EDT inserts the text above the current line. Whenever you omit the buffer specifier, EDT assumes the current buffer. If buffer-1 is specified without a range-1 specifier, the entire contents of buffer-1 are moved to location-2. If buffer-2 is specified without a range-2 specifier, the text is inserted at the beginning of buffer-2.

Range-1 can refer to one or more lines in the buffer, but range-2 is limited to a single line in the current or specified buffer. Remember to leave spaces between the buffer name and any range specifier if one is used. Otherwise, EDT might misinterpret the buffer name.

If you do not include a buffer-2 specifier, then EDT completes the MOVE command in the same buffer from which you issued the command. When you do specify buffer-2, that buffer becomes the current buffer.

The **/QUERY** qualifier allows you to make decisions during the MOVE operation. EDT prints out each line in range-1 and prompts you with a question mark (?) so that you can decide whether you want to move that line. The four valid responses are: Y (YES), N (NO), A (ALL), and Q (QUIT).

EXAMPLES

1
```
*MOVE 5 TO 15
```
Moves line 5 in the current buffer to be above line 15, also in the current buffer.

2
```
*MOVE =PAGE7 TO 198
```
Moves the entire contents of the buffer PAGE7 to be above line 198 in the current buffer.

3
```
*MOVE =ADDRESS 1 THRU 5 TO
```
Moves lines 1 through 5 in the buffer ADDRESS to be above the current line in the current buffer.

4
```
*MOVE 38 THRU 94 TO =SECTION1
```
Moves lines 38 through 94 in the current buffer to be at the beginning of the buffer SECTION1. If no buffer named SECTION1 exists, EDT creates it.

```
5    *MOVE TO =MAIN 26
```

Moves the current line in the current buffer to be above line 26 in the MAIN buffer.

related commands

- Keypad—CUT [+ "move"] + PASTE
- Nokeypad—CUT [+ "move"] + PASTE

NEXT Command—Line

SYNTAX **[SUBSTITUTE] NEXT***[/[string-1]/string-2/]*

DESCRIPTION See SUBSTITUTE NEXT.

/NOTYPE Qualifier—Line

SYNTAX /NOTYPE

DESCRIPTION

The /NOTYPE qualifier, which is used only with the line mode **SUBSTITUTE** command, instructs EDT not to display the lines in which substitutions have been made. EDT displays only the message giving the number of substitutions. Always precede an EDT qualifier with a slash.

EXAMPLES

example 1:

Instructs EDT not to display the lines in which substitutions occurred.

```
*SUBSTITUTE/chairman/Chairperson/ WHOLE /NOTYPE
14 substitutions
```

example 2:

Instructs EDT not to display the lines in which substitutions occurred.

```
*SUBSTITUTE*input/output*I/O* . THRU +20 /NOTYPE
7 substitutions
```

< null> Command—Line

SYNTAX

[=buffer] [[%]range]

DESCRIPTION

The <null> command causes EDT to display the specified text. This command consists only of optional specifiers; there is no command word to type. If you omit both the buffer and range specifiers, the command simply consists of pressing the RETURN key in response to the asterisk prompt. In that case, EDT displays the next line in the current buffer.

The effect of the <null> command is the same as that of the **TYPE** command. For this reason the <null> command is referred to as the "implied TYPE" command.

When you include a buffer specifier, EDT moves to that buffer. If you do not specify a range, EDT displays every line in the buffer. Otherwise, EDT displays only the lines specified by range. When you move from one buffer to another, you can return to the same position in a previous buffer by using the period specifier (.) after the buffer name. For example, the command **=MAIN[SP].** moves EDT back to the MAIN buffer on line 37.21 if 37.21 was the most recent current line in MAIN.

After EDT displays the specified lines, it positions itself at the first line of the range or buffer, not at the end.

When you specify certain ranges, there are some differences between the TYPE command and the <null> command. Range specifiers that begin with letters—ALL, BEGIN, END, LAST, and REST—must be preceded by a percent sign (%) when they are the first element of a <null> command. If the percent sign is omitted, EDT tries to interpret the word as a command or macro name and returns an error message.

The <null> command takes no qualifiers. You can use the **/BRIEF** and **/STAY** qualifiers with the TYPE command.

EXAMPLES

example 1:

Causes EDT to display lines 2 through 4.

```
    1       This is the first line in the MAIN buffer.
    2       This is the second line in the MAIN buffer.
    3       This is the third line in the MAIN buffer.
    4       This is the fourth line in the MAIN buffer.
    5       This is the fifth line in the MAIN buffer.
    6       This is the sixth line in the MAIN buffer.

*2 THRU 4

    2       This is the second line in the MAIN buffer.
    3       This is the third line in the MAIN buffer.
    4       This is the fourth line in the MAIN buffer.
```

example 2:

Causes EDT to display the line below the current line.

```
*+1

    3       This is the third line in the MAIN buffer.
```

example 3:

Causes EDT to display the first line of the buffer.

```
*%BEGIN

    1       This is the first line in the MAIN buffer.
```

example 4:

Causes EDT to display the third line after the current line.

```
*+3

    4       This is the fourth line in the MAIN buffer.
```

example 5:

Causes EDT to display the next line after the current line when no specifier is supplied.

```
* RETURN

    5       This is the fifth line in the MAIN buffer.
```

example 6:

Causes EDT to display the line two lines above the current line.

```
*-2

    3       This is the third line in the MAIN buffer.
```

example 7:

Causes EDT to display all remaining lines in the buffer starting with the current line.

```
*%REST

    3       This is the third line in the MAIN buffer.
    4       This is the fourth line in the MAIN buffer.
    5       This is the fifth line in the MAIN buffer.
    6       This is the sixth line in the MAIN buffer.
[EOB]
```

example 8:

Moves to the PASTE buffer and causes EDT to display the entire contents of that buffer.

```
*=PASTE

    1       This is the first line in the PASTE buffer.
    2       This is the last line in the PASTE buffer.
[EOB]
```

example 9:

Moves to the buffer MEMO and displays lines 17 through 42.

```
*=MEMO 17 THRU 42
```

```
17      after the product has been thoroughly tested.
18
.
.
.
41      Please notify me of any changes in your schedule
42      as soon as you become aware of them.
```

related commands

None.

PRINT Command—Line

SYNTAX

PRINT *file-specification [=buffer] [range]*

DESCRIPTION

The PRINT command puts a copy of the specified range or buffer in an external file in the current or specified directory. EDT adds a form feed and two blank lines for every 60 lines it copies to the external file. Also, the EDT line numbers become part of the text in the external file. EDT does not resequence the lines when it processes the PRINT command. The line numbers remain exactly as they were before you issued the PRINT command.

When you specify neither range nor buffer, the entire current buffer is copied. If you give only a buffer name with no range, EDT copies the entire contents of the specified buffer, but does *not* move to that buffer. Instead, EDT remains in the buffer from which you issued the PRINT command.

Nonprinting characters are shown in the external file as EDT displays them, for example, <ESC> for the escape character (decimal value 27) and ^@ for the null character (decimal value 0).

If you include a directory name in the file specification, that directory must exist and you must have access to it. When you use the PRINT command, EDT creates a file, but EDT cannot create a directory.

The line numbers that appear in a file created by the PRINT command are different from those created by the **WRITE** command with the /SEQUENCE qualifier.

EXAMPLES

1
```
*PRINT MEMO1.RNO
```
Copies the entire current buffer to the file MEMO1.RNO.

2
```
*PRINT [JONES.STATUS]WEEK43 12 THRU END
```
Copies the current buffer, starting with line 12, to the file WEEK43 in the subdirectory JONES.STATUS.

3
```
*PRINT ADDRESS.DAT =HEADING
```
Copies the contents of the buffer HEADING to the file ADDRESS.DAT.

4
```
*PRINT LETTER1.RNO =BODY 7 THRU 56
```
Copies lines 7 through 56 from the buffer BODY to the file LETTER1.RNO.

related commands

None.

/QUERY Qualifier—Line

SYNTAX

/QUERY

DESCRIPTION

The /QUERY qualifier gives you control over the processing of four commands: **COPY**, **DELETE**, **MOVE**, and **SUBSTITUTE**. Use the /QUERY qualifier when you are performing one of the four operations on a range of lines, but do not want every line in the range to be affected by the command. Always precede an EDT qualifier with a slash.

When you include the /QUERY qualifier in a command line, EDT prompts you for each line in the range before any action is taken. EDT prints the line and then displays the question mark prompt (?) so that you can decide whether to have EDT perform the command for that line, skip the operation for that line, process the command for all remaining lines in the range, or stop processing the command at that line.

The four responses to the question mark prompt are:

Y	(YES)	Perform the command on this line.
N	(NO)	Do not perform the command on this line.
A	(ALL)	Perform the command on all remaining lines in the range.
Q	(QUIT)	Stop performing the command at this line. The remaining lines in the range will not be affected.

When you use the /QUERY qualifier with the SUBSTITUTE command, it might seem that there are four delimiters on the command line.

```
SUBSTITUTE/1984/1985/ WHOLE /QUERY
```

The slash before the word **QUERY** is the qualifier signal, not a delimiter. If you use a different delimiter for the substitute strings, you can see the distinction:

```
SUBSTITUTE*1984*1985* WHOLE /QUERY
```

EXAMPLE

Prompts you to find out which lines in the range 5 through 9 you want to delete. The TYPE command shows which lines in the range 4 through 10 remain in the buffer.

```
     4       Jacksonville, FL
     5       Huntsville, AL
     6       Thomasville, GA
     7       Potsdam, NY
     8       Grand Forks, ND
     9       Missoula, MT
    10       Eugene, OR
```

```
*DELETE 5 THRU 9 /QUERY
    5       Huntsville, AL
?Y
    6       Thomasville, GA
?N
    7       Potsdam, NY
?Y
    8       Grand Forks, ND
?A
4 lines deleted
   10       Eugene, OR
*TYPE 4 THRU 10

    4       Jacksonville, FL
    6       Thomasville, GA
   10       Eugene, OR
```

QUIT Command—Line

SYNTAX

QUIT [/SAVE]

DESCRIPTION

The QUIT command ends an EDT session without copying any text to an external file. Only a copy of the original file as it was before you started your EDT session exists. There is no copy of the edited text. If you use EDT to create a new file and you type QUIT to end your editing session, no copy of the text will exist in any directory. Use the **EXIT** command to preserve a copy of the MAIN buffer text in an external file.

If you use the **/SAVE** qualifier with QUIT, EDT saves the journal file from your editing session. You might want to do this if your text has become confused or if you have accidentally deleted some text that you want to recover. First, use QUIT/SAVE to end your editing session. Then using EDT, delete the last few commands from the journal file. To resume your work, use the /RECOVER qualifier with your EDIT/EDT command to restore your editing work up to the point where the problem occurred.

Pressing CTRL/Z three times in a row has the same effect as issuing the QUIT/SAVE command.

EXAMPLES

example 1:

Ends your EDT session without copying the MAIN buffer contents to an external file. Only the system prompt appears.

```
*QUIT
$
```

example 2:

Lists the files in the directory. Then starts an EDT session, but ends it with QUIT/SAVE to retain the journal file. A second system DIRECTORY command shows the journal file (CHAPTER1.JOU) in the directory.

```
$ DIRECTORY
    CHAPTER1.RNO;2  CHAPTER1.RNO;1
$ EDIT /EDT CHAPTER1.RNO
   .
   .
   .
*QUIT/SAVE
$ DIRECTORY
    CHAPTER1.JOU;1  CHAPTER1.RNO;2  CHAPTER1.RNO;1
```

related commands

— Nokeypad—QUIT

range specifier—Line

SYNTAX

range-indicator

DESCRIPTION

The range specifier is used with these line mode commands:

CHANGE	<null>
COPY	PRINT
DELETE	REPLACE
FIND	RESEQUENCE
INCLUDE	SUBSTITUTE
INSERT	TYPE
MOVE	WRITE

The range specifier can reference one line or a group of lines depending on your needs or the command with which you use the range.

There are a number of ways to indicate a range. Some of the more common ones are listed here. When you use the string specifier with most line mode commands, EDT considers that you are referencing the entire line. For example, if you issue the command **DELETE "string"** , EDT deletes the entire line that contains the quoted string.

EDT keeps track of the string location when you use a string specifier with a command like **FIND** or **TYPE**. By remembering the position of the string, EDT allows you to locate the next occurrence of that string.

Range	Description
. (period)	The current line.
number	The exact line number, for example, 95.31.
range THRU range range:range	The lines starting at the first range indicator and continuing through the second range indicator.
WHOLE	Every line in the buffer.
BEGIN	The top of the buffer.
END	The bottom of the buffer.
BEFORE	The first line in the current buffer through the line above the current line.
REST	The current line in the current buffer through the last line in the buffer.
SELECT	The active select range established by the keypad **SELECT** function or the nokeypad **SEL**, **SSEL**, or **TGSEL** command. The select range can include only whole lines.

Range	Description
"string" 'string'	A string of characters located on or after the current line.
–"string" –'string'	A string located before the current line
+n	The number of lines below the current line; for example, +3 is the third line below the current one.
–n	The number of lines above the current line; for example, –10 is the tenth line above the current one.

When a string is used for the range specifier, it must be surrounded by either single (') or double (") quotation marks. When the first element of the <null> command is a word range specifier such as BEGIN, END, BEFORE, REST, and WHOLE it must be preceded with a percent sign (%) to indicate to EDT that it is not a command or an EDT macro.

EXAMPLES

1 `*.`

Types the current line.

2 `*CHANGE "page"`

Shifts to a change mode and puts the cursor to the right of the next occurrence of the word **page**.

3 `*DELETE 25 THRU END`

Deletes lines, starting with 25, through the end of the buffer.

4 `*FIND =PASTE 23`

Moves to line 23 in the PASTE buffer.

5 `*%REST`

Displays the remaining lines in the current buffer, starting with the current line.

6 `*WRITE OUTFILE.DAT . THRU +10`

Creates a file called OUTFILE.DAT and copies 11 lines into it, starting with the current line and ending with the tenth line after the current line.

7 `*RESEQUENCE 25 -5 THRU 30 /SEQUENCE:100`

Resequences the lines starting with five lines before line number 25 and continuing through line number 30 or as far as is necessary to maintain ascending line numbers. Assigns the number 100 to the first line in the range. Then increments the line numbers by 1.

8 `*WRITE LETTER1A.RNO "In response" THRU "cc: David North"`

Creates a file called LETTER1A.RNO and puts into it a copy of the lines starting with the one containing the string **In response** through the one with the string **cc: David North**.

REPLACE Command—Line

SYNTAX

REPLACE *[=buffer] [range] [;line to be inserted]*

REPLACE *[=buffer] [range]* RETURN *text* CTRL/Z

DESCRIPTION

The REPLACE command deletes the line(s) specified by range from the current or specified buffer. EDT then replaces the deleted line(s) with text that you enter at the terminal. There are two ways to enter new text: 1) on the same line as the REPLACE command, or 2) through the insert state.

If the text to be entered does not contain any line terminators, it can be typed on the same line as the REPLACE command. Separate the text from the command with a semicolon (;). When you press RETURN, the line or lines specified by range are deleted and replaced by the text following the semicolon.

To replace the deleted text with one or more lines, press RETURN after typing the REPLACE command. EDT shifts to the insert state as soon as it deletes the specified lines. Use **CTRL/Z** to exit from the insert state when you have finished entering the new text.

Only the single line form of the REPLACE command can be used in startup command files or EDT macros.

EXAMPLES

example 1:

Using the single line form of the REPLACE command, deletes line 3 and replaces it with new text. The TYPE command verifies that the new line has been entered.

```
    1       Mr. Theodore R. Swenson
    2       34 North Main Street
    3       Londonderry, NH  03053
    4       (603) 555-1234

*REPLACE "London" ;Derry, NH  03038
1 line deleted
    4       (603) 555-1234
*TYPE -1
    3       Derry, NH  03038
```

example 2:

Deletes line 7 and inserts new text. The TYPE command verifies the new text.

```
    6       Robert Harrison
    7       John Barber

*REPLACE 7 RETURN
1 line deleted
```

```
            Marjorie Dickerson
            James Newbold
            Dorothy Urquart
            Faith Jaspersen

CTRL/Z

    8       William Wetherall

*TYPE 6 THRU 8

    6       Robert Harrison
    6.1     Marjorie Dickerson
    6.2     James Newbold
    6.3     Dorothy Urquart
    6.4     Faith Jaspersen
    8       William Wetherall
```

example 3:

Using the single line form of the REPLACE command, moves to the buffer HEADING and replaces the first line with new text. Verifies the new text with the TYPE command.

(These are the contents of the buffer HEADING.)

```
    1       DATE:  July 8, 1985
    2       TO:    Sam Hill
    3       FROM:  John Archer
```

(The REPLACE command is typed from the MAIN buffer.)

```
*REPLACE =HEADING 1;DATE:  August 20, 1985
1 line deleted
    2       TO:    Sam Hill
*TYPE 1
    1       DATE:  August 20, 1985
```

related commands

- Keypad—REPLACE
- Nokeypad—R (replace)

RESEQUENCE Command—Line

SYNTAX

RESEQUENCE *[=buffer] [range]*
*[/***SEQUENCE***[:initial[:increment]]]]*

DESCRIPTION

The RESEQUENCE command assigns new EDT line numbers to the lines of the current buffer or the specified buffer. If you use the buffer specifier without a range specifier, EDT renumbers the entire buffer. If a range is given, the lines within that range must be contiguous.

When RESEQUENCE is used without the **/SEQUENCE** qualifier, EDT uses the value 1 for both of the /SEQUENCE specifiers—:initial and :increment. You get the same results if you use RESEQUENCE /SEQUENCE with no :initial and :increment specifiers. You can omit the /SEQUENCE qualifier and its specifiers only when you are renumbering an entire buffer or when the line specified by **range** has a number less than or equal to 1.

After EDT renumbers your text, it prints the number of lines that have been resequenced. This number reflects the actual number of lines that EDT had to resequence in order to avoid duplicating or overlapping line numbers.

EXAMPLES

1
```
*RESEQUENCE
*RESEQUENCE /SEQUENCE
*RESEQUENCE /SEQUENCE:1:1
```
All three commands resequence the entire current buffer using default increment and initial values of :1:1.

2
```
*RESEQUENCE =BUFF3
```
Resequences all of buffer BUFF3 using the default values :1:1.

3
```
*RESEQUENCE /SEQUENCE:100:10
```
Resequences the entire current buffer, starting with an initial line number of 100 and using increments of 10.

4
```
*RESEQUENCE =LIST 5 THRU END /SEQUENCE:10:10
```
Moves to the buffer LIST and resequences it, starting at line 5. Assigns the initial value of 10 to line 5 and then increments the successive line numbers by 10.

related commands

None.

/SAVE Qualifier—Line

SYNTAX /SAVE

DESCRIPTION

The /SAVE qualifier instructs EDT to save a copy of the journal file in the current directory. The journal file contains a record of all the keystrokes you have made during the current EDT session.

/SAVE is used with the line mode commands **EXIT** and **QUIT**. When a normal exit is made from EDT, whether by means of EXIT or QUIT, EDT discards the journal file. The /SAVE qualifier causes the journal file to be saved in your current directory. The journal file is automatically saved whenever your editing session ends abnormally.

All EDT qualifiers must be preceded by slashes.

EXAMPLES

1
```
*EXIT /SAVE
```
Exits from EDT, saving a copy of the MAIN buffer as well as the journal file.

2
```
$ EDIT /EDT LIST.DAT
   .
   .
   .
*EXIT NEWLIST.DAT /SAVE
```
Exits from EDT, saving a copy of the MAIN buffer in a file called **NEWLIST.DAT**. The journal file is called **LIST.JOU**.

3
```
*QUIT /SAVE
```
Exits from EDT without saving a copy of the MAIN buffer. Only the journal file is saved.

/SEQUENCE Qualifier—Line

SYNTAX

/SEQUENCE*[:initial[:increment]]*

DESCRIPTION

The /SEQUENCE qualifier is used with the line mode commands **EXIT**, **RESEQUENCE**, and **WRITE**. Using the /SEQUENCE qualifier enables you to renumber the EDT line numbers for a block of lines. With the EXIT and WRITE commands, the /SEQUENCE qualifier causes sequence numbers to be written to the external file created by those commands.

With the RESEQUENCE command, the :initial specifier is the line number you want assigned to the first line of the specified buffer or range. The default value for :initial is 1. The :increment specifier refers to the spacing you want between the numbers in the new sequence. The default value for :increment is also 1. You must include the :initial specifier if you want to use :increment.

When you use the /SEQUENCE qualifier with EXIT or WRITE, the default values for :initial and :increment are different. The default values are the current EDT line numbers for the block of lines being copied to the external file. However, line numbers with decimal fractions cannot be stored in a sequenced file, so EDT must adjust any decimal line numbers to be whole numbers when you use the /SEQUENCE qualifier with no specifiers.

All EDT qualifiers must be preceded by slashes.

EXAMPLES

1
```
*RESEQUENCE
*RESEQUENCE /SEQUENCE
*RESEQUENCE /SEQUENCE:1:1
```
All of these RESEQUENCE commands renumber the entire current buffer starting with 1 and incrementing by 1.

2
```
*EXIT /SEQUENCE:100:10
```
Exits from EDT. The output file has sequence numbers that start at 100 and increase by 10.

3
```
*WRITE LIST.DAT 25 THRU END /SEQUENCE:10:5
```
Writes a copy of lines 25 through the end of your current buffer to an external file called LIST.DAT. The file has sequence numbers starting at 10 for line 25 in the current buffer and incrementing by 5.

SET [NO]AUTOREPEAT Command—Line

SYNTAX

SET AUTOREPEAT

SET NOAUTOREPEAT

DESCRIPTION

The SET AUTOREPEAT command enables EDT to use the DECARM VT100 control sequence to prevent keypad keys (including arrow keys) from repeating faster than EDT can update the screen. SET AUTOREPEAT is the default.

On some VT100-type terminals, SET AUTOREPEAT causes the arrow keys to repeat very slowly (approximately .5 seconds for each repeated use). With SET NOAUTOREPEAT, the arrow keys repeat faster because EDT does not use the DECARM control sequence. If EDT gets behind, it simply skips intermediate screen updates.

Occasionally, on the VT100 terminal with printer port (but not on the VT102, which has an integrated printer port), use of SET NOAUTOREPEAT causes the terminal to stop transmission. If this occurs, press the SET-UP key (located at the upper lefthand corner of the keyboard) twice to clear the keyboard buffer.

SET [NO]AUTOREPEAT has no effect on the keypad mode GOLD/repeat feature or on the SPECINS function. These operations are affected by **SET [NO]REPEAT**.

EXAMPLE

Changes the default SET AUTOREPEAT setting to SET NOAUTOREPEAT.

```
*SET NOAUTOREPEAT
```

related commands

None.

SET CASE Command—Line

SYNTAX

SET CASE NONE

SET CASE LOWER

SET CASE UPPER

DESCRIPTION

The SET CASE command is useful when you are reading text which has both upper- and lowercase letters at a single-case terminal. The SET CASE command uses flags to distinguish lowercase letters from uppercase ones. If you use SET CASE UPPER, each uppercase letter in the original file is preceded by an apostrophe ('). When you specify SET CASE LOWER, each lowercase letter in the original file is marked by an apostrophe. The flags appear only when the text is displayed at a terminal under line mode. No permanent marks are placed in the text itself.

The default state for EDT is SET CASE NONE, meaning that no provisions are made for artificially distinguishing between upper- and lowercase letters.

EXAMPLE

Assumes that you are working at a single-case terminal. Issues the SET CASE UPPER command to flag all letters that are actually uppercase in the original text.

```
USE THE SET CASE UPPER COMMAND WITH A SINGLE-CASE TERMINAL.

*SET CASE UPPER

'USE THE 'S'E'T 'C'A'S'E 'U'P'P'E'R COMMAND WITH A SINGLE-CASE TERMINAL.
```

related commands

None.

SET COMMAND Command—Line

SYNTAX

SET COMMAND *file-specification*

DESCRIPTION

The SET COMMAND command allows you to process additional startup command files at the beginning of your EDT session. The word COMMAND refers to the startup command file, as does the /COMMAND qualifier used in EDIT/EDT command lines. You can use the SET COMMAND command only in a startup command file.

The default startup command file that EDT looks for is a system-wide file located in the system library. The file specification is:

SYNTAX

SYS$LIBRARY:EDTSYS.EDT

When you issue the system EDIT/EDT command, EDT first checks the command line to see whether you have included a command file specification. If EDT finds one, it processes that file. Otherwise, EDT looks for a system-wide startup-command file to process. If no system-wide command file exists, EDT looks for a file named EDTINI.EDT in your default directory and processes that file. If you use the /NOCOMMAND qualifier in your EDIT/EDT command line, or if EDT fails to find a startup command file, your session begins with the default settings for all parameters.

You must include at least the file name in the SET COMMAND command line. **.EDT** is the default file type. If the specified file does not exist, EDT ignores the SET COMMAND command and continues to process any remaining commands in the current startup command file. Thus, you can have several SET COMMAND commands in the system startup command file, each branching to a different group or personal startup command file.

To bypass the default startup command file, either specify a different startup command file in the EDIT/EDT command line or use the /NOCOMMAND qualifier in the EDIT/EDT command line.

If you include the **SHOW COMMAND** command in a startup command file, the name of that file is displayed at your terminal when EDT reads the file.

EXAMPLE

Shows the SET COMMAND entry in a system-wide startup command file, branching to the personal command file EDTEXPERT.EDT. Another SET COMMAND command at the end of the file transfers EDT to the EDTINI.EDT command file for users who do not have an EDTEXPERT.EDT command file in their default directories.

```
DEFINE KEY GOLD S AS "S^@?'Find: '^@?' Substitute: '^@."
SET WRAP 65
   .
   .
   .
SET COMMAND EDTEXPERT.EDT
DEFINE KEY GOLD L AS "SHL."
DEFINE KEY GOLD R AS "SHR."
   .
   .
   .
SET COMMAND EDTINI.EDT
```

related commands

None.

SET CURSOR Command—Line

SYNTAX

SET CURSOR *top:bottom*

DESCRIPTION

The SET CURSOR command controls the scrolling of the screen relative to the cursor position. SET CURSOR has no effect if you are using a screen terminal in line mode.

The **top** specifier refers to the number of lines from the top of the screen to the cursor. As you move the cursor up toward the beginning of the buffer, EDT begins to scroll the screen display. Lines are added at the top of the screen when the cursor is the specified number of lines from the top. The **bottom** specifier refers to the number of lines from the cursor to the bottom of the screen. As you move the cursor down toward the end of the buffer, EDT begins to scroll the screen display. Lines are added at the end of the screen when the cursor is the specified number of lines from the bottom.

The default cursor settings are 7:14. When you move the cursor down to the 15th line on the screen, the first line on the screen disappears and the next line after the current bottom line appears at the bottom of the screen. Conversely, as you move the cursor up the screen, when it reaches the seventh line from the top, EDT begins scrolling down. Of course, scrolling stops when you are near the beginning or end of the buffer.

When using the SET CURSOR command, **top** must be a smaller number than **bottom**, and both must be less than or equal to the total number of screen lines.

The maximum number of lines on a screen is 22, designated 0 through 21 for the SET CURSOR command. The minimum value for the top specifier is 0 and the maximum number for the bottom specifier is 21.

You can use the **SET LINES** command to reduce the maximum number of lines visible on your screen.

EXAMPLES

1
```
*SET CURSOR 1:20
```

The cursor remains either one line from the top of the screen or from the bottom of the screen, depending on the scrolling direction.

2
```
*SET CURSOR 9:11
```

The cursor remains one line on either side of the middle of the screen, depending on the scrolling direction.

related commands

None.

SET ENTITY Command—Line

SYNTAX

SET ENTITY WORD *"string"*

SET ENTITY SENTENCE *"string"*

SET ENTITY PARAGRAPH *"string"*

SET ENTITY PAGE *"string"*

DESCRIPTION

The SET ENTITY command defines the delimiters that mark certain entity boundaries for EDT commands and functions. You can use the SET ENTITY command to redefine the boundaries for four screen mode entities: WORD, SENTENCE, PARAGRAPH, AND PAGE. Other entities, such as character and line, cannot be redefined.

The default delimiters for the four EDT entities that can be reset are:

```
WORD                <LF><VT><FF><CR>
SENTENCE    .?!
PARAGRAPH   <CR><CR>
PAGE        <FF>
```

A WORD is defined as any group of characters bounded by a space, horizontal tab (displayed by the EDT as if it were a group of spaces), a line feed, a vertical tab, a form feed, or a carriage return. A SENTENCE is defined as any group of characters bounded by a period, a question mark, or an exclamation point, each of which must be followed by at least one space or line terminator. A PARAGRAPH is defined as a group of characters bounded on either side by an empty line (two line terminators in succession—displayed by EDT as <CR> <CR>). A PAGE is bounded by a form feed, top of buffer, or bottom of buffer.

In resetting delimiters, remember that the SET ENTITY command overwrites the default values. If you want to add delimiters to the existing ones, you must include all the original delimiters in your command string, not just the new ones you want to add. The delimiters you specify must be enclosed in quotation marks, single (') or double ("). If you want to include both kinds of quotation marks in your list of delimiters, double the kind that is used to enclose the list. The command *SET ENTITY SENTENCE """.?""'!"*\BOLD) has all the default sentence boundary characters plus both kinds of quotation marks.

The elements in the WORD and SENTENCE boundary list are single characters, each of which is a boundary by itself. You can have any number of different WORD and SENTENCE boundaries at the same time during your EDT session. Do not use spaces or any other characters to separate the individual boundary marks in the SET ENTITY WORD or SENTENCE command.

For PARAGRAPH and PAGE, you can only have a single boundary at any given time, but that boundary can be composed of several characters. For example, your PAGE boundary can be the word **PAGE**. If you use letters in your PARAGRAPH or PAGE boundary, EDT uses its current search parameters when locating the boundary string. Thus, if SET SEARCH EXACT is in effect and the PAGE boundary is **.PAGE**, EDT will not recognize **.page** as a page separator.

For SET ENTITY WORD, EDT considers all word boundaries, except spaces, to be word entities themselves unless **SET WORD NODELIMITER** is in effect.

EXAMPLES

1 `*SET ENTITY WORD '  {}[]()<LF><VT><FF><CR>'`

Sets the word entity to include all the default boundaries as well as braces, brackets, and parentheses.

2 `*SET ENTITY SENTENCE ".!?'""]")`

Sets the sentence entity to be any string of characters ending with ., !, ?, ", ',), or] followed by at least one space or line terminator. Notice that the double quotation mark (") inside the boundary list is doubled.

3 `*SET ENTITY PARAGRAPH "<CR><CR><CR>"`

Sets the paragraph entity to be any group of lines separated by two empty lines.

4 `*SET ENTITY PAGE '.HL1'`

Sets the page entity to be any text beginning with a RUNOFF level-one header.

related commands

None.

SET [NO]FNF Command—Line

SYNTAX

SET FNF

SET NOFNF

DESCRIPTION

The SET NOFNF command suppresses the message that appears when you use EDT to create a new file. (FNF stands for File Not Found.) Normally, when you call up EDT to create a file, you see this message:

```
Input file does not exist
```

This is the default state—SET FNF.

When you have the SET NOFNF command in an EDT startup command file, EDT does not display the **Input file does not exist** message at the start of your editing session. Note that SET NOFNF does not suppress the message **Input file does not have standard text file format**.

EXAMPLE

Shows the start of an EDT session in which the "Input" message is displayed and then shows a session in which the message has been suppressed by placing the SET NOFNF command in the startup command file.

```
$ EDIT /EDT JUL141989.RNO
Input file does not exist.
[EOB]
*
   .
   .
   .
*EXIT
```

(Now put the SET NOFNF command in your startup command file.)

```
$ EDIT /EDT JUL141989.RNO
[EOB]
*
```

related commands

None.

SET HELP Command—Line

SYNTAX

SET HELP *[file-specification]*

DESCRIPTION

The SET HELP command enables you to access different HELP files for your EDT session. Since EDT allows users to modify the HELP file text or create new HELP files, you might have access to more than one HELP file. The SET HELP command enables you to use the HELP file that corresponds to your key definitions and EDT macros, even if that file is not the default HELP text for your site.

If you have key definitions and EDT macros that are documented in a revised HELP file, include the appropriate SET HELP command in your startup command file so that your HELP text will be consistent with your editing environment.

When you issue the SET HELP command without a file specification or when you type SET HELP EDTHELP, EDT returns you to the default HELP file. The **SHOW HELP** command tells you the name of the current HELP file.

If you omit parts of the file specification in your SET HELP command, the following defaults will be supplied by the operating system:

Device Name	File Name	File Type
SYS$HELP:	EDTHELP	.HLB

EXAMPLES

1
```
*SET HELP EDTHELP
```
Specifies EDT's default HELP file.

2
```
*SET HELP DISK$USER:[SMITH]XEDTHELP.HLB
```
Specifies a different EDT HELP file.

related commands

None.

SET KEYPAD Command —Line
SET NOKEYPAD Command

SYNTAX

SET KEYPAD

SET NOKEYPAD

DESCRIPTION

The SET KEYPAD/NOKEYPAD command determines which screen mode EDT accesses. The default screen mode is keypad. When you begin an EDT session and give the **CHANGE** command, your editing session is automatically shifted to keypad mode. If you give the SET NOKEYPAD command and then the CHANGE command, you enter nokeypad mode.

If you have the **SET MODE** CHANGE command in your startup command file, EDT automatically starts your session in keypad mode. To have your EDT sessions start in nokeypad mode, you must include the SET NOKEYPAD command in the startup command file.

You can use the SET NOKEYPAD command with the keypad **COMMAND** function to shift directly to nokeypad editing from keypad mode. Conversely, when you are in nokeypad mode, you can issue the **EXT** SET KEYPAD command to shift directly to keypad editing.

Once you have issued a SET NOKEYPAD command, you can switch back and forth between line mode and nokeypad mode without having to reissue the SET NOKEYPAD command.

EXAMPLES

example 1:

Shifts from line mode to nokeypad mode at the start of an editing session.

```
$ EDIT /EDT LETTER.RNO
*SET NOKEYPAD
*CHANGE
```

example 2:

Shifts from line mode to keypad mode when SET NOKEYPAD is in effect.

```
*SET KEYPAD
*CHANGE
```

related commands

None.

SET LINES Command—Line

SYNTAX

SET LINES *number*

DESCRIPTION

The SET LINES command is used to limit the number of lines that EDT displays on the terminal screen at one time. The number specifier can be any integer from 1 to 22. The default value is the maximum: 22 lines per screen. This command has no effect on the number of lines moved by the keypad **SECT** function. SET LINES also has no effect in line mode.

If you reduce the number of lines on the screen, use the **SET CURSOR** command to adjust the cursor settings that tell EDT when to scroll the screen. If you set the number of lines small enough to invalidate the current SET CURSOR values, EDT adjusts the SET CURSOR values so that they will be valid. However, EDT does not change the SET CURSOR values when you increase the number of lines displayed per screen. Use the **SHOW CURSOR** command to see the current cursor settings.

The SET LINES command is useful if you are working at a terminal operating at a low data transmission rate. With fewer lines displayed on the screen, EDT requires less time to redraw the screen when you issue a command or keypad keystroke that significantly alters the screen image.

EXAMPLE

Limits the screen display to eight lines. Checks the new cursor setting and then uses the SET CURSOR command to reset the scrolling lines to 2 and 6.

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
This is line 6.
This is line 7.
This is line 8.
This is line 9.
This is line 10.
   .
   .
   .
This is line 22.
```

GOLD + PAGE COMMAND

```
Command:  SET LINES 8
```

ENTER SUBS

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
This is line 6.
This is line 7.
This is line 8.

*SHOW CURSOR
7:7
*SET CURSOR 2:6
```

related commands

None.

SET MODE Command—Line

SYNTAX

SET MODE CHANGE

SET MODE LINE

DESCRIPTION

The SET MODE command establishes the initial mode of your EDT session. When SET MODE CHANGE is in an EDT startup command file, your editing session automatically starts off in change mode instead of line mode, which is the default.

The default change mode is keypad mode if the operating system informs EDT that you have either a VT100-type or VT52 terminal. If you are using some other kind of terminal, the default change mode is hardcopy change mode.

The **SHOW MODE** command shows the mode that was last set by a SET MODE command. The mode that SHOW MODE displays will not necessarily correspond to the current EDT mode since you can use other commands to shift from one editing mode to another during your editing session.

EXAMPLE

Shows an EDT session starting out in the default mode—line. Then shows the beginning of the same session if SET MODE CHANGE is in the startup command file.

```
$ EDIT /EDT LETTER.RNO
    1        January 1, 1986
*
```

(After you put the SET MODE CHANGE command in the startup command file, reissue the EDIT /EDT command.)

```
$ EDIT /EDT LETTER.RNO

January 1, 1986

Mr. Jordan R. Kingsberry
    .
    .
    .
[EOB]
```

related commands

None.

SET [NO]NUMBERS Command—Line

SYNTAX

SET NUMBERS

SET NONUMBERS

DESCRIPTION

SET [NO]NUMBERS determines whether or not EDT displays its line numbers during line mode editing. If you request SET NONUMBERS, EDT displays text in line mode starting at the left margin of your screen or paper, without showing the EDT line numbers. To restore the default state, give the SET NUMBERS command. EDT maintains its line numbering system even when SET NONUMBERS is in effect.

EXAMPLE

The first TYPE command lists lines 3 through 5 with the EDT line numbers displayed. The second TYPE command lists the same lines when SET NONUMBERS is in effect.

```
*TYPE 3 THRU 5

    3       Ms. Phyllis Davenport
    4       3587 Charter Lane
    5       Hartford, CT  06107

*SET NONUMBERS
*TYPE 3 THRU 5

Ms. Phyllis Davenport
3587 Charter Lane
Hartford, CT  06107
```

related commands

None.

SET PARAGRAPH [NO]WPS Command—Line

SYNTAX

SET PARAGRAPH WPS

SET PARAGRAPH NOWPS

DESCRIPTION

The default boundary limits for a paragraph in EDT are two line terminators in succession. When you use the paragraph entity to move the cursor in a change mode with EDT in the forward direction, EDT positions the cursor on the character immediately following the second line terminator of the paragraph boundary. SET PARAGRAPH NOWPS is the default.

The SET PARAGRAPH WPS command recognizes an empty line as the paragraph boundary, but does not leave the cursor at the next character after the empty line if that character is another line terminator. With SET PARAGRAPH WPS in effect, EDT looks for the first character beyond the paragraph boundary that is not a line terminator, and moves the cursor there.

If EDT's direction is backward and SET PARAGRAPH WPS is in effect, EDT moves the cursor to the nearest character preceded by two line terminators in succession, providing that character is not a line terminator itself.

If you have changed the paragraph boundary with the SET ENTITY PARAGRAPH command and the new entity ends with a line terminator (<CR>), SET PARAGRAPH WPS will cause EDT to skip over any blank lines that might follow directly after the paragraph entity.

EXAMPLE

Shows the cursor position when the PAR entity is used with the default SET PARAGRAPH NOWPS. Then shows the cursor position when SET PARAGRAPH WPS is in effect. The EDT direction is forward.

```
at any time since the last war.

Later that day she was to regret her unkind remark.
True, it had been unintentional, but she could tell

PAR

at any time since the last war.

_
Later that day she was to regret her unkind remark.
True, it had been unintentional, but she could tell
```

(Move the cursor back to the **a** in **at**.)

```
EXT SET PARAGRAPH WPS
PAR
```

```
at any time since the last war.

Later that day she was to regret her unkind remark.
True, it had been unintentional, but she could tell
```

related commands

None.

SET PROMPT Command—Line

SYNTAX

SET PROMPT *prompt-type "string"*

DESCRIPTION

The SET PROMPT command is used during EDT development to facilitate automatic testing of EDT. It allows redefinition of the prompt string that is displayed in line mode, in keypad mode, in nokeypad mode, in hardcopy change mode, in line mode when inserting, in line mode when inserting with NONUMBERS, and in line mode with the **/QUERY** qualifier.

Prompt types cannot be abbreviated. They are:

```
LINE      KEYPAD    NOKEYPAD  HCCHANGE
INSERT    INSERTN   QUERY
```

EDT's automatic testing facility uses SET PROMPT to add a unique, nonprinting character (CTRL/A) to the end of each prompt. The character signals the tester that EDT is ready for simulated user input.

Use of this command is not recommended. Note that with SET PROMPT in effect, EDT will become confused if the new prompt string for the keypad or nokeypad prompt causes a net motion of the cursor. Also note that if a <CR> <LF> pair is not included in the new string for any prompt that has those characters in the default prompt, these characters will be inserted at the beginning of the new string. The insert prompt is modified if the line numbers near the point of insertion become large.

The default prompts are:

```
LINE:       <CR><LF>*          INSERT:    <CR><LF>xxxxxxxxxxxx
KEYPAD:                        INSERTN:   <CR><LF>
NOKEYPAD:                      QUERY:     <CR><LF>?
HCCHANGE:   <CR><LF>C*
```

EXAMPLES

1 `*SET PROMPT LINE "<CR><LF>!!!"`

Sets the line mode prompt to an end of line followed by three exclamation points.

2 `*SET PROMPT KEYPAD "^@"`

Sets the keypad mode prompt to a string containing one null character.

3 `*SET PROMPT NOKEYPAD ""`

Sets the nokeypad mode prompt to nothing.

4 `*SET PROMPT HCCHANGE "<CR><LF>CHANGE"`

Sets the hardcopy change mode prompt to an end of line followed by the word CHANGE.

5 `*SET PROMPT INSERT "<CR><LF>{"`

Sets the line mode insert prompt to an end of line followed by a left brace.

6 `*SET PROMPT INSERTN "<CR><LF>+"`

When SET NONUMBERS is in effect, sets the line mode insert prompt to an end of line followed by a plus sign.

7 `*SET PROMPT QUERY "<CR><LF>?????"`

Sets the line mode /QUERY prompt to an end of line followed by five question marks.

---

related commands

None.

SET [NO]QUIET Command—Line

SYNTAX

SET QUIET

SET NOQUIET

DESCRIPTION

The SET QUIET command silences the terminal bell that ordinarily sounds whenever EDT issues an error message during a screen mode editing session. Use SET NOQUIET, the default, to restore the bell sound.

SET [NO]QUIET has no effect on the nokeypad BELL command.

EXAMPLE

Shows the SET QUIET command in a startup command file.

```
SET SEARCH EXACT
SET QUIET
SET MODE CHANGE
```

related commands

None.

SET [NO]REPEAT Command—Line

SYNTAX

SET REPEAT

SET NOREPEAT

DESCRIPTION

The SET NOREPEAT command disallows use of the GOLD repeat feature, which enables you to repeat functions in keypad mode. The default is SET REPEAT, where pressing GOLD followed by a keyboard digit (or digits) causes the subsequent keypad function to be repeated as many times as the number entered. SET NOREPEAT also disallows the use of the **SPECINS** keypad function, which enables you to insert any character from the DEC Multinational Character Set into your text by using its decimal equivalent value.

If you are accustomed to a word processing system, you might find the NOREPEAT option more convenient to use.

EXAMPLE

Shows the SET NOREPEAT command in a startup command file.

```
SET PARAGRAPH WPS
SET NOREPEAT
SET MODE CHANGE
```

related commands

None.

SET SCREEN Command—Line

SYNTAX

SET SCREEN *width*

DESCRIPTION

The SET SCREEN command allows you to change the maximum number of characters displayed on each screen line during your editing session. The width specifier can be either 80 or 132 for VT100-type terminals (including VT102s) with advanced video option (AVO) or 80 for VT52s and VT100s without AVO (including VT101s). SET SCREEN also affects the number of characters displayed on your paper if you are using a hardcopy terminal. For hardcopy terminals, the width specifier can take any value between 1 and 132.

Be sure not to specify a value for width that exceeds your terminal's capacity or exceeds 132. Otherwise, EDT becomes confused and cannot process your edits properly.

To see if your VT100-type terminal has AVO, press the SET-UP key at the top left corner of the keyboard. The word **SET-UP** is displayed in large letters at the top of the screen. If SET-UP flashes, your terminal has the advanced video option and can accommodate a SET SCREEN value of 80 or 132. Press the SET-UP key again to restore the previous screen display.

The SET SCREEN command does not cause EDT to wrap long lines in the screen modes. Use the **SET NOTRUNCATE** command to show the ends of lines that exceed the screen width. Use **SET WRAP** in keypad mode to have EDT wrap lines of text as you insert them.

You must use the SET WRAP command to change the line width for the keypad **FILL** function and the nokeypad **FILL** command because SET SCREEN can accept values only of 80 or 132 in screen modes. EDT uses the SET SCREEN value to determine the line length for filling text only if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses the wrap value, regardless of the SET SCREEN width.

In line mode, EDT always wraps lines that are longer than the SET SCREEN width specifier.

EDT gets the initial screen width from the operating system or from commands either in your login file or an EDT startup command file. Remember that the only screen width possible for VT52 terminals is 80 characters. Using a width specifier of 132 on a VT52 causes confusing line wraps and truncations. If you try to use EDT on a VT52 and your working line length is greater than 80, use QUIT to leave EDT and reset whatever is sending EDT that information. Then start your editing session over again. You can also use EDT's SET SCREEN 80 command at the beginning of your session. Use the **SHOW SCREEN** command to determine the current screen width setting.

When a line exceeds the screen width in a screen mode, EDT indicates that there are additional characters on the line by placing a diamond (♦) in the last column. If the number of characters on your line equals the current screen width exactly, you will see all of them. If you have one more character, you will see one less character than the screen width value. The last column will contain the diamond. In other words, if your screen width is 80 and you have exactly 80 characters in the line, all 80 characters will be visible. If you have 81 or more characters in your line, only the first 79 will be visible.

EXAMPLE

Increases the screen width so that you can see the entire line.

```
This is a line of text that is longer than 80 characters, the screen default wi♦
```

GOLD + PAGE COMMAND

```
Command: SET SCREEN 132
```

ENTER SUBS

```
This is a line of text that is longer than 80 characters, the screen default
width for my VT100.
```

related commands

None.

SET SEARCH Command—Line

SYNTAX

```
SET SEARCH  GENERAL
            EXACT
            WPS
            CASE INSENSITIVE
            DIACRITICAL INSENSITIVE

SET SEARCH  BEGIN
            END

SET SEARCH  BOUNDED
            UNBOUNDED
```

DESCRIPTION

The SET SEARCH command influences how EDT locates strings during your editing sessions. All EDT commands that involve search strings, regardless of editing mode, are affected by the SET SEARCH parameters. The keypad mode functions that use search strings are **FIND**, **FNDNXT**, and **SUBS**. In line mode, both substitute commands and all commands that use a string for the range specifier are affected by the SET SEARCH parameters. In nokeypad mode, SET SEARCH influences any command using either the string specifier or the string entity.

There are three groups of SET SEARCH parameters. The first deals with the case and diacritical marks of letters in the search string as compared with the string EDT finds in your text. The second deals with the cursor position after the string has been found. The third determines how much of the text buffer EDT looks at to find a matching string.

1 SET SEARCH {GENERAL|EXACT|WPS|CASE INSENSITIVE|DIACRITICAL INSENSITIVE} determines how EDT matches the case and diacritical marks of the letters in the search string with the letters in the text. When SET SEARCH GENERAL, the default, is in effect, EDT disregards both case and diacritical marks in performing a search. Thus, **FUR**, **F ÜR**, **Fur**, **F ür**, **fur**, and **f ür** all match the search string **fur**.

With SET SEARCH EXACT, the case and diacritical mark of each letter in the search string must exactly match the case and diacritical mark of each corresponding letter in the string that EDT locates. If the search string is **angel**, EDT ignores **ANGEL**, **Angel** or **ang él** in the text.

SET SEARCH WPS requires that only uppercase letters in the search string match uppercase letters in the text. The search string **The** matches **THE**, **The**, **THe**, or **ThE**, but not **the** or **tHE**.

SET SEARCH CASE INSENSITIVE matches diacritical marks exactly, but disregards the case of the letters. The words **Eon** and **eon** match the search string **EON**; **E ón** and **e ón** do not. Conversely, SET SEARCH DIACRITICAL INSENSITIVE matches the case of letters exactly, but ignores any diacritical marks EDT might encounter. The words **piece** and **pi`ece** match the search string **piece**; **Piece** and **Pi`ece** do not.

2 SET SEARCH {BEGIN|END} determines where the cursor stops after EDT finds the search string when you are working in a screen mode. With SET SEARCH BEGIN, the default, EDT positions the cursor at the first character in the found string. With SET SEARCH END, the cursor stops at the character immediately to the right of the last character in the string. This parameter has no effect in line mode.

3 SET SEARCH {BOUNDED|UNBOUNDED} determines how much of the text EDT searches through when trying to find a string. With SET SEARCH UNBOUNDED, the default, EDT starts its search at the current cursor position or current line and continues to search until it either finds the string or reaches the end of the buffer. If the current direction is forward, EDT stops at the bottom of the buffer. If the current direction is backward, the search stops at the top of the buffer.

When SET SEARCH BOUNDED is in effect, EDT stops the search either when it finds the string or when it reaches a page boundary marker. The default page marker is the form feed (<FF>—CTRL/L). To insert form feeds in your text, simply press CTRL/L at the appropriate locations. If you have used the **SET ENTITY** PAGE command to establish a different page marker, EDT stops at that boundary marker when the search parameter is BOUNDED. If there are no page boundary markers in your buffer, SET SEARCH BOUNDED has the same effect as SET SEARCH UNBOUNDED.

Use the **SHOW SEARCH** command to have EDT display the current search parameters.

EXAMPLES

example 1:

Shows EDT printing different lines, depending on the case of letters in the search string, first with SET SEARCH EXACT and then with SET SEARCH WPS.

```
    1       EDT's screen modes use CUT and PASTE to delete, move,
    2       and copy text.  Cutting the text removes it from its
    3       original location and puts it in a separate buffer.
    4       Use PASTE to put a copy of the cut text in a new
    5       location.

*SET SEARCH EXACT
*TYPE ALL "CUT"

    1       EDT's screen modes use CUT and PASTE to delete, move,

*TYPE ALL "cut"

    4       Use PASTE to put a copy of the cut text in a new

*SET SEARCH WPS
*TYPE ALL "Cut"

    1       EDT's screen modes use CUT and PASTE to delete, move,
    2       and copy text.  Cutting the text removes it from its
```

example 2:

Shows the cursor located on the character after the search string in screen mode (nokeypad in this case).

```
Maynard, MA  01754

EXT SET SEARCH END
EXT TYPE "01754"

Maynard, MA  01754
```

example 3:

Stops EDT from looking for the string after line 254. Therefore, EDT prints the message **String was not found**.

```
  254       After the long trip home, he needed some rest.
  255       <FF>
  256       The next afternoon, Martha came to the house for tea.

*TYPE .

  254       After the long trip home, he needed some rest.

*SET SEARCH BOUNDED
*TYPE "Martha"
String was not found
```

related commands

None.

SET [NO]SUMMARY COMMAND—Line

SYNTAX

SET SUMMARY

SET NOSUMMARY

DESCRIPTION

The SET NOSUMMARY command suppresses the summary information printed at the terminal when you use the **EXIT** or **WRITE** command to copy text from your EDT session into an external file.

Normally with the default state, SET SUMMARY, EDT prints a line of information giving you the complete file specification and number of lines in the file that EDT has created as a result of your EXIT or WRITE command. With SET NOSUMMARY, only the appropriate prompt appears after the command has been processed.

EXAMPLE

Shows the default summary information printed after the EXIT command. Then shows no summary information being supplied when SET NOSUMMARY is in effect.

```
*EXIT
DISK$USER:[SMITH.SUBCAT1]MEMO35.RNO;3 36 lines
$

$ EDIT /EDT MEMO35.RNO
    .
    .
    .
*SET NOSUMMARY
*EXIT
$
```

related commands

None.

SET TAB Command—Line

SYNTAX

SET TAB *number*

SET NOTAB

DESCRIPTION

The SET TAB command establishes the SET TAB value for various tabbing functions in all three editing modes. The default for this command is SET NOTAB. With SET NOTAB in effect only the keypad mode **TAB** (CTRL/H) function and the nokeypad **TAB** command have any effect on your text. The TAB key also works when you are in line mode to insert horizontal tab characters in your text. The remaining tabbing commands and functions do nothing unless you have issued the SET TAB command.

SET TAB activates EDT's tabbing facility that enables you to format layered text, such as outlines and indented computer programs. The keypad tabbing functions are:

CTRL/A (tab compute)	CTRL/E (tab increment)
CTRL/D (tab decrement)	CTRL/T (tab adjust)

The corresponding nokeypad tabbing commands are:

TC (tab compute)	TI (tab increment)
TD (tab decrement)	TADJ (tab adjust)

SET TAB also activates the line mode **TAB ADJUST** command.

The SET TAB value establishes the working value for the tabbing facility. The tab compute, decrement, and increment functions use the SET TAB value to determine the number of tab stops for EDT to indent a line. The tab adjust functions enable you to indent blocks of text.

EXAMPLE

After setting the SET TAB value to 5, uses the TAB key to indent the line 5 columns.

```
PLASTIC RINGS
39   TC076   No. 46 Plastic Rings   .03
```

GOLD + PAGE COMMAND

```
Command: SET TAB 5
```

ENTER SUBS

TAB

```
PLASTIC RINGS
     39   TC076   No. 46 Plastic Rings    .03
```

related commands

None.

SET TERMINAL Command—Line

SYNTAX

SET TERMINAL HCPY
VT100
VT52

SET TERMINAL SCROLL
NOSCROLL

SET TERMINAL EIGHTBIT
NOEIGHTBIT

SET TERMINAL EDIT
NOEDIT

DESCRIPTION

With the SET TERMINAL command, you can correct or change EDT's assumptions about the type of terminal you are using. Generally, EDT is able to get all the information it needs to know about your terminal from the operating system. However, in some instances the data it receives is incorrect. At other times, you might want to change one or more of the settings for some special purpose.

Note that you must *not* issue a SET TERMINAL {VT100|VT52|HCPY} command from a screen mode. You can issue the other SET TERMINAL commands from any mode (for example, SET TERMINAL NOSCROLL).

There are four terminal characteristics that you can reset:

```
basic terminal type:           HCPY, VT100, VT52
scrolling regions:             SCROLL, NOSCROLL
additional character display:  EIGHTBIT, NOEIGHTBIT
enhanced screen editing:       EDIT, NOEDIT
```

If EDT does not support your terminal for screen editing, the terminal should be set for HCPY (hardcopy). This setting allows you to use both line mode and hardcopy change mode, but not the screen modes.

The VT100 setting is for terminals belonging to the VT100 family. These terminals have reverse video display and 22 keypad keys. VT100-type terminals include VT101, VT102, VT125, VT131, and VT132 and terminals that have the LK201 keyboard and VT100 emulation.

The VT52 setting is for VT52 terminals. VT52 terminals have 19 keypad keys and do not have reverse video display.

The SCROLL option shows that your terminal has scrolling regions that EDT can use. This is the default for most VT100-type terminals.

If you do not have a VT100-type terminal, be sure that EDT has your terminal set to NOSCROLL. This is the default for VT52 terminals.

The EIGHTBIT option is for terminals that can display characters requiring eight bits for their representation. The eight bit characters are those in the DEC Multinational Character Set with decimal values of 128 through 255. If your terminal has eightbit character capacity, EDT can display the additional printing characters on your screen in their proper graphic representation, for example, ñ for decimal value 241.

If your terminal cannot display the additional DEC Multinational Character Set characters, you need to use the NOEIGHTBIT option. On NOEIGHTBIT terminals, the characters from 128 through 225 are displayed using symbols inside angle brackets, for example, <A"> for decimal 196. You need to use the keypad **SPECINS** function or the **ASC** command in either nokeypad or hardcopy change mode to insert these characters in your text.

The EDIT option is for terminals, such as the VT102, that have internal screen editing features. These features are: IL (insert line), DL (delete line), the insert state of IRM (insertion-replacement mode), and DCH (delete character). If you have a VT102 or similar terminal and it is set to EDIT, EDT is able to use the terminal's internal screen editing features to enhance performance. Note that terminals using the LK201 keyboard have internal screen editing features. The NOEDIT option is for all terminals that do not have the VT102-type internal screen editing features.

The SET TERMINAL command can be used to turn off features that your terminal has, but it cannot be used to increase the capabilities of your terminal. For instance, if your terminal does not have internal editing capabilities, issuing the SET TERMINAL EDIT command merely confuses EDT and causes problems in your editing session.

EXAMPLES

example 1:

Changes a VT100 or VT52 terminal to the hardcopy setting.

```
*SET TERMINAL HCPY

*SHOW TERMINAL
Hardcopy noscroll noeightbit noedit
```

example 2:

Changes the EDIT/NOEDIT setting to accommodate a VT102 terminal.

```
*SET TERMINAL EDIT

*SHOW TERMINAL
VT100 scroll noeightbit edit
```

related commands

None.

SET TEXT Command—Line

SYNTAX

SET TEXT END *"string"*

SET TEXT PAGE *"string"*

DESCRIPTION

The SET TEXT command allows you to personalize two items during the course of your EDT session. SET TEXT PAGE displays the string you supply for any form feed characters (<FF>) in the buffer. SET TEXT END causes EDT to display your string instead of the [EOB] (end of buffer) mark at the end of each EDT buffer. Once you exit from EDT, these strings are not part of the file.

You can put the SET TEXT END and SET TEXT PAGE commands in a startup command file or you can issue them during your editing session. They have no permanent effect on the text that you are editing.

The strings used for PAGE and END can contain any printing characters (but control characters and DELETE are not allowed). These strings can have several words, but cannot exceed a single line.

EXAMPLE

Changes the characters that EDT displays for <FF> and [EOB].

```
<FF>
I will be in touch with you as soon as I have found a
source for the plastic tubing.
Sincerely yours,
   .
   .
   .
[EOB]

EXT SET TEXT PAGE "PAGE BREAK HERE!"
EXT SET TEXT END "END OF LETTER, DID YOU REMEMBER THE CCs?"

PAGE BREAK HERE!
I will be in touch with you as soon as I have found a
source for the plastic tubing.
Sincerely yours,
   .
   .
   .
END OF LETTER, DID YOU REMEMBER THE CCs?
```

related commands

None.

SET [NO]TRUNCATE Command—Line

SYNTAX

SET TRUNCATE

SET NOTRUNCATE

DESCRIPTION

The SET NOTRUNCATE command causes lines longer than the current screen width to wrap onto subsequent lines when you are working in a screen mode. The default screen width is the terminal width that the operating system reports to EDT at the start of your editing session. This width can be changed by the **SET SCREEN** command.

When you use SET NOTRUNCATE, EDT marks the start of the wrapped portion of a line with a diamond followed by a space. (SET NOTRUNCATE has no effect when you are working in line mode, thus no diamond appears on the screen or paper. In line mode, EDT always wraps long lines.)

SET NOTRUNCATE uses only the screen width value to determine where to break the line. It does not take word boundaries into consideration. Use **SET WRAP** to break lines at word boundaries when you insert text in keypad mode.

SET NOTRUNCATE has no effect on the text in your buffer, only on how that text is displayed on the screen. SET WRAP has a permanent effect on your text.

EXAMPLE

Shows the difference between the text when SET TRUNCATE is in effect, and again when SET NOTRUNCATE is in effect. The SET SCREEN width is 80.

```
We are now planning our new sales campaign for the fall.  I want every sales rep◆
to become fully aware of all the products being marketed by our competition.
```

GOLD + PAGE COMMAND

```
Command: SET NOTRUNCATE
```

ENTER SUBS

```
We are now planning our new sales campaign for the fall.  I want every sales rep
◆ resentative
to become fully aware of all the products being marketed by our competition.
```

related commands

None.

SET [NO]VERIFY Command—Line

SYNTAX

SET VERIFY

SET NOVERIFY

DESCRIPTION

The SET VERIFY command instructs EDT to print at the terminal the commands in a startup command file or EDT macro as the commands are being processed. Under the default condition, SET NOVERIFY, EDT does not display the contents of the startup command file or macro as it is being processed.

EXAMPLE

First shows the startup command file with SET VERIFY at the top. Then shows the start of your EDT session.

Suppose this is your startup command file:

```
SET VERIFY
FIND =TABLE
INCLUDE TABLE.MAC
FIND =MAIN
SET MODE CHANGE
```

Now start to edit a new file.

```
$ EDIT /EDT TABLE35.RNO
FIND =TABLE
INCLUDE TABLE.MAC
FIND =MAIN
SET MODE CHANGE

[EOB]
   .
   .
   .
Input file does not exist
```

related commands

None.

SET WORD [NO]DELIMITER Command—Line

SYNTAX

SET WORD DELIMITER

SET WORD NODELIMITER

DESCRIPTION

The SET WORD NODELIMITER command changes the way EDT responds to word entity boundaries. Normally, EDT considers all word delimiters (except the space) to be words themselves when deleting words or when moving the cursor a word at a time. After you issue the SET WORD NODELIMITER command, EDT still uses the word delimiters to determine where words begin and end, but it does not treat these characters as separate words. SET WORD DELIMITER is the default.

EXAMPLE

First shows the cursor moving to the line terminator when SET WORD DELIMITER is in effect. Then, using the same sample text, shows the cursor moving to the first character of the next line with SET WORD NODELIMITER in effect.

```
During the winter months, people who live in the northern
states like to travel to the south for vacations as well
as for business trips.
```

WORD CHNGCASE + WORD CHNGCASE

```
During the winter months, people who live in the northern_
states like to travel to the south for vacations as well
as for business trips.
```

(Now, move the cursor back to the first **t** in **the northern** and repeat the example, with SET WORD NODELIMITER in effect.)

GOLD + PAGE COMMAND

```
Command: SET WORD NODELIMITER
```

ENTER SUBS

WORD CHNGCASE + WORD CHNGCASE

```
During the winter months, people who live in the northern
states like to travel to the south for vacations as well
as business trips.
```

related commands

None.

SET [NO]WRAP Command—Line

SYNTAX

SET WRAP *number*

SET NOWRAP

DESCRIPTION

The SET WRAP command causes lines of text to wrap when new text is being inserted into a buffer in keypad mode. The command also determines the line length for the keypad **FILL** function and the line and nokeypad mode **FILL** commands. The number specifier tells EDT what the maximum line length should be for the inserted or filled text. SET NOWRAP is the default.

Two important things to remember about SET WRAP are: 1) that it has no effect on text already in your buffer, and 2) that it has no effect on text that you insert in line or nokeypad mode.

When you are inserting text in keypad mode, SET WRAP always breaks the line at a space, line terminator, or other word boundary so that no words are divided. If there are no spaces in the line, you can type up to 255 characters on a line and still have no wrapping take place, regardless of the SET WRAP value. If you are inserting characters in the middle of an existing line in such a way that no spaces are added, no break occurs, even if there are spaces elsewhere in the line.

SET WRAP has no effect on lines that are already in a buffer, but once set, it has a permanent effect on text that is inserted in keypad mode or text that has been reformatted by a FILL function. When a wrap takes place, EDT considers the wrapped portion to be a separate line and numbers it accordingly. So, although *you* did not type a line terminator, one now exists at the spot where the wrap was made. The wrapped portion of the line has its own line number and maintains its separateness during your editing session and in any external files that EDT creates from that text.

When you use SET WRAP with FILL, the number specified for the SET WRAP command is used by EDT to determine the maximum line length. If SET NOWRAP is in effect, the line length for FILL is determined by the **SET SCREEN** value. However, SET WRAP overrides SET SCREEN.

In screen mode, the **SET NOTRUNCATE** command causes EDT to wrap all lines that exceed the current SET SCREEN setting. SET NOTRUNCATE breaks the line at the SET SCREEN width, regardless of word boundaries. NOTRUNCATE has no permanent effect on your text.

EXAMPLES

example 1:

Before entering keypad mode, sets the SET WRAP value to 40. EDT then wraps the text as you insert it.

```
*SET WRAP 40
*CHANGE
```

(Here is the line as you start to type it.)

```
As we type this line, we pause just befo_
```

(Here is the text as it appears when you finish.)

```
As we type this line, we pause just
before the 41st character position._
```

example 2:

Sets the SET WRAP value to 40 and then uses the FILL nokeypad command to reformat the text.

```
You can devise command procedures to simplify and
enhance your program development.
For example,
you can write a command procedure that will
compile, link, and run a specific
PL/I program.

EXT SET WRAP 40
FILL6L

You can devise command procedures to
simplify and enhance your program
development. For example, you can write
a command procedure that will compile,
link, and run a specific PL/I program.

_
```

related commands

None.

SHOW AUTOREPEAT Command—Line

SYNTAX

SHOW AUTOREPEAT

DESCRIPTION

This command indicates whether **SET AUTOREPEAT** or SET NOAUTOREPEAT is currently in effect. The default for EDT is generally AUTOREPEAT. Use SET NOAUTOREPEAT to change the default if your terminal requires that setting.

EXAMPLE

Prints the current autorepeat setting.

```
*SHOW AUTOREPEAT
autorepeat
```

related commands

None.

SHOW BUFFER Command—Line

SYNTAX

SHOW BUFFER

DESCRIPTION

The SHOW BUFFER command lists all accessible buffers currently in your EDT session. Inaccessible storage areas, such as the delete character, delete word, delete line, search, and substitute buffers, do not appear in the SHOW BUFFER list. EDT uses the buffer specifier in both line mode and nokeypad mode to create buffers.

When you issue the SHOW BUFFER command, EDT indicates the current buffer by preceding its name with an equal sign (=). EDT also lists the number of lines in each buffer. If the number of lines in the MAIN buffer is marked with an asterisk (*), it means that EDT has not read through the entire buffer and therefore is only aware of the number of lines shown. You can have EDT "read" the entire buffer by issuing a command such as **TYPE** END.

Once a buffer has been created, it remains open for the remainder of the EDT session, unless it is deleted by the **CLEAR** command. The MAIN and PASTE buffers are automatically created by EDT at the start of every editing session. These buffers cannot be eliminated by the CLEAR command, although their contents can be deleted.

EXAMPLE

First shows the current buffer status at the start of your EDT session, when EDT has seen only the first line of the MAIN buffer. Then shows the buffer status after you have used the TYPE command to move to the end of the buffer.

```
$ EDIT/EDT LETTER.RNO

    1       March 4, 1985

*SHOW BUFFER

=MAIN    1*       lines
 PASTE   No       lines

*TYPE END

[EOB]

*SHOW BUFFER

=MAIN    45       lines
 PASTE   No       lines
```

related commands

None.

SHOW CASE Command—Line

SYNTAX

SHOW CASE

DESCRIPTION

The SHOW CASE command tells you which case, if any, has been established by the **SET CASE** command. The possible responses are: Upper, Lower, or None. None is the default.

EXAMPLE

Shows the current case setting.

```
*SHOW CASE
None
```

related commands

None.

SHOW COMMAND Command—Line

SYNTAX SHOW COMMAND

DESCRIPTION

The SHOW COMMAND command prints the name of the active startup command file. Include SHOW COMMAND in the startup command file to have EDT print the name of that file at the start of your EDT session.

When the **SET COMMAND** command appears in a startup command file, it causes EDT to look for another startup command file. You can include a SHOW COMMAND command in each startup command file so that you will see which files have been processed before you begin your editing session. If you issue the SHOW COMMAND command during your editing session, EDT displays nothing.

EXAMPLE

In this example, each SHOW COMMAND command is in a separate startup command file. The file name that EDT prints appears as soon as EDT completes processing that startup command file.

(First startup command file)

```
    .
    .
    .
SET MODE CHANGE
SHOW COMMAND
SET COMMAND EDTEXPERT
```

(Second startup command file)

```
    .
    .
    .
DEFINE KEY GOLD E AS "EXT EXIT."
SHOW COMMAND

$ EDIT/EDT ACCTPROG.COB
EDTSYS
EDTEXPERT
    .
    .
    .
```

related *commands*

None.

SHOW CURSOR Command—Line

SYNTAX

SHOW CURSOR

DESCRIPTION

The SHOW CURSOR command displays the values that have been set by the **SET CURSOR** command. The SET CURSOR command establishes the cursor positions at which scrolling of the screen image occurs. The default values are 7:14—that is, the screen image scrolls up when the cursor is on the 15th line of the screen and you are moving forward. The screen image scrolls down when the cursor is on the 8th line of the screen and you are moving backward. (The SET CURSOR values for the 22 lines on the screen range from 0 to 21.)

EXAMPLE

First shows the default cursor values. Then after a new set has been established, shows the new values.

```
*SHOW CURSOR
7:14
   .
   .
   .

*SET CURSOR 2:20
   .
   .
   .

*SHOW CURSOR
2:20
```

related commands

None.

SHOW ENTITY Command—Line

SYNTAX

SHOW ENTITY WORD

SHOW ENTITY SENTENCE

SHOW ENTITY PARAGRAPH

SHOW ENTITY PAGE

DESCRIPTION

The SHOW ENTITY command lists the current delimiters that determine the boundaries of four entities: WORD, SENTENCE, PARAGRAPH, and PAGE. Use the **SET ENTITY** command to change the boundaries for these entities.

EXAMPLES

example 1:

These examples show the default boundary settings for each entity.

```
*SHOW ENTITY WORD
        <LF><VT><FF><CR>

*SHOW ENTITY SENTENCE
.!?

*SHOW ENTITY PARAGRAPH
<CR><CR>

*SHOW ENTITY PAGE
<FF>
```

related commands

None.

SHOW FILES Command—Line

SYNTAX

SHOW FILES

DESCRIPTION

The SHOW FILES command displays the current input file and output file for your EDT session. The input file is displayed exactly as you entered it on the EDIT/EDT command line. If you use the /OUTPUT qualifier in the EDIT /EDT command line, SHOW FILES displays the output file exactly as you typed it. If you do not include an output file specification in the command line, SHOW FILES displays the same information for both the input file and the output file.

EXAMPLES

example 1:

Displays the input and output files when the /OUTPUT qualifier has been used.

```
$ EDIT /EDT DATALIST.DAT
*SHOW FILES
Input  File:  DATALIST.DAT
Output File:  DATALIST.DAT
```

example 2:

Displays the input and output files when no output file is specified in the EDIT/EDT command line.

```
$ EDIT /EDT /OUTPUT=SMITH.RNO LETTER.RNO
*SHOW FILES
Input  File:  LETTER.RNO
Output File:  SMITH.RNO
```

related commands

None.

SHOW FNF Command—Line

SYNTAX

SHOW FNF

DESCRIPTION

The SHOW FNF command indicates whether **SET FNF** or SET NOFNF is in effect. (FNF stands for File Not Found.) The default is SET FNF, meaning that the message **Input file does not exist** is displayed whenever you create a new file with EDT. Put the SET NOFNF command in your startup command file to suppress that message. SET NOFNF does not suppress the message **Input file does not have standard text file format**.

EXAMPLE

Shows the current FNF setting.

```
*SHOW FNF
nofnf
```

related commands

None.

SHOW HELP Command—Line

SYNTAX

SHOW HELP

DESCRIPTION

The SHOW HELP command tells you which EDT HELP file is currently available for your editing session. The default HELP file is:

```
SYS$HELP:EDHELP.HLB
```

If you or your site has created a customized HELP file, and you are not sure which HELP file version is currently in use, you can use the SHOW HELP command to find that information. Use the **SET HELP** command to access a different HELP file.

EXAMPLE

First shows the default system HELP file. After you issue a SET HELP command to access a different HELP file, shows the new current HELP file.

```
*SHOW HELP
Help file name:  SYS$HELP:EDTHELP.HLB;1
*SET HELP SYS$HELP:MYEDTHELP.HLB
   .
   .
   .
*SHOW HELP
Help file name:  SYS$HELP:MYEDTHELP.HLB;3
```

related commands

None.

SHOW KEY Command—Line

SYNTAX

SHOW KEY [GOLD] *keypad-key-number*

SHOW KEY GOLD *character*

SHOW KEY [GOLD] CONTROL *character*

SHOW KEY [GOLD] DELETE

SHOW KEY [GOLD] FUNCTION *key-number*

DESCRIPTION

The SHOW KEY command tells you the definition of any keys that have keypad editing functions. It shows you the definition for any key that EDT has preset, such as the keypad keys and the preset control keys, as well as any key that you have defined or redefined during your editing session. (Once you have redefined a preset key, SHOW KEY shows only the new definition.)

Because keypad mode function keys are defined in terms of nokeypad commands, the definition that is displayed for the SHOW KEY command uses the nokeypad command syntax. For example, when you ask for the definition of the CONTROL A key, EDT prints **TC.** in response.

All preset key definitions, except **RESET** and **GOLD**, end with a period. The period indicates that the operation is set into motion as soon as you press the key or key sequence. The RESET and GOLD function definitions are not nokeypad commands; they are special EDT keywords. For a keypad key to have the RESET function, it must be defined exactly as RESET with no period. The same is true for GOLD. Neither RESET nor GOLD can be part of a multicommand definition.

When using the SHOW KEY command, you must type the words GOLD and CONTROL in your commands. If you are looking for the definition of the DELETE key, you must type the word DELETE. To find the definition of a function key on the LK201 keyboard, type the word FUNCTION.

When you want to find the definition for the **BACKSPACE**, **LINEFEED** or **TAB** key, you must use the key's CONTROL equivalent: CONTROL H for BACKSPACE, CONTROL J for LINEFEED, and CONTROL I for TAB. For terminals with LK201 keyboards, you need to use the FUNCTION numbers for the F12 and F13 keys: FUNCTION 24 and FUNCTION 25.

Keypad-key-number refers to EDT's numeric designations for the keypad keys. For example, the period key on the keypad is number 16, the ENTER key is number 21, and the down arrow key is number 13.

FUNCTION key refers to the additional keys on the LK201 keyboard that you can define. These include the six keys on the terminal's "editing" keypad, located above the arrow keys, as well as the keys on the function key row from F6 through F20. You cannot define keys F1 through F5 on the function key row. EDT uses its own numbers for these keys, ranging from FUNCTION 1 through FUNCTION 99, to encompass the special user define keys (UDKs) that are allowed on terminals with LK201 keyboards.

EXAMPLE

First shows the default definition for the CTRL/N key sequence. Then shows the new definition after CTRL/N has been defined with the DEFINE KEY command.

```
*SHOW KEY CONTROL N
No definition

*DEFINE KEY CONTROL N AS "?'Nokeypad Command: '."

*SHOW KEY CONTROL N
?'Nokeypad Command: '.
```

OTHER EXAMPLES

1
```
*SHOW KEY 7
PAGETOP.
```
Shows the default definition for the 7 key on the keypad.

2
```
*SHOW KEY 16
SEL.
```
Shows the default definition for the period (.) key on the keypad.

3
```
*SHOW KEY GOLD 16
RESET
```
Shows the default definition for GOLD+. keypad key sequence.

4
```
*SHOW KEY CONTROL U
DBL.
```
Shows the default definition for the CTRL/U key sequence.

5
```
*SHOW KEY DELETE
D-C
```
Shows the default definitiongfor the DELETE key.

6
```
*SHOW KEY 20
GOLD
```
Shows the default definition for the GOLD key.

7
```
*SHOW KEY FUNCTION 5
 (-16L).
```
Shows the default definition for the Prev Screen key on the LK201 keyboard editing keypad.

8
```
*SHOW KEY FUNCTION 24
BL.
```
Shows the default definition for the F12 key on the LK201 function key row.

related commands

None.

SHOW KEYPAD Command—Line

SYNTAX

SHOW KEYPAD

DESCRIPTION

The SHOW KEYPAD command tells you which screen editing mode is in effect: KEYPAD or NOKEYPAD. Use the **SET [NO]KEYPAD** command to reset your session to a different screen mode.

EXAMPLE

Shows the current screen mode setting from line mode when SET NOKEYPAD is in effect.

```
*SHOW KEYPAD
nokeypad
```

related commands

None.

SHOW LINES Command—Line

SYNTAX

SHOW LINES

DESCRIPTION

The SHOW LINES command tells you the number of lines that EDT is displaying on the screen at one time. The default (and maximum) number is 22. Rather than count the number of lines being displayed, you can use SHOW LINES. The **SET LINES** command resets the number of lines per screen. The number of lines ranges from 1 to 22. The **SET CURSOR** command resets the cursor positions that activate the scrolling mechanism.

EXAMPLE

First shows the default setting for LINES. Then shows the new value after SET LINES 10 is in effect.

```
*SHOW LINES
22
*SET LINES 10
    .
    .
    .
*SHOW LINES
10
```

related commands

None.

SHOW MODE Command—Line

SYNTAX

SHOW MODE

DESCRIPTION

The SHOW MODE command tells you which **SET MODE** command was most recently issued. The response does not necessarily correspond to your current editing mode if you have used the line mode **CHANGE** command, the nokeypad **EX** command, or the keypad **CTRL/Z** function to shift editing modes.

EXAMPLE

Shows that SET MODE CHANGE was the most recent SET MODE command.

GOLD + PAGE COMMAND

```
Command: SHOW MODE
Change
```

ENTER SUBS

```
Command: SHOW MODE
Change
```

related commands

None.

SHOW NUMBERS Command—Line

SYNTAX

SHOW NUMBERS

DESCRIPTION

The SHOW NUMBERS command tells you if the EDT line numbers are being displayed in line mode. When SET NONUMBERS is in effect, EDT does not display the line numbers in line mode. Your text begins at the left margin of your screen or paper. However, even when you have used SET NONUMBERS, EDT continues to keep track of the EDT line numbers. Thus, you can still use them in line mode range specifiers. The default is **SET NUMBERS**.

EXAMPLE

Shows that SET NONUMBERS is in effect.

```
*SHOW NUMBERS
nonumbers
```

related commands

None.

SHOW PARAGRAPH Command—Line

SYNTAX

SHOW PARAGRAPH

DESCRIPTION

The SHOW PARAGRAPH command tells you whether **SET PARAGRAPH WPS** or SET PARAGRAPH NOWPS is currently in effect for your editing session. SET PARAGRAPH NOWPS is the default. SET PARAGRAPH WPS causes EDT to position the cursor on the first character after the default paragraph boundary that is not a line terminator.

EXAMPLE

Shows that SET PARAGRAPH NOWPS is in effect.

```
*SHOW PARAGRAPH
nowps
```

related commands

None.

SHOW PROMPT Command—Line

SYNTAX

SHOW PROMPT *prompt-type*

DESCRIPTION

The SHOW PROMPT command tells you which prompt settings are currently in effect. The **SET PROMPT** commands are used with EDT's testing facility to signal that EDT is ready for simulated user input.

The prompt types cannot be abbreviated. They are:

LINE	KEYPAD	NOKEYPAD	HCCHANGE
INSERT	INSERTN	QUERY	

EXAMPLE

These examples show all the default SET PROMPT settings.

```
*SHOW PROMPT LINE
<CR><LF>*
*SHOW PROMPT KEYPAD
*SHOW PROMPT NOKEYPAD
*SHOW PROMPT HCCHANGE
<CR><LF>C*
*SHOW PROMPT INSERT
             --------------
<CR><LF>!--------------!
*SHOW PROMPT INSERTN
<CR><LF>
*SHOW PROMPT QUERY
<CR><LF>?
```

related commands

None.

SHOW QUIET Command—Line

SYNTAX

SHOW QUIET

DESCRIPTION

The SHOW QUIET command tells you whether the bell, which sounds when EDT returns an error message in screen mode, has been turned off. With **SET QUIET** in effect, the terminal bell does not sound when EDT issues an error message in one of the screen modes. SET NOQUIET is the default. SET QUIET has no effect on the nokeypad BELL command.

EXAMPLE

First shows the default setting. Then shows the new setting after SET QUIET is in effect.

```
*SHOW QUIET
noquiet
*SET QUIET
    .
    .
    .
*SHOW QUIET
quiet
```

related commands

None.

SHOW REPEAT Command—Line

SYNTAX

SHOW REPEAT

DESCRIPTION

The SHOW REPEAT command lets you know whether you can use the GOLD repeat feature and the **SPECINS** function in keypad mode. **SET REPEAT** is the default. With SET NOREPEAT, you cannot use GOLD and a digit to repeat a keypad mode function nor can you use the SPECINS function.

EXAMPLE

Shows the current REPEAT setting.

```
*SHOW REPEAT
repeat
```

related commands

None.

SHOW SCREEN Command—Line

SYNTAX

SHOW SCREEN

DESCRIPTION

The SHOW SCREEN command tells you the current screen width setting. The default screen width is the terminal's width as determined by the operating system. For VT100-type terminals and VT52s the default width is usually 80. VT52 terminals and VT100-type terminals without advanced video option (AVO) cannot accommodate screen widths greater than 80 characters. The only valid screen width settings for screen mode editing are 80 and 132. Line mode can use a screen width value from 1 to 132. Use the **SET SCREEN** command to change the screen width.

EXAMPLE

First shows the current screen width. Then shows the new screen width after SET SCREEN 132 is in effect.

```
*SHOW SCREEN
80
*SET SCREEN 132
    .
    .
    .
*SHOW SCREEN
132
```

related commands

None.

SHOW SEARCH Command—Line

SYNTAX SHOW SEARCH

DESCRIPTION

The SHOW SEARCH command tells you the current search parameters that EDT uses to locate strings in your text. There are three sets of parameters: 1) those that deal with the case or diacritical marks of letters, 2) those that determine the position of the cursor after the search is completed, and 3) those that determine what portion of the buffer EDT will search to find the string.

The first set contains GENERAL (the default), EXACT, WPS, CASE INSENSITIVE (CI), AND DIACRITICAL INSENSITIVE (DI). GENERAL means that EDT ignores case differences and diacritical marks in searching for strings. EXACT means that all case differences and diacritical marks are considered when matching the search string. WPS means that the case of uppercase letters must be matched exactly when locating search strings. With CASE INSENSITIVE, case is ignored, but diacritical marks are not. Conversely, with DIACRITICAL INSENSITIVE, diacritical marks are ignored, but case differences are not.

The second set of parameters, BEGIN and END, determines the new cursor position after EDT locates the string in a screen mode. With BEGIN, the default, the cursor is on the first character of the string. END puts the cursor on the character to the right of the last string character.

The third set of parameters is BOUNDED or UNBOUNDED. UNBOUNDED, the default, means that EDT searches from the current cursor position to the top or bottom of the buffer, depending on the direction of the search. BOUNDED limits the search from the current cursor position to the next PAGE boundary mark that EDT encounters in the current direction. If there are no PAGE markers in your text, the effect of BOUNDED and UNBOUNDED is the same. The default PAGE marker is the form feed (<FF>—CTRL/L). Use the **SET ENTITY** PAGE command to change the PAGE boundary marker.

The search parameters are used in keypad mode with the **FIND**, **FNDNXT**, and **SUBS** functions. In nokeypad mode, the search parameters are used with the string entity, the string specifier, and the substitute commands. Line mode uses search parameters with substitute operations as well as with the string range specifier.

EXAMPLE

First, shows the default search settings. Then shows the new ttings after SET SEARCH EXACT and SET SEARCH END are in effect.

```
*SHOW SEARCH
general begin unbounded
*SET SEARCH EXACT
*SET SEARCH END
    .
    .
    .
*SHOW SEARCH
exact end unbounded
```

related commands

None.

SHOW SUMMARY Command—Line

SYNTAX

SHOW SUMMARY

DESCRIPTION

The SHOW SUMMARY command tells you whether the **SET SUMMARY** feature is in effect for your editing session. If the SET NOSUMMARY command has been given, EDT does not display the summary file information after processing either an **EXIT** or **WRITE** command. SET SUMMARY is the default.

EXAMPLE

First, uses the SET NOSUMMARY command to change the default. Then shows the current SUMMARY setting. Finally, shows the effects of SET NOSUMMARY when you issue the EXIT command.

```
*SET NOSUMMARY
   .
   .
   .
*SHOW SUMMARY
nosummary
   .
   .
   .
*EXIT
$
```

related commands

None.

SHOW TAB Command—Line

SYNTAX SHOW TAB

DESCRIPTION

The SHOW TAB command tells you the current **SET TAB** value (or size) as well as the current tab indentation level count. If no tab value is currently in use, EDT prints **notab** in response to the SHOW TAB command. NOTAB is the default. Use the SET TAB command to establish a tab value.

In order for the following commands and functions to have any effect on your text, you must establish a SET TAB value:

Keypad		Nokeypad	Line Mode
CTRL/A	(tab compute)	TC	
CTRL/D	(tab decrement)	TD	
CTRL/E	(tab increment)	TI	
CTRL/T	(tab adjust)	TADJ	TAB ADJUST

The tab level count displayed by the SHOW TAB command is not affected by any of the following:

- The **n** specifier in the line mode TAB ADJUST command
- A count specifier with the nokeypad TADJ command
- A repeat count used with the keypad CTRL/T function

EXAMPLES

example 1: Shows the default TAB setting.

```
*SHOW TAB
notab
```

example 2: Shows the TAB setting after SET TAB 10 is in effect.

```
*SET TAB 10
    .
    .
    .
*SHOW TAB
tab size 10; tab level 1
```

example 3: First sets the SET TAB value to 5 and shifts to keypad mode. Then uses CTRL/A to reset the tab level count. Shows the new tab size and tab level.

```
*SET TAB 5
*CHANGE
```

GOLD + @ 2 +) 0 + →

CTRL/A

GOLD + PAGE COMMAND

```
Command:  SHOW TAB
```

ENTER SUBS

```
tab size 5; tab level 4
```

---

related commands

None.

SHOW TERMINAL Command—Line

SYNTAX SHOW TERMINAL

DESCRIPTION

The SHOW TERMINAL command shows the terminal settings that are currently in effect for your editing session. Normally, EDT receives the terminal settings from the operating system when you start your editing session. You can use the **SET TERMINAL** command to change these default settings if EDT is receiving incorrect or insufficient information about the terminal you are using.

There are four groups of terminal settings. The first group indicates the terminal type: VT100, VT52, HCPY (hardcopy). The second group refers to scrolling regions in your terminal: SCROLL, NOSCROLL. The third group refers to the number of bits that your terminal uses to represent an individual character: EIGHTBIT, NOEIGHTBIT. The fourth group refers to internal editing features built into the terminal: EDIT, NOEDIT.

EXAMPLE

Shows terminal settings for a variety of terminals.

General VT100-type terminal with advanced video option

```
*SHOW TERMINAL
VT100 scroll noeightbit noedit
```

VT52 terminal

```
*SHOW TERMINAL
VT52 noscroll noeightbit noedit
```

Hardcopy terminal or terminal not supported for screen mode editing.

```
*SHOW TERMINAL
Hardcopy noscroll noeightbit noedit
```

VT102 terminal

```
*SHOW TERMINAL
VT100 scroll noeightbit edit
```

related commands

None.

SHOW TEXT Command—Line

SYNTAX

SHOW TEXT END

SHOW TEXT PAGE

DESCRIPTION

The SHOW TEXT command tells you what text EDT is currently displaying for the <FF> page mark or the [EOB] (end of buffer) mark. Use the **SET TEXT** PAGE and SET TEXT END commands to customize these messages. Neither the SET TEXT END command nor the SET TEXT PAGE command has any effect on the text you are editing.

EXAMPLE

Shows the new text that EDT will display in place of <FF> page mark or the [EOB] (end of buffer) mark after you have issued the SET TEXT commands.

```
*SET TEXT END "No more lines in this buffer."
*SET TEXT PAGE "Oh no, not another page!"
   .
   .
   .
*SHOW TEXT END
No more lines in this buffer.
   .
   .
   .
*SHOW TEXT PAGE
Oh no, not another page!
```

related commands

None.

SHOW TRUNCATE Command—Line

SYNTAX

SHOW TRUNCATE

DESCRIPTION

The SHOW TRUNCATE command tells you whether **SET TRUNCATE** or SET NOTRUNCATE is currently in effect for your editing session. The default is SET TRUNCATE. In screen editing, NOTRUNCATE causes EDT to display the portion of a line that exceeds the current screen width on the next line. SET NOTRUNCATE does not add line terminators to your text. Use **SET WRAP** to add line terminators to your text when inserting text in keypad mode or filling text in all three editing modes. Lines exceeding the current **SET SCREEN** width are always wrapped in line mode.

EXAMPLE

Shows that SET NOTRUNCATE is currently in effect.

```
*SHOW TRUNCATE
notruncate
```

related commands

None.

SHOW VERIFY Command—Line

SYNTAX

SHOW VERIFY

DESCRIPTION

The SHOW VERIFY command tells you whether **SET VERIFY** or SET NOVERIFY is currently in effect for your editing session. If SET VERIFY is in effect, EDT displays the commands in the startup command file or EDT macro that are being processed. SET NOVERIFY is the default.

EXAMPLE

Shows that the default NOVERIFY is in effect.

```
*SHOW VERIFY
noverify
```

related commands

None.

SHOW VERSION Command—Line

SYNTAX SHOW VERSION

DESCRIPTION

The SHOW VERSION command displays the version of EDT that is being used by your operating system. The command also displays EDT's copyright notice. There is no way for you to reset the version of EDT currently available on your system from within EDT. (EDT has no SET VERSION command.) However, this information is useful when reporting EDT problems.

The version syntax is:

```
support version.update-edit patch
```

Support is a letter that indicates the support status of the version of EDT that you are using. The possible letters are:

B	Benchmark configuration; no support
D	Demonstration configuration; no support
S	Special customer configuration; negotiated support
T	Field test version; not supported after field test ends
V	Released version; full support
X	Experimental version; no support.

Version is the version number. If it does not match the number on the title page of this manual, the book does not apply to your version of EDT. **Update** represents an update to the base version. The update involves minor changes to the documentation. Check to be sure that this number is the same as that given in the front of this manual.

Edit represents a minor change to EDT that has no documentation impact. **Patch** is an optional letter, which represents a minor change to EDT with no documentation impact. The change has been applied by means of a binary patch.

When you fill out a Software Performance Report (SPR) for EDT, be sure to include the full version number that EDT displays when you issue the SHOW VERSION command. Use the space provided on the SPR form to enter the version number.

EXAMPLE

Explains the meaning of the version number.

```
*SHOW VERSION
V3.00-1         COPYRIGHT (c) DIGITAL EQUIPMENT CORPORATION 1983
```

This is a fully supported released version. The version number is **3**. The **00** indicates that this version has no updates. The **1** in the edit position indicates a minor change that has no impact on the documentation. No patch letter is present.

related commands

None.

SHOW WORD Command—Line

SYNTAX

SHOW WORD

DESCRIPTION

The SHOW WORD command tells you how EDT is interpreting word boundaries when you use the word entity in keypad or nokeypad mode. When NODELIMITER is in effect, EDT does not consider word delimiters, such as line terminators and horizontal tabs, to be words themselves. (The space word boundary is always considered to be part of a word, never as a separate word.) The default is **SET WORD DELIMITER**.

EXAMPLE

Shows that SET WORD NODELIMITER is currently in effect.

```
*SHOW WORD
nodelimiter
```

related commands

None.

SHOW WRAP Command—Line

SYNTAX

SHOW WRAP

DESCRIPTION

The SHOW WRAP command tells you if a **SET WRAP** command is in effect and if so, what the SET WRAP value is. SET WRAP has two functions. It causes EDT to wrap lines when you are inserting text in keypad mode and it sets the right margin for all three EDT **FILL** functions. SET WRAP always overrides the current **SET SCREEN** width in determining the line length for FILL functions.

In the screen modes, **SET NOTRUNCATE** wraps lines that are already in your buffer for display purposes only; it has no effect on the text.

EXAMPLES

example 1:

Shows that the default NOWRAP setting is in effect.

```
*SHOW WRAP
nowrap
```

example 2:

Shows that SET WRAP 50 is currently in effect.

```
*SHOW WRAP
50
```

related commands

None.

/STAY QUALIFIER—Line

SYNTAX /STAY

DESCRIPTION

The /STAY qualifier is used with the line mode **TYPE** command. All EDT qualifiers must be preceded by a slash. /STAY keeps the EDT pointer at its current position, regardless of which lines have just been displayed by the TYPE command.

EXAMPLES

example 1:

Shows that EDT remains on line 35 even after typing lines 2 through 4.

```
   35       whenever you are ready to send me the information,

*TYPE 2 THRU 4 /STAY

    2       August 17, 1985
    3
    4       Dr. Hugo Arndt

*TYPE .

   35       whenever you are ready to send me the information,
```

example 2:

Displays the contents of the PASTE buffer while still in the MAIN buffer.

```
*TYPE =PASTE /STAY

    1       whether or not you have made your decision.
    2       But, of course, we would welcome your joining
[EOB]

*TYPE .

  284       the details of our current business plan.
```

related commands

None.

string specifier—Line

SYNTAX

"string"
'string'
/string-1/string-2/

DESCRIPTION

The string specifier is generally used either to locate characters in a buffer or to replace the located characters. When a string specifier is used to locate a piece of text, it is referred to as the search string. All three editing modes use search strings. Generally a search string must be enclosed in quotation marks (single (') or double (")) to distinguish it from other characters in the command line.

When strings are used in EDT substitute commands, they are generally referred to as string-1 or string-2. String-1 is always a search string. EDT must locate that string and then replace it with string-2, the substitute string. Strings in substitute commands must be surrounded by delimiters to separate them from the other characters in the command line. A variety of nonalphanumeric characters can be used as delimiters as long as the delimiter does not appear in either string-1 or string-2. (In line mode you cannot use the percent sign (%) or the underscore (_) as string delimiters.) All delimiters used in a single substitute command line must be identical.

Whenever you issue a search string, EDT overwrites the contents of the search buffer. Similarly, when you issue a substitute string, EDT overwrites the contents of the substitute buffer.

The search and substitute buffers cannot be edited or entered. Their names never appear in the SHOW BUFFER list. You can use the nokeypad **CLSS** (clear search string) command to delete the contents of the search buffer.

EDT has a number of ways to perform searches. See the discussion of the **SET SEARCH** command for information about the EDT search parameters.

EXAMPLES

example 1:

Deletes the line containing the string **afternoon**.

```
*DELETE "afternoon"
```

example 2:

Displays the line containing the string **January**.

```
*TYPE "January"
```

example 3:

Finds the next occurrence of the string **line** and replaces it with the string **keypad**.

```
*SUBSTITUTE NEXT/line/keypad/
```

SUBSTITUTE Command—Line

SYNTAX

SUBSTITUTE*/[string-1]/string-2/ [=buffer] [range]*
[/*BRIEF*[:n]] [/*QUERY*] [/*NOTYPE*]

DESCRIPTION

The SUBSTITUTE command replaces string-1 with string-2 everywhere in the specified range. If no range is given, the substitution is made on the first occurrence of the string in the current line. You must reissue the SUBSTITUTE command or use the **SUBSTITUTE NEXT** command to change subsequent occurrences.

String-1 and string-2 must be surrounded by delimiters. You can use any nonalphanumeric character as a delimiter (except the percent sign (%) and the underscore (_)) provided that the character does not appear in either string-1 or string-2. The three delimiters must be identical. Notice that the delimiters are not optional.

The SUBSTITUTE command displays the lines in which substitutions were made as well as a message indicating the number of substitutions. If EDT is unable to process the command, it prints the message **No substitutions**.

The SUBSTITUTE command can take several specifiers and qualifiers. The range specifier tells EDT to make as many substitutions as there are occurrences of string-1 in the range. If your range covers only one line, but there are two occurrences of string-1 in that line, both substitutions are made. Without the range specifier, only the first substitution would be made.

You can also perform substitutions in other buffers. If you do not use a range specifier following the buffer name, all occurrences of string-1 in the specified buffer are found and replaced with string-2. EDT leaves you in the specified buffer. You must use another command to return to your original buffer.

You do not have to include both string-1 and string-2 in your SUBSTITUTE command, but you must include at least one of them. If you omit string-1, EDT always uses the current search string. If you omit string-2, EDT always deletes string-1 and inserts nothing. If you use a string for range, EDT considers that to be the current search string if string-1 is omitted. The following samples show what happens:

```
This is file A.     SUBSTITUTE/file/buffer/     This is buffer A.
This is file B.     SUBSTITUTE///               Search string cannot be null
This is file C.     SUBSTITUTE/file//           This is  C.
This is file D.     SUBSTITUTE//buffer/         This is buffer D.
This is file E.     SUBSTITUTE//buffer/ "e."    This is file buffer
```

The **/BRIEF** qualifier limits the number of characters displayed for each line in which a substitution is made. The default value for the **n** specifier is 10. By using the **n** specifier, you can have more or fewer than 10 characters displayed. When the /BRIEF qualifier is not used, the entire line is printed, unless the /NOTYPE qualifier is in effect.

The **/QUERY** qualifier prints the line containing string-1 before the substitution is made. EDT then prompts you with the question mark (?) to let you decide whether to make the substitution in that line or go on to the next occurrence of string-1. Type one of these four responses to the question mark prompt: Y (YES), N (NO), A (ALL), or Q (QUIT).

When you use the **/NOTYPE** qualifier, EDT does not display the lines after making the substitutions. However, EDT does print the message that tells you how many substitutions were made.

Note: The slashes preceding the qualifiers /BRIEF, /QUERY, and /NOTYPE are signals to EDT, not delimiters. Only the slash can indicate that the following letters refer to a qualifier.

EXAMPLES

example 1: Replaces the first occurrence of **this editor** with **EDT** on the current line.

```
   95       with this editor.  This editor can also

*SUBSTITUTE/this editor/EDT/

   95       with EDT.  This editor can also
1 substitution
```

example 2: Replaces both occurrences of **this editor** with **EDT** on the current line.

```
   95       with this editor.  This editor can also

*SUBSTITUTE/this editor/EDT/ 95

   95       with EDT.  EDT can also
2 substitutions
```

(Note that **SUBSTITUTE/this editor/EDT/ .** produces the same results.)

example 3: Replaces all occurrences of **April 19** with **May 16** in lines 15 through 30.

```
   15       The meeting is scheduled for
   16       April 19, 1985 at 7:00 p.m. at the Campus Inn.
   .
   .
   .
   29       All programmers planning to attend the April 19th
   30       meeting should contact Marsha Lambert as soon as

*SUBSTITUTE/April 19/May 16/ 15 THRU 30

   16       May 16, 1985 at 7:00 p.m. at the Campus Inn.
   29       All programmers planning to attend the May 16th
2 substitutions
```

example 4: Replaces **2B** with **1C** for the entire buffer, but uses the /QUERY qualifier to let you decide individually which instances to replace. Note that the delimiter character is the exclamation point (!).

```
    1       John Hershey            2A
    2       Max Greenstein          2B
    3       Jennifer Grogan         2B
    4       Larry Sadler            2B
    5       Quincy Marcus           2A
    6       Shirley Green           2A
    7       Thomas Orlovsky         2B
    8       Theodore Rossmann       2B
    9       Marion Andrews          2B

*SUBSTITUTE!2B!1C! WHOLE /QUERY

    2       Max Greenstein          2B
?Y
    2       Max Greenstein          1C
    3       Jennifer Grogan         2B
?N
    4       Larry Sadler            2B
?Y
    4       Larry Sadler            1C
    7       Thomas Orlovsky         2B
?N
    8       Theodore Rossmann       2B
?A
    8       Theodore Rossmann       1C
    9       Marion Andrews          1C
4 substitutions
```

example 5:

Replaces **PICTURE** with **PIC** on lines 21 through 23. Only the number of substitutions made appears, not the altered lines. Uses the TYPE command to verify the change.

```
   21       00120   01  STUDENT-CARD
   22       00121       05  STUDENT-NAME      PICTURE IS A(20)
   23       00122       05  SOC-SEC-NO        PICTURE IS 9(9)

*SUBSTITUTE+PICTURE+PIC+ 21 THRU 23 /NOTYPE
2 substitutions
*TYPE 22 THRU 23

   22       00121       05  STUDENT-NAME      PIC IS A(20)
   23       00122       05  SOC-SEC-NO        PIC IS 9(9)
```

example 6:

Replaces the string **125 State** with the string **1001 Main**. Displays only the first 20 characters on the line after the substitution is made.

```
    15          125 State Street, North Adams, Massachusetts

*SUBSTITUTE'125 State'1001 Main' /BRIEF:20
    15          1001 Main Street, No
1 substitution
```

related commands

- Keypad—SUBS
- Nokeypad—S (substitute)

SUBSTITUTE NEXT Command—Line

SYNTAX

[SUBSTITUTE] NEXT*[/[string-1]/string-2/]*

DESCRIPTION

The SUBSTITUTE NEXT command causes EDT to search for the next occurrence of string-1 and replace it with string-2. The SUBSTITUTE NEXT command takes neither a buffer specifier nor a range specifier. It can perform only one substitution at a time. Hence EDT never prints the number of substitutions for the command. If EDT cannot find string-1, it stops at the *end* of the buffer. No message is printed.

Since the word SUBSTITUTE is optional in the SUBSTITUTE NEXT command, this command is sometimes referred to simply as the NEXT command. If you do include SUBSTITUTE or its abbreviation—S—in your command line, be sure to put a space before the word NEXT or its abbreviation—N.

The string-1 and string-2 specifiers must be surrounded by delimiters. You can use any nonalphanumeric character as a delimiter (except the percent sign (%) and the underscore (_)), provided that the character does not appear in either string-1 or string-2. The three delimiters must be identical.

The string-1 or string-2 specifier can sometimes be omitted from the SUBSTITUTE NEXT command. If you omit string-1, EDT always uses the current search string. If you omit string-2, but supply string-1, EDT deletes the search string from the text and replaces it with nothing. If you supply neither string-1 nor string-2 *and* do not include the delimiters, EDT uses the current search string for string-1 and the current substitute string for string-2. If you omit both strings, but include the delimiters, EDT displays an error message.

The following samples show what happens when you omit one or both string specifiers:

```
This is file A.     SUBSTITUTE NEXT/file/buffer/     This is buffer A.
This is file B.     SUBSTITUTE NEXT///               Search string cannot be null
This is file C.     SUBSTITUTE NEXT/file//           This is  C.
This is file D.     SUBSTITUTE NEXT//buffer/         This is buffer D.
This is file E.     SUBSTITUTE NEXT                  This is buffer E.
```

EXAMPLES

example 1: Moves to line 6 to replace **formula** with **FORmula**.

```
    5       FORTRAN gets its name from the two words
    6       formula and translation.

*SUBSTITUTE NEXT/formula/FORmula/

    6       FORmula and translation.
```

example 2:

Uses SUBSTITUTE to replace **84** with **85** on line 12. Then uses SUBSTITUTE NEXT without any strings to make the substitution on line 13.

```
   12       January 1, 1985
   13       May 30, 1985

*SUBSTITUTE/84/85/ 12
   12       January 1, 1985
1 substitution
*SUBSTITUTE NEXT
   13       May 30, 1985
```

example 3:

First replaces **the VT52** with **the VT100** on line 59. Then using the same search string, replaces it with **a VT100**.

```
   58       EDT is optimized for use on a high-speed
   59       terminal with a keypad, such as the VT52.
   60       You can use the VT52 keypad both as a
   61       numerical keypad and as an editing keypad.

*SUBSTITUTE NEXT/the VT52/the VT100/
   59       terminal with a keypad, such as the VT100.
*SUBSTITUTE NEXT//a VT100/
   60       You can use a VT100 keypad both as a
```

example 4:

Replaces the string **input-output** with the string **I/O**. Uses the colon as the delimiter character.

```
   37       Computer terminals are input-output devices.

*SUBSTITUTE NEXT:input-output:I/O:
   37       Computer terminals are I/O devices.
```

related commands

- Keypad—SUBS
- Nokeypad—SN (substitute next)

TAB ADJUST Command—Line

SYNTAX

TAB ADJUST *[-]n [=buffer] [range]*

DESCRIPTION

The TAB ADJUST command enables you to indent whole lines to format outlines, indent computer programs, or create other types of layered text. In order for the command to work, you must establish a **SET TAB** value for your editing session. (The default is SET NOTAB.) The TAB ADJUST command cannot be used to arrange text into columns.

The **n** specifier is the multiple that EDT applies to the SET TAB value to determine how far to indent the line(s). For example, if your SET TAB value is 5, you must use an **n** specifier of 3 to indent a line 15 columns. The lines of text are always moved to the right unless you use a minus sign before the **n** specifier. The minus sign enables you to move indented text back toward the left margin. Notice that the **n** specifier is *not* optional. You must include a value of 1 to indent the text one tab stop.

When you specify a range, EDT indents every line specified by the range. If you include a buffer specifier with no range, the entire buffer is indented. When you omit both the buffer and range specifiers, EDT assumes that you have an active select range. Use the keypad **SELECT** function or the nokeypad **SEL** command to establish a select range. The select range can contain only whole lines.

To find out the current SET TAB value, use the **SHOW TAB** command. The level count displayed by the SHOW TAB command is not affected by the TAB ADJUST **n** specifier. The level count is only affected by the tab compute, tab decrement, and tab increment functions in the keypad and nokeypad modes.

To see the results of the TAB ADJUST command as you are working on the text, use the **SET NONUMBERS** command. With SET NUMBERS in effect, the EDT line numbers interfere with the indentation display.

EXAMPLE

First establishes a SET TAB value of 3 and issues the SET NONUMBERS command. Then indents lines 2 through 5 so that each line moves one tab stop farther over than the line above. Finally moves lines 4 and 5 to the left to change the format.

```
1       This is the first line.
2       This is the second line.
3       This is the third line.
4       This is the fourth line.
5       This is the fifth line.

*SET TAB 3
*SET NONUMBERS
*TAB ADJUST 1 2
*TAB ADJUST 2 3
*TAB ADJUST 3 4
*TAB ADJUST 4 5
*TYPE WHOLE
```

```
This is the first line.
   This is the second line.
      This is the third line.
         This is the fourth line.
            This is the fifth line.

*TAB ADJUST -2 4
*TAB ADJUST -4 5
*TYPE WHOLE

This is the first line.
   This is the second line.
      This is the third line.
   This is the fourth line.
This is the fifth line.
```

related commands

- Keypad—CTRL/T
- Nokeypad—TADJ (tab adjust)

TYPE Command—Line

SYNTAX

TYPE *[=buffer] [range] [/***BRIEF***[:n]] [/***STAY***]*

DESCRIPTION

The TYPE command is used to display lines of text at your terminal. If you omit both the buffer and range specifiers, EDT prints the current line. You can use the range specifier to print one or more lines anywhere in the current or specified buffer. If you use the buffer specifier without a range specifier, EDT moves to the named buffer and prints all its lines. (Use CTRL/C if you want to stop EDT from printing all the lines in a long buffer.) When you include the buffer specifier, EDT moves to the named buffer and remains there unless you use the /STAY qualifier.

The **/BRIEF** qualifier signals EDT to display only part of each line. If you omit the **n** specifier, EDT prints the first 10 characters (the default). Use the **n** specifier to have EDT print more or less than 10 characters per line.

The **/STAY** qualifier instructs EDT to leave its pointer at the current line rather than relocate it at the first line of the buffer or range being displayed. Even if the lines being printed are in another buffer, the /STAY qualifier maintains EDT's position as it was before you issued the TYPE command. Thus, you can display the contents of another buffer without having EDT transfer you to that buffer.

The line mode <**null**> command is similar to TYPE, but it has no command word and cannot take any qualifiers.

EXAMPLES

example 1:

Displays the current line.

```
    5       All classes begin at 8:30 a.m., unless otherwise
    6       noted.  Please be prompt.  Course material has
    7       been designed so that it fills the entire scheduled
    8       time.  Breakfast will be provided, so no need to
    9       stop for a cup of coffee on the way to class.

*TYPE

    5       All classes begin at 8:30 a.m., unless otherwise
```

example 2:

Using the same text as in Example 1, displays lines 6 through 8.

```
*TYPE 6 THRU 8

    6       noted.  Please be prompt.  Course material has
    7       been designed so that it fills the entire scheduled
    8       time.  Breakfast will be provided, so no need to
```

example 3:

Using the same text as in Example 1, displays only the first five characters of line 9.

```
*TYPE 9 /BRIEF:5

    9       stop
```

example 4:

Using the same text as in Example 1, displays line 5, but the current line does not change.

```
*TYPE 5 /STAY

    5       All classes begin at 8:30 a.m., unless otherwise

*TYPE

    9       stop for a cup of coffee on the way to class.
```

example 5:

Moves to the PASTE buffer and displays every line.

```
*TYPE =PASTE

    1       for the meeting.  When you are ready to put
    2       together an agenda, contact the Sales Manager
 [EOB]
```

example 6:

Returns to the MAIN buffer and displays lines 5 through 9.

```
*TYPE =MAIN 5 THRU 9

    5       All classes begin at 8:30 a.m., unless otherwise
    6       noted.  Please be prompt.  Course material has
    7       been designed so that it fills the entire scheduled
    8       time.  Breakfast will be provided, so no need to
    9       stop for a cup of coffee on the way to class.
```

related commands

— Nokeypad—"move"

WRITE Command—Line

SYNTAX

WRITE *file-specification [=buffer] [range] [/***SEQUENCE***[:initial[:increment]]]*

DESCRIPTION

The WRITE command copies text from an EDT buffer to an external file. The WRITE command has no effect on the buffer you are editing or the contents of any buffer in your EDT session. When you omit both the buffer and range specifiers from your WRITE command, EDT copies the entire contents of the current buffer to the external file. If you use the buffer specifier with no range, EDT copies the entire contents of the named buffer. When you issue a WRITE command with a buffer specifier, EDT does *not* move to the named buffer. You are still in the same buffer from which you issued the WRITE command.

The **/SEQUENCE** qualifier instructs EDT to put sequence numbers in the external file for use in subsequent editing sessions. The sequence numbers used with the WRITE command are not part of the text of the external file (as is the case with the **PRINT** command). However, these sequence numbers become EDT line numbers when the file is edited with EDT. (The sequence numbers are ignored if you use the **INCLUDE** command to add the file to an EDT session already in progress.) Use the **RESEQUENCE** command to renumber your text when you are working in EDT.

The WRITE command assigns EDT's default file attributes to the file it creates.

EXAMPLES

1
```
*WRITE DATA.DAT
```
Copies the entire current buffer to the file DATA.DAT.

2
```
*WRITE LETTER1.RNO 9 THRU 87
```
Copies lines 9 through 87 in the current buffer to the file LETTER1.RNO.

3
```
*WRITE MEMO11125.MEM =REVED11125
```
Copies the entire buffer REVED11125 to the file MEMO11125.RNO.

4
```
*WRITE YESNO.BAS =YESNOFIXD 123 THRU END
```
Copies lines 123 through the end in the buffer YESNOFIXD to the file YESNO.BAS.

5
```
*WRITE OVER.DAT /SEQUENCE:10:5
```
Copies the entire current buffer to the file OVER.DAT. Assigns sequence numbers to the external file. The numbers begin with 10 and increment by 5.

related commands

None.

ADV (Advance) Command—Nokeypad

SYNTAX ADV

DESCRIPTION

The ADV (advance) command sets the direction for subsequent editing work to forward (to the right of the cursor and down toward the end of the buffer).

When ADV is in effect, you can move backward for a single command by preceding that command with a minus sign (–).

ADV can be used with a search string to ensure that the direction of the search is forward. In this case, ADV follows the quoted string ("New Jersey" ADV). The current direction is now forward.

If a keypad key whose definition starts with the ADV command is used to process a prompt, the function of that key is also performed. For example, if you have defined a key to be **ADV BR.**, when you press that key to send the search string to EDT in response to the **Search for:** prompt, EDT not only processes the search, but also the rest of the ADV BR. definition, thus moving the cursor to the top of the buffer.

ADV is the default and remains in effect until you give the **BACK** command.

EXAMPLE

Sets the direction to forward. Moves the cursor to the period in **Md.** and deletes the period. Uses the minus sign to reverse direction for the CHGCC command.

```
Mr. John H. Jones
Digital Equipment Corporation
6707 Whitestone Road
Baltimore, Md.  21207

ADV
"Md."
2C
DC
-CHGCC

Mr. John H. Jones
Digital Equipment Corporation
6707 Whitestone Road
Baltimore, MD  21207
```

related commands

— Keypad—ADVANCE

APPEND Command—Nokeypad

SYNTAX

[+|−][count] **APPEND***[+|−][count]entity[=buffer]*

DESCRIPTION

The APPEND command deletes the specified entity from the current buffer and adds it to the end of either the PASTE buffer (the default) or the specified buffer. The previous contents of the PASTE or specified buffer remain in that buffer; nothing is deleted. The appended entity is inserted below the text that is already in the buffer.

EXAMPLE

Alphabetizes the list of names using CUT, APPEND, and PASTE.

```
Joe Spitzer
Trudi Schutz
Becky Santerre
Kathy Waldbauer
```

```
CUTL
```

(Move the cursor to the **T** in **Trudi**.)

```
APPENDL
```

(Move the cursor to the **J** in **Joe**.)

```
PASTE
```

```
Becky Santerre
Trudi Schutz
Joe Spitzer
Kathy Waldbauer
```

related commands

— Keypad—APPEND

ASC (ASCII) Command—Nokeypad

SYNTAX

[number] **ASC**

DESCRIPTION

The ASC (ASCII) command causes EDT to insert a special character into the text. The inserted character corresponds to the decimal equivalent **number** specified with the ASC command. EDT inserts the character to the left of the cursor. You must use a separate ASC command for each different character you want to insert.

If you do not supply the number specifier with the ASC command, EDT inserts the null character—^@ (decimal value 00). The number specifier can range from 0 to 255.

In order to use a count specifier with the ASC command, you must enclose the decimal number and the ASC command word in parentheses, for example, 5(12ASC).

The ASC command is most useful when you need to add control and other "nonprinting" characters to your text. You can use the **circumflex** command (^) to insert ASCII control characters (decimal value 0 through 31) in both text and EDT command lines.

EXAMPLE

Inserts three bell control characters (CTRL/G) at the end of a line.

```
Today is Halloween!_
3(7ASC)
Today is Halloween!^G^G^G_
```

related commands

— Keypad—SPECINS

BACK (Backup) Command—Nokeypad

SYNTAX

BACK

DESCRIPTION

The BACK command sets the direction for subsequent editing work to backward (to the left of the cursor and up toward the beginning of the buffer).

After giving the BACK command, you can move forward for a single command by preceding that command with a plus sign (+).

You can use BACK with a search string to set the direction of the search to backward. In this case, BACK follows the quoted string ("New Jersey" BACK). The current direction is now backward.

If a keypad key whose definition starts with the BACK command is used to process a prompt, the function of that key is also performed. For example, if you have defined a key to be **BACK BR.**, when you press that key to send the search string to EDT in response to the **Search for:** prompt, EDT not only processes the search, but also the rest of the BACK BR. definition, thus moving the cursor to the top of the buffer.

To change EDT's direction to forward, use the **ADV** command.

EXAMPLE

Sets the direction to backward for the "move" command. The plus sign reverses direction for the CHGUW command only.

```
Mr. Edward Johnson
Technical Writer
Digital Equipment Corporation
Nashua, Nh  03061

BACK
2W
+CHGUW

Mr. Edward Johnson
Technical Writer
Digital Equipment Corporation
Nashua, NH  03061
```

related commands

— Keypad—BACKUP

BELL Command—Nokeypad

SYNTAX

BELL

DESCRIPTION

The BELL command is designed primarily for use in keypad key definitions. It causes the terminal bell (buzzer) to sound when the command is processed.

EXAMPLE

Puts the BELL command in a keypad key definition.

```
*DEFINE KEY GOLD L AS "2500DL BELL."
```

The key sequence GOLD/L is defined to delete 2500 lines. When EDT finishes deleting the lines, it sounds the terminal bell, letting you know that the work is complete.

related commands

None.

buffer specifier—Nokeypad

SYNTAX

=buffer name

DESCRIPTION

The buffer specifier consists of two elements, the signal to EDT that the following characters are a buffer name and the buffer name itself.

The specifier **buffer** refers to the name of a location used by EDT to store text during an editing session. When you begin your EDT session, a copy of your file is put in the buffer called MAIN. EDT also creates a buffer called PASTE for use with the **CUT**, **PASTE**, and **APPEND** commands. When you exit from EDT using the **EXIT** command, a copy of the MAIN buffer text is put in an external file. All other buffers created during your editing session disappear. If you use **QUIT** to leave EDT, no copy of the MAIN buffer is made.

You create additional buffers during an EDT session each time you specify a new buffer name in a line or nokeypad command. For example, the command **CUT=LIST** creates a buffer called LIST.

Buffer names can have more than 80 alphanumeric characters. Only letters, digits, and the underscore character (_) can be used to create buffer names. A letter must always be the first character in the name.

You can edit any buffer you create, as well as the MAIN and PASTE buffers. Text can be copied or moved from one buffer to another. The **SHOW BUFFER** command lists all the accessible buffers in your editing session and indicates the current buffer by preceding the name with an equal sign (=).

EDT has several storage areas that you cannot access. These include the delete character, delete line, delete word, search, and substitute buffers. The first three are used only in keypad and nokeypad editing. These buffers contain the most recent character, line, or word deleted by the respective delete functions. The search buffer is used by the various find, substitute, and change case functions. The substitute buffer is used only by substitute commands. Storage areas also exist for entity and prompt definitions. Although you have some control over what goes into these various buffers, you cannot enter them or edit them, and their names never appear in the SHOW BUFFER list.

EXAMPLES

1 `CUT6L=SAVE`

This nokeypad command moves six lines to the buffer called SAVE. EDT remains in the current buffer.

2 `PASTE=PAR10`

This nokeypad command inserts the contents of the buffer called PAR10 to the left of the cursor.

CHGC (Change Case) Command—Nokeypad

SYNTAX

[+/−][count] **CHGC***[+/−][count]entity*

DESCRIPTION

The CHGC (change case) command causes the case of all letters within the specified entity to be changed to the opposite case. For example, if the entity is W (word), **Boston** becomes **bOSTON**.

The **CHGL** (change case lower) command only changes uppercase letters to lowercase. Conversely, the **CHGU** (change case upper) command only changes lowercase letters to uppercase.

When CHGC is used with the SR (select range) entity and there is no active select range, the case of one or more letters will be changed, depending on the cursor location. See the chart under the description of the keypad **CHNGCASE** function for more information.

EXAMPLE

Changes the case of the **M** in **Maynard** and then changes the case of the entire word so that Maynard has all uppercase letters.

```
Maynard, MA
CHGCC
```

(Move the cursor back to the **m** in **maynard**.)

```
CHGCW
MAYNARD, MA
```

related commands

— Keypad—CHNGCASE

CHGL (Change Case Lower) Command—Nokeypad

SYNTAX

[+/−][count] **CHGL***[+/−][count]entity*

DESCRIPTION

The CHGL (change case lower) command changes all uppercase letters within the specified entity to be lowercase. Letters that are already lowercase remain unchanged.

To change lowercase letters to uppercase, use **CHGU** (change case upper). To change all letters to their opposite case use **CHGC** (change case).

When CHGL is used with the SR (select range) entity and there is no active select range, the case of one or more letters might be changed, depending on the cursor location. See the chart under the description of the keypad **CHNGCASE** function for more information. Remember, however, that for CHGL, only uppercase letters are changed to lowercase; lowercase letters remain the same.

EXAMPLE

Changes all uppercase letters to lowercase for all three words in the line.

```
Word Processing/Small Systems
CHGL3W
word processing/small systems_
```

related commands

— Keypad—CHNGCASE

CHGU (Change Case Upper) Command—Nokeypad

SYNTAX

[+|−][count] **CHGU** *[+|−][count]entity*

DESCRIPTION

The CHGU (change case upper) command changes all lowercase letters to uppercase within the entity. Letters that are already uppercase remain unchanged.

CHGL (change case lower) changes all uppercase letters to lowercase. **CHGC** (change case) changes all lowercase letters to uppercase and all uppercase letters to lowercase.

When CHGU is used with the SR (select range) entity and there is no active select range, the case of one or more letters will be changed, depending on the cursor location. See the chart under the description of the keypad **CHNGCASE** function for more information. Remember, however, that for CHGU, only lowercase letters are changed to uppercase; uppercase letters remain the same.

EXAMPLE

Changes all lowercase letters to uppercase for the entire line.

```
The United States of America
CHGUL
THE UNITED STATES OF AMERICA
```

related commands

— Keypad—CHNGCASE

^(Circumflex) Command—Nokeypad

SYNTAX

[count]^character

DESCRIPTION

The circumflex (^) command enables you to insert ASCII control characters (decimal value 0 through 31) in your text or enter them in nokeypad command lines. When you type the circumflex first and then a character from the keyboard, EDT interprets the two symbols as one control character.

The ASCII control characters are formed from the combination of the control key with the 26 letters and the following nonalphanumeric characters: the at sign (@), the backslash (\), square brackets ([ ]) the circumflex (^), and the underscore (_).

You can use the **ASC** command to enter any character from the DEC Multinational Character Set list by using the decimal equivalent values 0 through 255. The circumflex works only for characters with decimal values 0 through 31.

EXAMPLE

Uses the main keyboard characters circumflex (^) and Z to complete the single line form of the nokeypad I (insert) command.

```
Mail to Digital Equipment Corporation_
I, Merrimack, NH  03054^Z
Mail to Digital Equipment Corporation, Merrimack, NH  03054_
```

related commands

— Keypad—SPECINS

CLSS (Clear Search String) Command—Nokeypad

SYNTAX CLSS

DESCRIPTION

The CLSS (clear search string) command clears the search string that is currently in the search buffer. After you issue the CLSS command, the search buffer is empty. This command can be used to redefine keypad keys that include SR (select range) in their definitions so that only the select range is affected by the function, never the search string.

When you use the SR (select range) nokeypad entity, EDT first checks to see if there is an active select range. If there is none and the cursor is located in the appropriate position of the current search string, the command that contains the SR entity affects the search string. You might find this inconvenient, particularly with the keypad **CHNGCASE** command, where you often use **FIND** to locate a word in which you want to change the case of the initial letter only. By including CLSS in the CHNGCASE definition, you can empty the search buffer before EDT processes the CHNGCASE function.

Other keypad definitions that include the SR entity are **APPEND**, **CUT**, and **FILL**.

EXAMPLES

example 1:

Moves to the string **wednesday**. Then clears the search buffer before issuing the CHGCSR command to change the case of the letter **w**.

```
The meeting will take place on wednesday afternoon at 2:30.

"wednesday" CLSS CHGCSR

The meeting will take place on Wednesday afternoon at 2:30.
```

example 2:

This example takes place in keypad mode. You first redefine the CHNGCASE function to include the CLSS command ahead of the CHGCSR definition. After finding the string **albu**, EDT changes the case of only the first letter because there is no current search string.

```
Before leaving for the airport, be sure to contact Jim
In albuquerque.  If he has not made arrangements for the
```

GOLD + PAGE COMMAND

```
Command:  DEFINE KEY GOLD 1 AS "CLSS CHGCSR."
```

ENTER SUBS

GOLD + FNDNXT (FIND)

```
Search for:  albu
```

ENTER (SUBS)

```
Before leaving for the airport, be sure to contact Jim
in albuquerque.  If he has not made arrangements for the
```

GOLD + WORD (CHNGCASE)

```
Before leaving for the airport, be sure to contact Jim
in Albuquerque.  If he has not made arrangements for the
```

---

related commands

None.

CTRL/C (Control C) Function—Nokeypad

KEY SEQUENCE

CTRL/C

DESCRIPTION

CTRL/C interrupts certain operations before EDT finishes processing them. You can use CTRL/C to stop a runaway search through a long file or to stop a long repeat count. CTRL/C halts certain EDT operations. For example, you can use CTRL/C to stop EDT from printing a whole buffer when you have used the line mode **TYPE** command to move to another buffer.

When EDT aborts the operation, it prints the message **Aborted by CTRL/C**. If EDT cannot stop a particular process, it prints the message **CTRL/C ignored**.

EXAMPLE

Stops EDT from printing the entire contents of the buffer DARCY_LET.

```
EXT SHOW BUFFER
 DARCY_LET      40      lines
=MAIN   12      lines
 PASTE  No      lines
Press return to continue
EXT TYPE =DARCY_LET

    1       August 20, 1985
    2
    3       Mr. Charles R. Darcy
    4       Production Manager
    5       Midland Manufacturing Corporation
    6       East St. Louis, IL  62202

CTRL/C

Aborted by CTRL/C
```

related commands

— Keypad—CTRL/C

— Line—CTRL/C

CTRL/Z (Control Z) Function—Nokeypad

KEY SEQUENCE

CTRL/Z

DESCRIPTION

CTRL/Z performs the same function in nokeypad editing as it does in line editing. (It has a different function in keypad editing.) You use CTRL/Z to exit from the insert state. After you have given the appropriate insert command and typed the text you want to add, use CTRL/Z to complete the insert procedure. CTRL/Z is used with both the **INSERT** and **REPLACE** commands in line mode and the **I** (insert) and **R** (replace) commands in nokeypad mode. In nokeypad mode, typing the **circumflex** and then the letter Z (_^Z) has the same effect as pressing CTRL/Z.

EXAMPLE

Completes the insert operation and returns EDT to the command state.

```
Customer called Repair Service 5/25/85.
One of his telephones was not working.
He was told to disconnect his phone and
take it to his nearest phone store.
```

```
I RET
```

(Now type in the line **REPAIR SERVICE REPORT #825** where the cursor is and press RETURN to move the **Customer called** text to the next line.)

```
CTRL/Z
```

```
REPAIR SERVICE REPORT #825
Customer called Repair Service 5/25/85.
One of his telephones was not working.
He was told to disconnect his phone and
take it to his nearest phone store.
```

related commands

— Line—CTRL/Z

count specifier—Nokeypad

SYNTAX

digit(s)

DESCRIPTION

The count specifier indicates either the number of times a nokeypad command is to be repeated or the number of entities that the command will affect. The maximum value for the count specifier is 32767.

EXAMPLES

1 `5W`

Moves the cursor five words.

2 `CHGC27C`

Changes e case of the next 27 characters.

3 `8UNDC`

Inserts the contents of the delete character buffer eight times.

4 `D4SEN`

Deletes the next four sentences.

5 `-3SN`

Performs the same substitution three times, but toward the beginning of the buffer.

6 `5(+V D+W)`

Deletes the first word in each of the next five lines. The parentheses indicate that the repeat count applies to both commands—+V and D+W. If you omit the parentheses, EDT deletes only the first word on the fifth line.

7 `2(7ASC)`

Inserts two bell control characters (CTRL/G) into your text. The parentheses are necessary to distinguish the count number from the decimal equivalent value of CTRL/G.

CUT Command—Nokeypad

SYNTAX

[+|−][count] **CUT** *[+|−][count]entity[=buffer]*

DESCRIPTION

The CUT command removes the specified entity from the text buffer and stores it in the PASTE buffer or the specified buffer. EDT uses the PASTE buffer when you do not specify a buffer name with the CUT command.

You can use the CUT command to delete large or small sections of text. When you use CUT in conjunction with the **PASTE** command, you can move or copy text from one place to another in the same buffer, or from one EDT buffer to another, but not to or from external files. The line mode commands **INCLUDE** and **WRITE** are used with external files.

When you use CUT to put part of a line into a buffer, EDT adds a line terminator at the end of the line fragment. The line terminator is necessary because EDT cannot store partial lines in a buffer. When you use the PASTE command to insert that text, EDT removes the extra line terminator.

The steps for moving or copying text are described under the nokeypad PASTE command.

EXAMPLE

Moves the first line to the TJ buffer and then uses the PASTE command to insert that line in the proper chronological sequence between Adams and Madison.

```
Thomas Jefferson
George Washington
John Adams
James Madison

CUTL=TJ
```

(Move cursor to the **J** in **James Madison**.)

```
PASTE=TJ

George Washington
John Adams
Thomas Jefferson
James Madison
```

related commands

— Keypad—CUT

D (Delete) Command—Nokeypad

SYNTAX

[+/−][count] **D***[+/−][count]entity*

DESCRIPTION

The D (delete) command removes the specified entity from the current buffer. If the entity involves a character, word, or line, the deleted text is stored in the delete character, delete word, or delete line buffer. EDT overwrites the contents of the appropriate buffer each time you use a D command that involves a character, word, or line entity. You can use the **UNDC**, **UNDW**, and **UNDL** commands to insert the current contents of the respective delete buffers anywhere in the current buffer.

You can use the count specifiers to process several deletions in one command. For example, D3W deletes three words; 2DL deletes two lines. The sign specifier enables you to delete entities to the left of the cursor without changing EDT's direction. For example, –2DC deletes the two characters to the left of the cursor.

EXAMPLE

Deletes the first line. Then deletes the third line and uses UNDL to insert that line in chronological order. Deletes the last character in **Todd** and the word at the end of the last line.

```
Coming Events:
January 18, 1985      Luncheon Seminar on Technical Communications
January 30, 1985      Meeting of the EDT Development Staff
January 23, 1985      Seminar on Writing for Programmers
February 2, 1985      Lecture by Todd Brockman
February 7, 1985      Meeting of the Secretarial Staff Group

DL
```

(Move the cursor to the January 23 line.)

```
DL
```

(Move the cursor up to the January 30 line.)

```
UNDL

January 23, 1985      Seminar on Writing for Programmers
January 30, 1985      Meeting of the EDT Development Staff
```

(Move the cursor to the second **d** in Todd.)

```
DC

February 2, 1985      Lecture by Tod_Brockman
```

(Move the cursor to the **G** in Group.)

```
DW

Coming Events:
January 23, 1985      Seminar on Writing for Programmers
January 30, 1985      Meeting of the EDT Development Staff
February 2, 1985      Lecture by Tod Brockman
February 7, 1985      Meeting of the Secretarial Staff_
```

related commands

- Keypad—DEL C, DEL EOL, DELETE, DEL L, DEL W, LINEFEED, CTRL/U
- Line—DELETE

DATE Command—Nokeypad

SYNTAX

DATE

DESCRIPTION

The DATE command inserts the current date into your text. The format of the DATE text is determined by the operating system. Generally the format includes the time as well as a leading and a trailing space. The text is always placed to the left of the cursor, regardless of EDT's current direction.

EXAMPLE

Inserts the current date and time to the right of the heading.

```
DATE/TIME: _
DATE
DATE/TIME:   16-APR-1985 09:33:12 _
```

related commands

None.

DEFK (Define Key) Command—Nokeypad

SYNTAX

DEFK

DESCRIPTION

The DEFK command enables you to define a keypad function key or key sequence other than **CTRL/K** to handle the key definition process. The DEFK command does *not* allow you to create keypad definitions from nokeypad mode. In fact, DEFK has no function at all in nokeypad editing.

EXAMPLES

example 1:

Using the line mode DEFINE KEY command, defines CTRL/D to have the define key function in keypad mode.

```
*DEFINE KEY CONTROL D AS "DEFK."
```

example 2:

Using the keypad CTRL/K function, defines CTRL/D to have the define key function in keypad mode.

CTRL/K

```
Press the key you wish to define
```

CTRL + D

```
Enter the definition terminated by ENTER
DEFK.
```

ENTER
SUBS

related commands

- Keypad—CTRL/K
- Line—DEFINE KEY

DESEL (Deactivate Select) Command—Nokeypad

SYNTAX

DESEL

DESCRIPTION

DESEL (deactivate select) cancels a select range after you have used the **SEL** (select) command. If you change your mind about a select range, issue the DESEL command. You can then continue with your editing work. If you want to change a select range that is currently active, use DESEL to cancel it and then SEL to mark the beginning of the new range.

DESEL also cancels a select range set by either the **SSEL** (search and select) or **TGSEL** (toggle select) command.

EXAMPLE

Creates a select range of five lines. Then cancels that range, because the first line of the next address was included by mistake. Moves the cursor back one line and creates a four-line select range, using the minus sign (–) to reverse direction only for that command.

```
SEL5L

Albuquerque, NM  87112
Mr. Edward B. Sanders
1749 General Lee Highway
Richmond, VA  23235
Dr. Jane Marantz-Connor

–
DESEL

Albuquerque, NM  87112
Mr. Edward B. Sanders
1749 General Lee Highway
Richmond, VA  23235
Dr. Jane Marantz-Connor

–
BL SEL-4L

Albuquerque, NM  87112
Mr. Edward B. Sanders
1749 General Lee Highway
Richmond, VA  23235
Dr. Jane Marantz-Connor
```

related commands

– Keypad—RESET

DLWC (Default Lowercase) Command—Nokeypad

SYNTAX

DLWC

DESCRIPTION

The DLWC (default lowercase) command instructs EDT to change all uppercase letters to lowercase letters whenever you move the cursor during your EDT session. As the cursor moves through the buffer, every uppercase letter it passes over is changed to lowercase.

Since DLWC resets the default state, uppercase letters continue to be changed to lowercase each time you move the cursor. Use the **DMOV** (default move) command to reset EDT so that case is not affected by move operations. **DUPC** (default uppercase) causes EDT to change all lowercase letters that the cursor passes over to uppercase.

EXAMPLE

Changes EDT's default to lowercase and then shows the effect by moving the cursor over four lines of text.

```
DATE:   06-APR-85
FROM:   Linda Westbrook
DEPT:   CED
NET:    QQQQ::WESTBROOK

DLWC 4L

date:   06-apr-85
from:   linda westbrook
dept:   ced
net:    qqqq::westbrook
_
```

related commands

None.

DMOV (Default Move) Command—Nokeypad

SYNTAX

DMOV

DESCRIPTION

The DMOV (default move) command returns your editing session to EDT's default state after you have used either **DLWC** (default lowercase) or **DUPC** (default uppercase). With DLWC, EDT automatically changes the case of every uppercase letter to lowercase as the cursor moves through the text. With DUPC, all lowercase letters become uppercase as EDT moves through the buffer. For example, if you move from the current line to the end of the buffer with DUPC in effect, that entire portion of your buffer will contain only uppercase letters. DMOV sets things back to the normal state, where EDT does not alter the case of letters during move operations.

EXAMPLE

Uses DLWC to change the default to lowercase and moves four words. Then restores the default to EDT's normal state—DMOV.

```
The SOFTWARE SERVICES Training Proposal was issued last
May for the current fiscal year.

DLWC 4W

The software services training proposal was issued last
May for the current fiscal year.

DMOV
```

related commands

None.

Down Arrow—Nokeypad

KEY SEQUENCE

DESCRIPTION

The Down Arrow key moves the cursor down one line toward the bottom of the buffer, regardless of EDT's direction.

When you use the Down Arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of that line. If you continue to use Down Arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for Down Arrow and resets it the next time you use the function. Note that the Down Arrow is not a true nokeypad command.

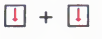
EXAMPLE

Moves the cursor from the end of the first line to the end of the last line.

```
Asuncion, Paraguay_
Bogota, Colombia
Brasilia, Brazil
Buenos Aires, Argentina
Caracas, Venezuela
Georgetown, Guyana
```

```
Asuncion, Paraguay
Bogota, Colombia_
```

```
Bogota, Colombia
Brasilia, Brazil
Buenos Aires, Argentina
```



```
Buenos Aires, Argentina
Caracas, Venezuela
Georgetown, Guyana_
```

DUPC (Default Uppercase) Command—Nokeypad

SYNTAX

DUPC

DESCRIPTION

The DUPC (default uppercase) command instructs EDT to change all lowercase letters to uppercase whenever you move the cursor during your EDT session. As the cursor moves through the buffer, every lowercase letter it passes over is changed to uppercase.

Since DUPC resets the default state, lowercase letters continue to be changed to uppercase each time you move the cursor. Use the **DMOV** (default move) command to reset EDT so that case is not affected by move operations. **DLWC** (default lowercase) causes EDT to change all uppercase letters that the cursor passes over to lowercase.

EXAMPLE

Changes EDT's default to uppercase and then shows the effect by moving the cursor over four lines of text.

```
Karen Islington
Paul Ohlmeyer
William Jacobson
Sarah Rapf

DUPC 4L

KAREN ISLINGTON
PAUL OHLMEYER
WILLIAM JACOBSON
SARAH RAPF

_
```

related commands

None.

entity specifier—Nokeypad

SYNTAX

entity

DESCRIPTION

An **entity** is a group of contiguous characters that EDT recognizes as a unit. The entity specifier is used with the following nokeypad commands:

APPEND	FILL
CHGC	"move"
CHGL	R
CHGU	SEL
CUT	TADJ
D	TGSEL

The nokeypad mode entities are:

C	character	BR	beginning of range
L	line	ER	end of range
BL	beginning of line	SR	select range
EL	end of line	SEN	sentence
NL	next line	BSEN	beginning of sentence
PAGE	page	ESEN	end of sentence
BPAGE	beginning of page	"string"	string of characters
EPAGE	end of page	V	vertical
PAR	paragraph	W	word
BPAR	beginning of paragraph	BW	beginning of word
EPAR	end of paragraph	EW	end of word

EXAMPLES

1 `CHGCW`

Changes the case of the letters in the next word.

2 `APPEND2PAR`

Deletes the next two paragraphs and places them at the end of the PASTE buffer.

3 `DBR`

Deletes to the beginning of the buffer.

4 `3CUTSEN=EXTRA`

Deletes the next three sentences and places them in the buffer named EXTRA.

EX (Exit to Line Mode) Command—Nokeypad

SYNTAX

EX

DESCRIPTION

The EX (exit to line mode) command shifts your editing session from nokeypad mode to line mode. It does *not* cause an exit from EDT.

To exit from EDT when in nokeypad mode you can use EX to shift to line mode and then give the line mode **EXIT** or **QUIT** command. Use the nokeypad **QUIT** command to exit directly from nokeypad mode, when you do not need to save your work. The nokeypad **EXT** (extend) command allows you to issue the line mode EXIT or QUIT command directly from nokeypad mode.

EXAMPLE

Shifts from nokeypad mode to line mode.

```
    .
    .
    .
Jennifer Smith      (401) 555-1111
John W. Snyder      (401) 555-2222
    .
    .
    .

EX
*WRITE DELNAMES.DAT . THRU +1
DISK$USER:[SMITH]DELNAMES.DAT  2 lines
*FIND  =MAIN
*
```

related commands

- Keypad—CTRL/Z
- Line—CHANGE

EXT (Extend) Command—Nokeypad

SYNTAX

EXT *line mode command*

DESCRIPTION

The EXT (extend) command allows you to use a line mode command while still in nokeypad mode. The text following EXT must fit on a single line.

You can issue two or more line mode commands on the same line by separating the commands with semicolons (;). If you want to put nokeypad commands after a line mode command, use **CHANGE ;nokeypad-command(s)** as the last line mode command on the EXT command line. You can use an EDT macro with the EXT command just as you would any other line mode command.

EXAMPLES

example 1:

Enables you to enter several line mode SET commands on a single line directly from nokeypad mode.

```
EXT SET SEARCH EXACT ;SET WORD NODELIMITER ;SET SCREEN 100
```

example 2:

Enables you to use the line mode EXIT command directly from nokeypad mode.

```
EXT EXIT

DISK$USER:[SMITH:MEMOS]JAN1.RNO;6  34 lines
```

example 3:

Uses two line mode commands followed by a nokeypad command to reposition the cursor after the line mode commands are processed.

```
EXT COPY =SAVE 4 THRU 27 TO =TEST ;SUBSTITUTE/10/5/ WHOLE; CHANGE ;5L 3C
```

related commands

— Keypad—COMMAND

FILL Command—Nokeypad

SYNTAX

[+/−][count] **FILL***[+/−][count]entity*

DESCRIPTION

The FILL command takes a select range of lines and reorganizes the text so that the maximum number of whole words can fit within the current line width. The default line width for EDT is the terminal width that the operating system passes to EDT. Use the line mode **SHOW SCREEN** command to find out the current screen/line width. The valid screen width values for screen mode editing are 80 and 132. (132 is valid only for VT100-type terminals with AVO—advanced video option.) If your screen width is set to 80, EDT will fill lines to column 79; if your screen width is 132, EDT will fill lines to column 131.

If you want to use a line length other than 80 or 132 for filling text, you must use the line mode **SET WRAP** command. The SET WRAP command also affects the line length that EDT uses for inserting text in keypad mode. EDT uses the SET SCREEN value to determine the line length for filling text only if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses the wrap value, regardless of the SET SCREEN width. You can use the **SHOW WRAP** command to find out the current wrap value or setting.

The filling process considers a blank line to be a break between paragraphs. Even if there are spaces on the blank line, EDT fills the text up to the blank line and then resumes filling with the next line that contains text.

The valid entities for the FILL command are: L (line), SEN (sentence), PAR (paragraph), PAGE, and SR (select range).

EXAMPLE

Uses the SET WRAP command to limit the line length to 60 characters. Then reformats the first six lines of text.

```
A computer is a tool that you can use to
process information and solve complicated problems.
It accepts large amounts of information,
called data,
performs various operations on that data,
and provides you with a result.

In order to process data,
there must be communication between you and the computer.

EXT SET WRAP 60
FILL6L

A computer is a tool that you can use to process information
and solve complicated problems. It accepts large amounts of
information, called data, performs various operations on
that data, and provides you with a result.

In order to process data,
there must be communication between you and the computer.
```

related commands

- Keypad—FILL
- Line—FILL

HELP Command—Nokeypad

SYNTAX

HELP

DESCRIPTION

The nokeypad HELP command is used exclusively for defining a different key or key sequence in keypad mode to carry out the keypad HELP function. **HELP.** is the preset definition of keypad **HELP** key.

The nokeypad HELP command has no effect on your nokeypad editing session. To get help information on nokeypad commands, you must use the line mode **HELP** command:

```
HELP CHANGE [subtopic [subtopic] ]
```

The **subtopic** specifier in the HELP CHANGE command line refers to a nokeypad help topic. If you do not include any subtopics, EDT prints general information about nokeypad command syntax and a list of the available subtopics. Information on nokeypad entities appears under HELP CHANGE ENTITIES. Information on nokeypad commands appears under HELP CHANGE SUBCOMMANDS.

EXAMPLES

example 1:

Defines the key sequence GOLD/H to have the HELP function in keypad mode.

```
*DEFINE KEY GOLD H AS "HELP."
```

example 2:

Shows how to get three different types of information on nokeypad mode using the EXT (extend) command and the line mode HELP command.

```
EXT HELP CHANGE SUBCOMMANDS
```

Prints general information about nokeypad commands.

```
EXT HELP CHANGE SUBCOMMANDS S
```

Prints information about the nokeypad S (substitute) command.

```
EXT HELP CHANGE ENTITIES
```

Prints information about nokeypad entities.

related commands

None.

I (Insert) Command—Nokeypad

SYNTAX

I < *RET*> *text* CTRL/Z

I*text* ^Z

DESCRIPTION

The I (insert) command shifts EDT to the insert state so that you can insert new text in the current buffer. **CTRL/Z** signals EDT to exit from the insert state.

The I command has two forms. The first can be used for inserting any amount of text from a single character or word to several paragraphs or pages. The second is limited to text that fits on a single line and does not include a line terminator.

To insert any length of text, type I and press RETURN. The I remains on the command line while the cursor returns to its former position on the screen. Now type in the text you want to insert. When you have finished entering the new text, press CTRL/Z to exit from the insert state. The new text is now part of the current buffer and the I disappears from the command line to show that you are no longer in the insert state.

To insert less than a line of text, type the I command and then follow it immediately with the text you want to insert. (EDT interprets a space after the I as part of the text to be inserted.) Then type ^Z or press CTRL/Z to signal EDT that you have finished your inserted text. Finally, press RETURN to have EDT insert the text to the left of the cursor.

You can use the DELETE key to edit the text while you are inserting it. Or, you can wait until you finish typing the new text, and then use any nokeypad editing command to make the necessary changes.

EXAMPLES

example 1:

Adds two lines of text to the current buffer.

```
Have the following materials ready:
interim report
planning budget
-
I RET
staffing projections
space requirements
CTRL/Z
```

```
Have the following materials ready:
interim report
planning budget
staffing projections
space requirements_
```

example 2:

Adds **Wednesday,[SP]** to the middle of the existing line.

```
Before the meeting on January 16th, please have the
IWednesday, ^Z
Before the meeting on Wednesday, January 16th, please have the
```

related commands

— Line—INSERT

KS (KED Substitute) Command—Nokeypad

SYNTAX

KS

DESCRIPTION

The KS (KED substitute) command is used after the nokeypad **PASTE** command to modify the position of the cursor at the completion of the PASTE command. KS can be used only directly after the PASTE command. Do not put a space between the command words PASTE and KS.

Normally after you issue a PASTE command, the cursor is located one character to the right of the inserted text, regardless of EDT's current direction. When you put KS after the PASTE command, the cursor position changes and becomes dependent on EDT's current direction. If the direction is forward, the cursor is on the last character of the inserted text. If the direction is backward, the cursor is positioned on the first character of the inserted text.

The KS command enables you to locate a search string that includes the character immediately after or before the inserted text.

The nokeypad definition for the keypad mode **SUBS** function includes the KS command. Notice the KS command following directly after the PASTE command.

```
(CUTSR=DELETE PASTEKS"").
```

EXAMPLE

Assumes that the text **Atlanta, GA,** is in the PASTE buffer. First resets the direction to backward. Then inserts the contents of the PASTE buffer and positions the cursor on the first character of the inserted text.

```
following locations:  Oakland, CA, Augusta, ME,
Billings, MT, Knoxville, TN.

BACK PASTEKS

following locations:  Oakland, CA, Augusta, ME,
Billings, MT, Atlanta, GA, Knoxville, TN.
```

related commands

None.

Left Arrow—Nokeypad

KEY SEQUENCE

DESCRIPTION

Left Arrow moves the cursor one character to the left, regardless of EDT's direction.

If the cursor is at the first character position of a line, pressing Left Arrow moves the cursor to the line terminator of the previous line.

Note that the Left Arrow is not a true nokeypad command.

EXAMPLE

Moves the cursor to the left, first to the beginning of the second line and then to the end of the first line.

```
July 1, 1985
October 1, 1985
```

```
July 1, 1985
October 1, 1985
```

```
July 1, 1985
October 1, 1985
```

related commands

— Keypad—Left Arrow

"move" Command—Nokeypad

SYNTAX

[+|–][count]entity

DESCRIPTION

The "move" command moves the cursor within the current buffer. The **entity** specifier is the only required element in the "move" command. There is no command word or abbreviation.

You can put several "move" commands together. For example, **L 3W -4C** moves the cursor down to the beginning of the next line, then three words to the right, and finally four characters to the left. In cases where you want to repeat several "move" commands, you can enclose the entities in parentheses: 8(EPAR +C).

EXAMPLES

example 1: Moves the cursor to the beginning of the next line.

```
Only a few computer programs work successfully the first
time they are tried.  Programmers are expected to make
mistakes.  The mark of a good programmer is how quickly
he or she can find and correct the errors that occur.

L

time they are tried.  Programmers are expected to make
```

example 2: Using the same text and the new cursor position, moves the cursor to the string **programmer**.

```
"programmer"

mistakes.  The mark of a good programmer is how quickly
```

example 3: Using the same text and the new cursor position, moves the cursor three words to the right.

```
3W

mistakes.  The mark of a good programmer is how quickly
```

example 4: Using the same text and the new cursor position, *moves the cursor to the end* of the last line.

```
2EL

he or she can find and correct the errors that occur._
```

related commands

- Keypad—FIND, CHAR, WORD, LINE, PAGE
- Line: FIND

PASTE Command—Nokeypad

SYNTAX

[count] **PASTE***[=buffer]*

DESCRIPTION

The PASTE command copies the contents of any buffer into the current buffer. The keypad PASTE function only copies the PASTE buffer contents into the current buffer.

When used together with the **CUT** or **APPEND** command, the nokeypad PASTE command enables you to move or copy text from one location to another in the same or different buffers. When you do not supply a buffer name with the PASTE command, EDT uses the PASTE buffer. Do not put a space between the word PASTE and the equal sign (=) when you are specifying another buffer name. Otherwise, EDT uses the PASTE buffer and tries to interpret the buffer name you specified as another command.

Note that the syntax for the PASTE command differs from the CUT command. You cannot use an entity specifier with PASTE; you can only copy entire buffers. However, remember that you can enter the PASTE buffer or another buffer and edit it before using PASTE to copy the contents into your current buffer. Use the line mode **FIND** command to move to the other buffer.

To move text, use the CUT command to delete the text from its present location. Then move the cursor to the new location and insert the text with the PASTE command. To copy text, use the CUT command to move the material you want to copy to the PASTE buffer. Then *before* moving the cursor, issue the PASTE command to restore the deleted text to its original location. Now move the cursor to the place where you want the copy inserted and issue the PASTE command again.

You can use the APPEND command to add text to the contents of the buffer you want to insert later with the PASTE command. When you move or copy text with CUT and PASTE or CUT, APPEND, and PASTE, be sure to use the same buffer name for all the commands. You can use the line mode **SHOW BUFFER** command to list all the buffers that currently exist in your editing session.

Each time you issue the CUT command, EDT overwrites the contents of the PASTE buffer or the specified buffer. When you use CUT to put part of a line into a buffer, EDT adds a line terminator at the end of the text since EDT buffers cannot contain partial lines. The PASTE command removes the added line terminator so that when you insert the text, you do not have an unwanted line break.

For information on how to modify the cursor position after EDT processes a PASTE command, see the nokeypad **KS** command.

EXAMPLES

example 1:

Inserts the text from the buffer PAGEBLANK at the current cursor location.

```
in the following three chapters.
<FF>

_
PASTE=PAGEBLANK

in the following three chapters.
<FF>
[This page intentionally left blank.]
<FF>

_
```

example 2:

After using CUT to delete the last line, inserts that line in the middle of the list.

```
George G. Haraway
John W. Hardwick
Mason G. Harbaugh
_
CUTL
```

(Move the cursor to the **J** in **John**.)

```
PASTE

George G. Haraway
Mason G. Harbaugh
John W. Hardwick
_
```

example 3:

Uses CUT, APPEND, and PASTE to alphabetize a list of cities.

```
Dallas
Houston
Fort Worth
Austin
San Antonio
CUTL
```

(Move the cursor to the **D** in **Dallas**.)

```
APPENDL
```

(Move the cursor to the **F** in **Fort Worth**.)

```
APPENDL
```

(Move the cursor to the **H** in **Houston**.)

```
PASTE

Austin
Dallas
Fort Worth
Houston
San Antonio
```

related commands

- Keypad—PASTE
- Line—COPY, MOVE

QUIT Command—Nokeypad

SYNTAX

QUIT

DESCRIPTION

The QUIT command ends your nokeypad editing session without saving your edits and without passing through line mode. You are no longer in EDT, no new file is saved in any directory, and your terminal is now at the system command level.

The nokeypad QUIT command cannot take the /SAVE qualifier that the line mode **QUIT** command has. The nokeypad QUIT command n only end your editing session.

EXAMPLE

Ends the nokeypad editing session without copying the MAIN buffer contents to an external file. Only the system prompt appears.

```
QUIT
$
```

related commands

— Line—QUIT

R (Replace) Command—Nokeypad

SYNTAX

[+/−][count] **R***[+/−][count]entity*

DESCRIPTION

The R (replace) command combines the delete and insert functions. First EDT deletes the specified text; then it shifts to the insert state so that you can enter text at your terminal. When you finish entering the new text, press **CTRL/Z** to exit from the insert state. While you are inserting the new text, EDT continues to display the R command at the bottom of the screen. The entity specifier always refers to the text you want to delete.

EXAMPLE

Deletes the last line and replaces it with new text.

```
This seminar will cover the following topics:
     Management by Objective
     Management by Design
     Management by Intimidation

RL
```

(Now type in the new line, **Management by Consensus**, and complete the insert by pressing CTRL/Z.)

```
This seminar will cover the following topics:
     Management by Objective
     Management by Design
     Management by Consensus
```

related commands

- Keypad—REPLACE
- Line—REPLACE

REF (Refresh) Command—Nokeypad

SYNTAX

REF

DESCRIPTION

The REF command clears and redraws the entire screen. Use REF to eliminate any extraneous characters or messages that have appeared on the screen but are not part of the current text you are editing. After the REF command is given, EDT discards any characters that are not part of the current text. The screen is restored to its state prior to the interruption. REF has no effect on the text you are editing.

EXAMPLE

Refreshes the screen to eliminate the notification of new mail.

```
and expect to hear from you in the near future.
>NEW MAIL FROM YYYYYY::WHITEif you have any questions or
problems.

REF

and expect to hear from you in the near future.
In the meantime, please call if you have any questions or
problems.
```

related commands

- Keypad—CTRL/R, CTRL/W
- Line—CTRL/R

Right Arrow—Nokeypad

KEY SEQUENCE

→

DESCRIPTION

Right Arrow moves the cursor one character to the right, regardless of EDT's direction.

If the cursor is on a line terminator, Right Arrow moves the cursor to the first character on the next line.

Note that Right Arrow is not a true nokeypad command.

EXAMPLE

Moves the cursor to the right to the end of first line. Then moves the cursor to the beginning of the second line.

```
January 1, 1985
April 1, 1985
```

→ + → + →

```
January 1, 1985
April 1, 1985
```

→

```
January 1, 1985
April 1, 1985
```

related commands

— Keypad—Right Arrow

S (Substitute) Command—Nokeypad

SYNTAX

[+|–][count] **S***/[string-1]/string-2/*

DESCRIPTION

The S (substitute) command looks for the next occurrence of string-1, deletes it, and replaces it with string-2.

The count specifier instructs EDT to perform the specified number of searches and substitutions. If the count exceeds the total number of occurrences of string-1 between the current cursor position and either the beginning or end of the buffer (depending on the direction specified), EDT performs as many substitutions as it can, and then prints the message **String was not found** to indicate that it could not find another string-1. The cursor is now located just after the last substitution that was made.

The sign specifier enables you to change the direction of the search for string-1, for that S command only, without affecting EDT's direction.

The slashes are string delimiters. Any nonalphanumeric character can be used for a string delimiter as long as that character is not contained in either string-1 or string-2. All three delimiters must be identical.

String-1 or string-2 can sometimes be omitted from the S command. If you omit string-1, EDT always uses the current search string. If the search buffer is empty, EDT simply inserts string-2 into the text to the left of the cursor. If you omit string-2, EDT always deletes string-1 and inserts nothing. The following samples show what happens when you omit string-1, string-2, or both:

```
This is file A.     S/file/buffer/     This is buffer A.
This is file B.     S///               This is  B.
This is file C.     S/file//           This is  C.
This is file D.     S//buffer/         This is buffer D.
This is file E.     S                  This is  E.
```

The **SN** command, which does not take any strings as specifiers, always uses the current contents of the search buffer and the substitute buffer.

EDT performs searches in several ways. For more information see the **SET SEARCH** command. **SHOW SEARCH** tells which search parameters are currently in effect.

EXAMPLES

example 1: Changes **1987** to **1988**.

```
January 3, 1987
S/7/8/
January 3, 1988
```

example 2:

Changes **pages** to **Sections** in two places.

```
in pages 17 through 25.  Then in pages 32 through 40 we
2S/pages/Sections/
in Sections 17 through 25.  Then in Sections 32 through 40 we
```

example 3:

With SET SEARCH EXACT, moves back to the first line to change **users** to **you**, skipping over **Users** on line 2.

```
If users are able to access the line printer, then the DCL
PRINT command can be given.  Users who only have access to
their terminals are restricted to the DCL TYPE command.
EXT SET SEARCH EXACT
-S/users/you/
If you are able to access the line printer, then the DCL
PRINT command can be used.  Users who only have access to
their terminals are restricted to the DCL TYPE command.
```

related commands

- Keypad—SUBS
- Line—SUBSTITUTE

SEL (Select) Command—Nokeypad

SYNTAX SEL

DESCRIPTION

The SEL (select) command marks the current cursor position as one end of a select range. To set up the select range, first position the cursor at one end of the text you want to select. Then issue the SEL command to have EDT mark that spot. Use the arrow keys or the **"move"** command to move the cursor to the other end of the selected text. The select range is now set. To use that select range, use SR (select range) as the entity specifier with your nokeypad mode command.

You can move the cursor either forward or backward in the buffer to set the other end of the select range once the initial end has been marked. To move backward toward the start of the buffer, use the Left and Up arrow keys or precede the "move" command with a minus sign (–). If you have included more text than you want in the select range, you can move the cursor toward the position marked by SEL, using the arrow keys and "move" commands to reduce the size of the range. Adjusting select ranges on VT100 terminals is easy because EDT displays the text in reverse video. On VT52 terminals, you might find it easier to use the DESEL command to cancel the select range and then start over.

The **DESEL** (deactivate select) command cancels a select range. The **TGSEL** (toggle select) command allows you to switch between SEL and DESEL with the same command, either setting SEL if no select range is active, or DESEL if there is a current select range. The **SSEL** (search and select) command performs a search and makes the found string the select range.

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that the select range include only whole lines.

EXAMPLES

example 1:

Creates a select range of the word **Digital** and then changes all lowercase letters to uppercase.

```
This manual was produced by Digital for its customers.

SEL W
CHGUSR

This manual was produced by DIGITAL for its customers.
```

example 2:

Creates a select range starting near the end of the third line and including the first two lines. Deletes the unwanted text.

```
The reason for this change of plans should be obvious:  no one
in their right mind would want to have a meeting in North Alaska in
the middle of winter even if they were an avid skier.  The meeting
is now relocated to Lake Tahoe.  For those of you wishing to see
the forty-ninth state, our summer meeting is scheduled for Fairbanks.

SEL
BL -2L
DSR

The meeting
is now relocated to Lake Tahoe.  For those of you wishing to see
the forty-ninth state, our summer meeting is scheduled for Fairbanks.
```

related commands

— Keypad—SELECT

SHL (Shift Left) Command—Nokeypad

SYNTAX

[count] **SHL**

DESCRIPTION

The SHL (shift left) command moves the entire buffer text eight characters (one tab stop) to the left. The first eight characters on each line no longer appear on the screen. Assuming that your screen width is set to 80, all characters in positions 80 through 87 that were previously truncated are now visible.

No characters are actually added to or deleted from the text when you use the SHL and **SHR** commands. The effects of the commands are merely visual and temporary. On a VT100 terminal with advanced video option, or on a VT102, you can use the **SET SCREEN** command to display up to 132 characters on a single line, but the size of the characters is reduced.

The count specifier designates the number of tab stops to shift left. When count is 4, the buffer is shifted 32 character positions to the left. Use the SHR (shift right) command to move the buffer contents to the right, back to the original position.

You can define the GOLD/ ← key sequence on VT100-type terminals to perform the SHL function in keypad mode. The following **DEFINE KEY** command can be included in your startup command file or issued during your editing session:

```
DEFINE KEY GOLD 15 AS "SHL."
```

EXAMPLE

Shows the effect of SHL on a line of text that has been truncated because it exceeds the screen width of 80.

```
AAAAAAAAbbbbbbbbCCCCCCCCddddddddEEEEEEEEffffffffGGGGGGGGhhhhhhhhIIIIIIIIjjjjjjjj◆
SHL
bbbbbbbbCCCCCCCCddddddddEEEEEEEEffffffffGGGGGGGGhhhhhhhhIIIIIIIIjjjjjjjjKKKKKKKK
```

related commands

None.

SHR (Shift Right) Command—Nokeypad

SYNTAX

[count] **SHR**

DESCRIPTION

The SHR (shift right) command moves the entire current buffer eight character positions (one tab stop) to the right. SHR works only after you have used the **SHL** (shift left) command. When SHL has been used to move the buffer over one tab stop, SHR will move the buffer back to its original position.

You cannot shift text farther to the right with SHR than the text has been moved to the left. Once column 1 of your buffer is again at the left edge of the screen, further SHR commands have no effect on the display.

No characters are actually added or deleted from the buffer when SHR or SHL is used. The effects of these commands are merely visual and temporary. On a VT100-type terminal with advanced video option, you can use the **SET SCREEN** command to display up to 132 characters on a line. However, the size of each character is reduced.

The count specifier designates the number of tab stops to shift right. When the count is 4, the buffer is shifted 32 character positions to the right. If the count used with the SHL command is 5, and you then give the command 4SHR, the buffer will end up still shifted eight character positions to the left. If you use the SHR command with a count specifier greater than the number of left shifts that have been made, EDT moves the text back to its original position and ignores the remaining repeats.

You can define the GOLD/ → key on VT100-type terminals to perform the SHR function in keypad mode. The following **DEFINE KEY** command can be included in your startup command file or issued during your editing session:

```
DEFINE KEY GOLD 14 AS "SHR."
```

EXAMPLE

Uses the SHL command to move the line 16 columns to the left. Then uses SHR to restore the characters in columns 9 through 16 to view.

```
AAAAAAAAbbbbbbbbCCCCCCCCddddddddEEEEEEEEffffffffGGGGGGGGhhhhhhhhIIIIIIIIjjjjjjjj◆
2SHL
CCCCCCCCddddddddEEEEEEEEffffffffGGGGGGGGhhhhhhhhIIIIIIIIjjjjjjjjKKKKKKKKllllllll◆
SHR
bbbbbbbbCCCCCCCCddddddddEEEEEEEEffffffffGGGGGGGGhhhhhhhhIIIIIIIIjjjjjjjjKKKKKKKK◆
```

related commands

None.

SN (Substitute Next) Command—Nokeypad

SYNTAX

[+|−][count] **SN**

DESCRIPTION

The SN (substitute next) command is a substitute command that does not take any string specifiers. It uses strings that have been stored in the search buffer (string-1) and the substitute buffer (string-2). (Remember that you cannot access either of these buffers.) The SN command looks for the next occurrence of string-1 (the current search string) and replaces it with string-2 (the current substitute string).

Since SN uses the contents of the search buffer as its search string, you must be aware of the current contents of that buffer. If you have used a string entity after your initial **S** (substitute) command, that new string will now be in the search buffer, not the string you want to use with the SN command.

With the count specifier you can repeat the substitution several times. Use the sign specifier to change EDT's direction for the SN command.

EXAMPLE

First uses the S command to make the initial substitution on line 2. Then uses SN to make the substitution on line 5.

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a languge that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
languge related to everyday speech.

S/languge/language/
SN

BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a language that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
language related to everyday speech.
```

related commands

- Keypad—SUBS
- Line—SUBSTITUTE NEXT

SSEL (Search and Select) Command—Nokeypad

SYNTAX

[+/–] **SSEL** *[+/–]"string"*

DESCRIPTION

The SSEL (search and select) command allows you to find a string and designate it as a select range in one operation. The string must be enclosed in quotation marks, single (') or double ("). Do not use spaces to separate the command from the string. However, you can include leading spaces in the search string.

When you use SSEL, the contents of the search buffer are overwritten by the string. To search and select that string again (or whatever string is currently in the search buffer), simply type two quotation marks with no intervening characters, as follows:

```
SSEL""
```

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that a select range contain only whole lines. Thus, you can only use the SSEL command for this purpose if the string is an entire line of text. You might find it easier to create longer select ranges with the nokeypad **SEL** command.

EXAMPLES

example 1:

First creates a select range from the search string **[SP]EQUIPMENT CORPORATION**. Then deletes the select range. Finally creates a new select range from the next occurrence of the same search string.

```
DIGITAL EQUIPMENT CORPORATION
Digital Equipment Corporation is located in Maynard, MA.

SSEL" EQUIPMENT CORPORATION"
DSR

DIGITAL
Digital Equipment Corporation is located in Maynard, MA.

SSEL""

DIGITAL
Digital Equipment Corporation is located in Maynard, MA.
```

related commands

None.

string specifier—Nokeypad

SYNTAX

"string"
'string'
/string-1/string-2/

DESCRIPTION

The string specifier is generally used either to locate characters in a buffer or to replace the located characters. When a string specifier is used to locate a piece of text, it is referred to as the search string. All three editing modes use search strings. Generally a search string must be enclosed in quotation marks, single (') or double ("), to distinguish it from other characters in the command line.

When strings are used in EDT substitute commands, they are generally referred to as string-1 or string-2. String-1 is always a search string. EDT must locate that string and then replace it with string-2, the substitute string. Strings in substitute commands must be surrounded by delimiters to separate them from the other characters in the command line. A variety of nonalphanumeric characters can be used as delimiters as long as the delimiter does not appear in either string-1 or string-2. (In line mode you cannot use the percent sign (%) or the underscore (_) as string delimiters.) All delimiters used in a single substitute command line must be identical.

Whenever you issue a search string, EDT overwrites the contents of the search buffer. Similarly, when you issue a substitute string, EDT overwrites the contents of the substitute buffer.

The string entity in nokeypad mode refers to all the characters in the text between the initial cursor position and the start of the search string. However, only the search string itself is stored in the search buffer.

The search and substitute buffers cannot be edited or entered. Their names never appear in the SHOW BUFFER list. You can use the nokeypad **CLSS** (clear search string) command to delete the contents of the search buffer.

EDT has a number of ways to perform searches. See the discussion of the **SET SEARCH** command for information about the EDT search parameters.

EXAMPLES

example 1:

Substitutes the string **BASIC** for the string **COBOL**.

```
S/COBOL/BASIC/
```

example 2:

Deletes all characters starting at the current cursor position and continuing until the next period is reached.

```
D"."
```

example 3:

Finds the string **Friday, May 13th** and makes it a select range.

```
SSEL"Friday, May 13th"
```

TAB Command—Nokeypad

SYNTAX

[count] **TAB**

DESCRIPTION

The TAB command moves text to the right, regardless of EDT's direction. The number of column positions that the text moves depends on the cursor position, the **SET TAB** value, if one is in effect, and the indentation level count, if one is in effect. (SET NOTAB is the default.)

EDT has preset tab stops every eight characters, regardless of how your terminal is set. If no SET TAB command has been given, the TAB command moves the cursor character, as well as all the characters on the current line to the right of the cursor, to the nearest preset tab position.

If a SET TAB command has been given, the TAB command moves the entire line to the column designated by the SET TAB value only if the cursor is located in column 1. If the cursor is located anywhere else on the line, the TAB command moves the text to the nearest preset EDT tab stop.

If a tab indentation level count is in effect and the cursor is located in column 1 of the line, the TAB command moves the text to the indentation level position. The indentation level count is determined by three functions: 1) **TC** (tab compute) which can be used to compute the indentation level count, 2) **TD** (tab decrement) which decrements the count, and 3) **TI** (tab increment) which increments the count. Use the **SHOW TAB** command to find out the current SET TAB value and the indentation level count.

The **TADJ** command indents whole lines of text by the SET TAB value.

EXAMPLE

Using EDT's default tab settings (SET NOTAB), indents the lines of text, each one eight columns farther than the one above.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.
```

(Start with the cursor at the beginning of the second line.)

```
TAB
```

(Move the cursor to the beginning of the third line.)

```
2TAB
```

(Move the cursor to the beginning of the fourth line.)

```
3TAB
```

```
This is the first line of text.
        This is the second line of text.
                This is the third line of text.
                        This is the fourth line of text.
```

related commands

— Keypad—TAB, CTRL/I

TADJ (Tab Adjust) Command—Nokeypad

SYNTAX

[+/−][level-count] **TADJ** *[+/−][entity-count]entity*

DESCRIPTION

The TADJ (tab adjust) command uses the value established by the line mode **SET TAB** command to indent lines of text. If no SET TAB value exists for the editing session, TADJ does nothing.

The level count specifier determines how many tab stops the text will be indented. The TADJ command moves text to the right. If you precede the level count specifier with a minus sign, you can move text, which has already been indented, back toward the left margin.

The entity count determines how many lines, paragraphs, or pages will be affected by the TADJ command. A minus sign before the entity or entity count determines whether EDT will work forward or backward in determining which entities to indent. Note that 5TADJL is *not* identical to TADJ5L. You can use both a level count and an entity count with TADJ, for example, 2TADJ4L.

To find the current SET TAB value, use the line mode **SHOW TAB** command. EDT's default is SET NOTAB. Notice that the level count specifier has no effect on the tab level displayed by the SHOW TAB command.

EXAMPLE

Using a SET TAB value of 3, indents line 2 one tab stop and line 3 two tab stops.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.

EXT SET TAB 3
NL TADJL
NL 2TADJL

This is the first line of text.
   This is the second line of text.
      This is the third line of text.
```

related commands

- Keypad—CTRL/T
- Line—TAB ADJUST

TC (Tab Compute) Command—Nokeypad

SYNTAX

TC

DESCRIPTION

The TC (tab compute) command sets the indentation level count for the **TAB** command. The indentation level is the number of columns starting at the left of the screen that you want to leave blank before beginning a line of text. The indentation level in EDT must be a multiple of the **SET TAB** value. The indentation level count is the quotient of the indentation level divided by the SET TAB value. Use the line mode SET TAB command to establish the SET TAB value.

To use the TC command, first move the cursor as many positions to the right as you want to indent a block of text when you give the TAB command. Remember that this cursor position must be a multiple of the SET TAB value. If it is not, EDT prints an error message when you issue the TC command.

Once the cursor is in place, issue the TC command. You have now set the indentation level. If you divide the indentation level by the SET TAB value, you have the indentation level count. EDT retains the indentation level count for the remainder of the editing session or until you reset the count with a subsequent TC command, TD (tab decrement) command, or TI (tab increment) command. The TC command has no effect on EDT if SET NOTAB (the default) is in effect.

When you give the TC command, nothing happens to the text you are editing. This command simply tells EDT how to process the next TAB command(s). In order to indent any lines of text, you must issue the TAB command.

To find out the current SET TAB value and indentation level count, use the line mode **SHOW TAB** command.

EXAMPLE

Using a SET TAB value of 3, establishes a tab level count of 5 in order to indent the first line of text 15 columns.

```
This is the first line of text.
This is the second line of text.

EXT SET TAB 3
```

(Move the cursor over 15 places so that it is on the **s** in ***first***.)

```
TC
EXT SHOW TAB
tab size 3; tab level 5
```

(Now move the cursor back to the beginning of the first line.)

```
TAB

              This is the first line of text.
This is the second line of text.
```

related commands

— Keypad—CTRL/A

TD (Tab Decrement) Command—Nokeypad

SYNTAX

[count] **TD**

DESCRIPTION

The TD (tab decrement) command decreases the current indentation level count. The indentation level count is generally set by the **TC** (tab compute) command. It can be altered by a subsequent TC command, TD command, or **TI** (tab increment) command. The TD command only resets the indentation level count. You must use the **TAB** command to move the text. TD has no effect on EDT if SET NOTAB (the default) is in effect. Use the line mode **SET TAB** command to establish a SET TAB value.

The count specifier determines the amount that the indentation level count decreases. If no specifier is given, the indentation level count decreases by one.

To find out the current SET TAB value and indentation level count, use the line mode **SHOW TAB** command.

EXAMPLE

Using a SET TAB value of 4, sets the tab level count to 4. Then formats the text, using TAB to indent the lines, and TD to change the indentation level.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.

EXT SET TAB 4
```

(Move the cursor 16 places to the right.)

```
TC
EXT SHOW TAB
tab size 4; tab level 4
```

(Move the cursor to the beginning of the first line.)

```
TAB
```

(Move the cursor to the beginning of the second line.)

```
TD
TAB
```

(Move the cursor to the beginning of the third line.)

```
TD
TAB
```

(Move the cursor to the beginning of the fourth line.)

```
TD
TAB
```

TD (Tab Decrement) Command

```
            This is the first line of text.
        This is the second line of text.
    This is the third line of text.
This is the fourth line of text.

EXT SHOW TAB
tab size 4; tab level 1
```

---

related commands

— Keypad—CTRL/D

TGSEL (Toggle Select) Command—Nokeypad

SYNTAX

TGSEL

DESCRIPTION

The TGSEL (toggle select) command combines the functions of the **SEL** and **DESEL** commands into one. When there is an active select range, the TGSEL command cancels it, performing the same function as the DESEL command. When there is no active select range, TGSEL initiates the process of creating a select range, just as the SEL command does. The character that the cursor is on when you issue the TGSEL command constitutes one end of the select range. Move the cursor to the other end to define the limits of the select range.

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that the select range contain only whole lines.

EXAMPLE

First creates a select range of lines 2 through 4 and then cancels it.

```
Market Requirements
Preliminary Business Plan
Engineering Plan
Training Plan

TGSEL
3L

Market Requirements
Preliminary Business Plan
Engineering Plan
Training Plan

-

TGSEL

Market Requirements
Preliminary Business Plan
Engineering Plan
Training Plan

-
```

related commands

— Keypad—SELECT/RESET

TI (Tab Increment) Command—Nokeypad

SYNTAX

[count] **TI**

DESCRIPTION

The TI (tab increment) command increases the current indentation level count. The indentation level count is generally set by the **TC** (tab compute) command. It can be altered by a subsequent TC command, TI command, or **TD** (tab decrement) command. The TI command only resets the indentation level count. You must use the **TAB** command to move the text. TI has no effect on EDT if SET NOTAB (the default) is in effect. Use the line mode **SET TAB** command to establish the SET TAB value.

The count specifier determines the amount that the indentation level count increases. If no specifier is given, the indentation level count increases by one.

To find out the current SET TAB value and indentation level count, use the line mode **SHOW TAB** command.

EXAMPLE

Using a SET TAB value of 5, sets the tab level count to 2. Then formats the text using TAB to indent the lines, and TI to change the indentation level.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.

EXT SET TAB 5
```

(Move the cursor 10 places to the right.)

```
TC
EXT SHOW TAB
tab size 5; tab level 2
```

(Move the cursor to the beginning of the first line.)

```
TAB
```

(Move the cursor to the beginning of the second line.)

```
TI
TAB
```

(Move the cursor to the beginning of the third line.)

```
TI
TAB
```

(Move the cursor to the beginning of the fourth line.)

```
TI
TAB
```

```
        This is the first line of text.
             This is the second line of text.
                  This is the third line of text.
                       This is the fourth line of text.

EXT SHOW TAB
tab size 5; tab level 5
```

related commands

— Keypad—CTRL/E

TOP Command—Nokeypad

SYNTAX

TOP

DESCRIPTION

The TOP command moves the line on which the cursor is located to the top of the display screen. If there are not enough lines between the current cursor position and the end of the buffer to fill the screen, TOP does nothing. When you use the TOP command, the cursor remains in the same position on the current line that it was in before TOP was issued.

Normally you must have 21 lines below the current line in order for TOP to have any effect on your screen display. However, if you have used the line mode **SET LINES** command to limit the number of lines per screen, TOP needs fewer lines between the cursor line and the end of the buffer in order to have an effect on the screen display.

Once a line has been placed at the top of the screen by the TOP command, EDT tries to keep it there as long as possible.

EXAMPLE

Moves the line that the cursor is on to the top of the screen. SET LINES 5 limits the number of lines EDT displays.

```
EXT SET LINES 5

This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.
This is the fifth line of text.

TOP

This is the fourth line of text.
This is the fifth line of text.
This is the sixth line of text.
This is the seventh line of text.
This is the eighth line of text.
```

related commands

— Keypad—PAGE

UNDC (Undelete Character) Command—Nokeypad

SYNTAX

[count] **UNDC**

DESCRIPTION

The UNDC (undelete character) command inserts the current contents of the delete character buffer into your text to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is positioned on the inserted character if you used DC to delete the character. The cursor is positioned to the right of the inserted character if you used –DC (or D–C) to delete the character.

When you include a count specifier, EDT inserts the contents of the delete character buffer as many times as the count value.

Both the **DC** command and the **DELETE** key place the character they delete in the delete character buffer. Each time you use the DC command or the DELETE key, the contents of the delete character buffer are overwritten. The buffer contains only the most recently deleted character. When you use a count specifier with the DC command or with the DELETE key, only the last character deleted is in the delete character buffer. If no character has been deleted thus far during the current EDT session, UNDC inserts nothing. Note that if you use the DELETE key to delete characters in a command line, these characters are *not* stored in the delete character buffer and will not affect the character inserted by UNDC.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in your text and you delete it. When you undelete this character, EDT changes the <CR> character into a line terminator and then inserts the line terminator in your text.

EXAMPLE

First loads the asterisk character (*) into the delete character buffer. Then inserts five asterisks in the text.

```
This is the end.
*_
D-C
5UNDC

This is the end.
*****_
```

related commands

— Keypad—UND C

UNDL (Undelete Line) Command—Nokeypad

SYNTAX

[count] **UNDL**

DESCRIPTION

The UNDL command inserts the current contents of the delete line buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to a new line below the current line, if there is a line terminator at the end of the buffer contents. Otherwise, the text just moves to the right. The cursor is now on the first character of the inserted test if you deleted the text in the forward direction (for example, DL, DNL, or DEL). If you deleted the text toward the beginning of the buffer (for example, DBL, –DEL, or –DL), the cursor is located to the right of the inserted text.

When the count specifier is used, the contents of the delete ne buffer are inserted as many times as the count value.

The delete line buffer is loaded by using the **D** (delete) command with a line entity: L, BL, EL, or NL. Each time a DL type command is used, the contents of the delete line buffer are overwritten. The delete line buffer can contain only one line or line portion at a time. If you use a count specifier with the D command, for example D2L or 3DL, only the last line or line portion deleted is in the delete line buffer. If no line has been deleted thus far in your EDT session, UNDL inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer.

EXAMPLES

example 1:

Using DL and UNDL, reorganizes the list so that the cities are in alphabetical order by state.

```
Montgomery, Alabama
Juneau, Alaska
Little Rock, Arkansas
Phoenix, Arizona
```

```
DL
```

(Move the cursor to the **L** in **Little Rock**.)

```
UNDL
```

```
Montgomery, Alabama
Juneau, Alaska
Phoenix, Arizona
Little Rock, Arkansas
```

example 2:

Using DL and UNDL, inserts the second line of text after **you**.

```
When you give it more thought, please contact me.
have had a chance to
```

```
DL
```

(Move the cursor to the **g** in **give**.)

```
UNDL
```

```
When you have had a chance to
give it more thought, please contact me.
```

example 3:

Using DL and UNDL, duplicates the separator line in the current location.

```
This is the end.
**********
```

```
DL
2UNDL
```

```
This is the end.
**********
**********
```

related commands

— Keypad—UND L

UNDW (Undelete Word) Command—Nokeypad

SYNTAX

[count] **UNDW**

DESCRIPTION

The UNDW (undelete word) command inserts the current contents of the delete word buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is now located on the first character of the inserted word or word portion if the deletion was made in the forward direction (for example, with DW or DEW). If the deletion was made toward the beginning of the buffer (for example, DBW or –DW), the cursor is located to the right of the inserted text.

When you include the count specifier, the contents of the delete word buffer are inserted as many times as the count value.

The delete word buffer is loaded by using the **D** (delete) command with a word entity: W, BW, or EW. Each time you use one of these commands, the contents of the delete word buffer are overwritten. The current contents of the buffer are the most recently deleted word or word portion. When you use a count specifier with a DW type command, only the last word or word portion is in the delete word buffer. If no word has been deleted thus far in your EDT session, UNDW inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer.

EXAMPLES

example 1:

Using DW and UNDW, inserts the word **PRINT** at the beginning of lines 1, 2, and 3.

```
LETTER1.MEM
LETTER2.MEM
LETTER3.MEM
PRINT LETTER4.MEM
```

```
DW
```

(Move the cursor to the **L** in **LETTER1.MEM**.)

```
UNDW
```

(Move the cursor to the **L** in **LETTER2.MEM**.)

```
UNDW
```

(Move the cursor to the **L** in **LETTER3.MEM**.)

```
UNDW
```

(Move the cursor to the **L** in **LETTER4.MEM**.)

```
UNDW

PRINT LETTER1.MEM
PRINT LETTER2.MEM
PRINT LETTER3.MEM
PRINT LETTER4.MEM
```

example 2:

Using DW and UNDW, creates a separator line below the current line.

```
***** This is the end.
DW
```

(Move the cursor to the beginning of the next line, which is blank.)

```
5UNDW

This is the end.
***** ***** ***** ***** *****
```

related commands

— Keypad—UND W

Up Arrow—Nokeypad

KEY SEQUENCE

DESCRIPTION

The Up Arrow key moves the cursor up one line toward the top of the buffer regardless of EDT's direction. **-V.** is the nokeypad definition for Up Arrow.

When you use the Up Arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of the short line. If you continue to use Up Arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for Up Arrow and resets it the next time you use the function.

Note that Up Arrow is not a true nokeypad command.

EXAMPLE

Moves the cursor from the end of the last line to the end of the first line.

```
La Paz, Bolivia
Lima, Peru
Montevideo, Uruguay
Parimaribo, Surinam
Quito, Ecuador
Santiago, Chile_
```

```
Quito, Ecuador_
Santiago, Chile
```

 + 

```
Montevideo, Uruguay
Parimaribo, Surinam
Quito, Ecuador
```

 + 

```
La Paz, Bolivia_
Lima, Peru
Montevideo, Uruguay
```

related commands

— Keypad—Up Arrow

XLATE Command—Nokeypad

SYNTAX

XLATE*string ^Z*

DESCRIPTION

The XLATE command can be used when EDT has been called by a running program on a VAX/VMS system.

The string specifier enables you to pass information back to the calling program. CTRL/Z signals the end of the string. The nature of the string is determined by the type of program being run and the kind of information you need to pass to the program. For example, the information could be a DCL command or a variable that the program needs to continue running.

Since XLATE is available only as a nokeypad command, you might want to define a key to have the XLATE function. You can either include the exact information you want to pass to the program as the string specifier, or you can include a prompt so that you can enter the information when you press the function key. For example:

```
DEFINE KEY GOLD X AS "XLATEdirectory^Z."
DEFINE KEY GOLD X AS "XLATE?'Enter data: '^Z."
```

For more information on using the XLATE command with callable EDT, see the EDT routines section in *VAX/VMS Utility Routines Reference Manual.*

related commands

None.

A DEC Multinational Character Set

The DEC Multinational Character Set is an 8-bit character set with 256 characters. Each character is assigned a decimal equivalent number. These numbers range from 0 to 255. The first 128 characters in the set correspond to the ASCII character set.

Graphics shown in parentheses are ASCII control characters. These are produced on most terminals by pressing the key indicated while holding down the CONTROL key. Characters with numbers greater than 127 can only be entered on VT100 and VT52 type terminals from a screen mode. For terminals with LK201 keyboards, you can use the compose sequences to have the graphic symbol for characters 128 through 255 appear on the screen. For other VT100-type and VT52 terminals, you must use the keypad SPECINS function or the nokeypad ASC command to enter these characters in your text; EDT then displays the EDT symbol that corresponds to the character, not the character graphic.

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
(@)	^@	0	NUL	null character
(A)	^A	1	SOH	start of heading
(B)	^B	2	STX	start of text
(C)	^C	3	ETX	end of text
(D)	^D	4	EOT	end of transmission
(E)	^E	5	ENQ	enquiry
(F)	^F	6	ACK	acknowledge
(G)	^G	7	BEL	bell
(H)	^H	8	BS	backspace
(I)		9	HT	horizontal tabulation
(J)	<LF>	10	LF	line feed
(K)	<VT>	11	VT	vertical tabulation
(L)	<FF>	12	FF	form feed
(M)	<CR>	13	CR	carriage return
(N)	^N	14	SO	shift out
(O)	^O	15	SI	shift in
(P)	^P	16	DLE	data link escape
(Q)	^Q	17	DC1	device control 1
(R)	^R	18	DC2	device control 2
(S)	^S	19	DC3	device control 3
(T)	^T	20	DC4	device control 4

ZK-1737/1-84

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
(U)	^U	21	NAK	negative acknowledge
(V)	^V	22	SYN	synchronous idle
(W)	^W	23	ETB	end of transmission block
(X)	^X	24	CAN	cancel
(Y)	^Y	25	EM	end of medium
(Z)	^Z	26	SUB	substitute
([)	<ESC>	27	ESC	escape
(\)	^\	28	FS	file separator
(])	^]	29	GS	group separator
(^)	^^	30	RS	record separator
(_)	^_	31	US	unit separator
		32	SP	space
!	!	33	!	exclamation point
"	"	34	"	quotation marks (double quote)
#	#	35	#	number sign
$	$	36	$	dollar sign
%	%	37	%	percent sign
&	&	38	&	ampersand
'	'	39	'	apostrophe (single quote)
(	(	40	(	opening parenthesis
)	)	41	)	closing parenthesis
*	*	42	*	asterisk
+	+	43	+	plus
,	,	44	,	comma
-	-	45	-	hyphen or minus
.	.	46	.	period or decimal point
/	/	47	/	slash
0	0	48	0	zero
1	1	49	1	one
2	2	50	2	two
3	3	51	3	three
4	4	52	4	four
5	5	53	5	five
6	6	54	6	six
7	7	55	7	seven
8	8	56	8	eight
9	9	57	9	nine
:	:	58	:	colon
;	;	59	;	semicolon
<	<	60	<	less than
=	=	61	=	equals
>	>	62	>	greater than
?	?	63	?	question mark
@	@	64	@	commercial at
A	A	65	A	uppercase A
B	B	66	B	uppercase B
C	C	67	C	uppercase C
D	D	68	D	uppercase D
E	E	69	E	uppercase E
F	F	70	F	uppercase F
G	G	71	G	uppercase G
H	H	72	H	uppercase H

ZK-1737/2-84

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
I	I	73	I	uppercase I
J	J	74	J	uppercase J
K	K	75	K	uppercase K
L	L	76	L	uppercase L
M	M	77	M	uppercase M
N	N	78	N	uppercase N
O	O	79	O	uppercase O
P	P	80	P	uppercase P
Q	Q	81	Q	uppercase Q
R	R	82	R	uppercase R
S	S	83	S	uppercase S
T	T	84	T	uppercase T
U	U	85	U	uppercase U
V	V	86	V	uppercase V
W	W	87	W	uppercase W
X	X	88	X	uppercase X
Y	Y	89	Y	uppercase Y
Z	Z	90	Z	uppercase Z
[	[	91	[	opening bracket
\	\	92	\	back slash
]	]	93	]	closing bracket
^	^	94	^	circumflex
_	_	95	_	underline (underscore)
`	`	96	`	grave accent
a	a	97	a	lowercase a
b	b	98	b	lowercase b
c	c	99	c	lowercase c
d	d	100	d	lowercase d
e	e	101	e	lowercase e
f	f	102	f	lowercase f
g	g	103	g	lowercase g
h	h	104	h	lowercase h
i	i	105	i	lowercase i
j	j	106	j	lowercase j
k	k	107	k	lowercase k
l	l	108	l	lowercase l
m	m	109	m	lowercase m
n	n	110	n	lowercase n
o	o	111	o	lowercase o
p	p	112	p	lowercase p
q	q	113	q	lowercase q
r	r	114	r	lowercase r
s	s	115	s	lowercase s
t	t	116	t	lowercase t
u	u	117	u	lowercase u
v	v	118	v	lowercase v
w	w	119	w	lowercase w
x	x	120	x	lowercase x
y	y	121	y	lowercase y
z	z	122	z	lowercase z
{	{	123	{	opening brace
|	|	124	|	vertical line
}	}	125	}	closing brace
~	~	126	~	tilde

ZK-1737/3-84

DEC Multinational Character Set

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
DEL	<DEL>	127	DEL	delete, rubout
	<X80>	128	---	[reserved]
	<X81>	129	---	[reserved]
	<X82>	130	---	[reserved]
	<X83>	131	---	[reserved]
	<IND>	132	IND	index
	<NEL>	133	NEL	next line
	<SSA>	134	SSA	start of selected area
	<ESA>	135	ESA	end of selected area
	<HTS>	136	HTS	horizontal tab set
	<HTJ>	137	HTJ	horizontal tab set with justification
	<VTS>	138	VTS	vertical tab set
	<PLD>	139	PLD	partial line down
	<PLU>	140	PLU	partial line up
	<RI>	141	RI	reverse index
	<SS2>	142	SS2	single shift 2
	<SS3>	143	SS3	single shift 3
	<DCS>	144	DCS	device control string
	<PU1>	145	PU1	private use 1
	<PU2>	146	PU2	private use 2
	<STS>	147	STS	set transmit state
	<CCH>	148	CCH	cancel character
	<MW>	149	MW	message waiting
	<SPA>	150	SPA	start of protected area
	<EPA>	151	EPA	end of protected area
	<X98>	152	---	[reserved]
	<X99>	153	---	[reserved]
	<X9A>	154	---	[reserved]
	<CSI>	155	CSI	control sequence introducer
	<ST>	156	ST	string terminator
	<OSC>	157	OSC	operating system command
	<PM>	158	PM	privacy message
	<APC>	159	APC	application program command
	<XA0>	160	---	[reserved]
¡	<!!>	161	¡	inverted exclamation mark
¢	<C/>	162	¢	cent sign
£	<L->	163	£	pound sign
	<XA4>	164	---	[reserved]
¥	<Y->	165	¥	yen sign
	<XA6>	166	---	[reserved]
§	<S0>	167	§	section sign
¤	<X0>	168	¤	general currency sign
©	<C0>	169	©	copyright sign
ª	<a_>	170	ª	feminine ordinal indicator
«	<< >	171	«	angle quotation mark left
	<XAC>	172	---	[reserved]
	<XAD>	173	---	[reserved]
	<XAE>	174	---	[reserved]
	<XAF>	175	---	[reserved]
°	<0^>	176	°	degree sign
±	<+->	177	±	plus/minus sign
²	<2^>	178	²	superscript 2

ZK-1737/4-84

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
³	<3^>	179	³	superscript 3
	<XB4>	180	---	[reserved]
µ	</U>	181	µ	micro sign
¶	<P!>	182	¶	paragraph sign, pilcrow
·	<.^>	183	·	middle dot
	<XB8>	184	---	[reserved]
¹	<1^>	185	¹	superscript 1
º	<o_>	186	º	masculine ordinal indicator
»	<>>>	187	»	angle quotation mark right
¼	<14>	188	¼	fraction one quarter
½	<12>	189	½	fraction one half
	<XBE>	190	---	[reserved]
¿	<??>	191	¿	inverted question mark
À	<A`>	192	À	uppercase A with grave accent
Á	<A´>	193	Á	uppercase A with acute accent
Â	<A^>	194	Â	uppercase A with circumflex
Ã	<A~>	195	Ã	uppercase A with tilde
Ä	<A">	196	Ä	uppercase A with umlaut,(diaeresis)
Å	<A*>	197	Å	uppercase A with ring
Æ	<AE>	198	Æ	uppercase AE diphthong
Ç	<C,>	199	Ç	uppercase C with cedilla
È	<E`>	200	È	uppercase E with grave accent
É	<E´>	201	É	uppercase E with acute accent
Ê	<E^>	202	Ê	uppercase E with circumflex
Ë	<E">	203	Ë	uppercase E with umlaut, (diaeresis)
Ì	<I`>	204	Ì	uppercase I with grave accent
Í	<I´>	205	Í	uppercase I with acute accent
Î	<I^>	206	Î	uppercase I with circumflex
Ï	<I">	207	Ï	uppercase I with umlaut, (diaeresis)
	<XD0>	208	---	[reserved]
Ñ	<N~>	209	Ñ	uppercase N with tilde
Ò	<O`>	210	Ò	uppercase O with grave accent
Ó	<O´>	211	Ó	uppercase O with acute accent
Ô	<O^>	212	Ô	uppercase O with circumflex
Õ	<O~>	213	Õ	uppercase O with tilde
Ö	<O">	214	Ö	uppercase O with umlaut, (diaeresis)
Œ	<OE>	215	Œ	uppercase OE ligature
Ø	<O/>	216	Ø	uppercase O with slash
Ù	<U`>	217	Ù	uppercase U with grave accent
Ú	<U´>	218	Ú	uppercase U with acute accent
Û	<U^>	219	Û	uppercase U with circumflex
Ü	<U">	220	Ü	uppercase U with umlaut, (diaeresis)
Ÿ	<Y">	221	Ÿ	uppercase Y with umlaut, (diaeresis)
	<XDE>	222	---	[reserved]
ß	<ss>	223	ß	German lowercase sharp s
à	<a`>	224	à	lowercase a with grave accent
á	<a´>	225	á	lowercase a with acute accent
â	<a^>	226	â	lowercase a with circumflex
ã	<a~>	227	ã	lowercase a with tilde
ä	<a">	228	ä	lowercase a with umlaut, (diaeresis)
å	<a*>	229	å	lowercase a with ring
æ	<ae>	230	æ	lowercase ae diphthong

ZK-1737/5-84

DEC Multinational Character Set

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
ç	<c, >	231	ç	lowercase c with cedilla
è	<e` >	232	è	lowercase e with grave accent
é	<e´ >	233	é	lowercase e with acute accent
ê	<e^ >	234	ê	lowercase e with circumflex
ë	<e" >	235	ë	lowercase e with umlaut, (diaeresis)
ì	<i` >	236	ì	lowercase i with grave accent
í	<i´ >	237	í	lowercase i with acute accent
î	<i^ >	238	î	lowercase i with circumflex
ï	<i" >	239	ï	lowercase i with umlaut, (diaeresis)
	<XF0>	240	---	[reserved]
ñ	<n~ >	241	ñ	lowercase n with tilde
ò	<o` >	242	ò	lowercase o with grave accent
ó	<o´ >	243	ó	lowercase o with acute accent
ô	<o^ >	244	ô	lowercase o with circumflex
õ	<o~ >	245	õ	lowercase o with tilde
ö	<o" >	246	ö	lowercase o with umlaut, (diaeresis)
œ	<oe>	247	œ	lowercase oe ligature
ø	<o/>	248	ø	lowercase o with slash
ù	<u` >	249	ù	lowercase u with grave accent
ú	<u´ >	250	ú	lowercase u with acute accent
û	<u^ >	251	û	lowercase u with circumflex
ü	<u" >	252	ü	lowercase u with umlaut, (diaeresis)
ÿ	<y" >	253	ÿ	lowercase y with umlaut, (diaeresis)
	<XFE>	254	---	[reserved]
	<XFF>	255	---	[reserved]

ZK-1737/6-84

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

W

X

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line