Guide to VMS Text Processing

Order Number: AA–LA13A–TE

April 1988

This manual contains tutorial information about the EVE editor, the EDT editor, and DIGITAL Standard Runoff (DSR).

Revision/Update Information:	This document supersedes the *Guide to Text Processing on VAX/VMS*, Version 4.4
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

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™

ZK4528

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

Contents

EXAMPLES

FIGURES

TABLES

Preface

Intended Audience

This manual is intended for the novice user.

Document Structure

This manual is divided into three chapters and two appendixes.

- Chapter 1 explains how to use the EVE editor.
- Chapter 2 explains how to use the EDT editor.
- Chapter 3 explains how to use DIGITAL Standard Runoff (DSR).
- Appendix A explains how to customize the EVE editor using different types of startup files.
- Appendix B shows EDT commands and functions along with their EVE equivalents.

Associated Documents

For detailed information on the EVE editor, which is the interface to the VAX Text Processing Utility (VAXTPU), see the *VAX Text Processing Utility Manual.* For further information on the EDT editor, see the *VAX EDT Reference Manual.* For reference information on DSR, see the *VAX DIGITAL Standard Runoff Reference Manual.*

Conventions

Convention	Meaning
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (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.

New and Changed Features

EVE Version 2.0 offers many new features, including the following:

- EDT and WPS keypads. You can now select an EDT keypad or a WPS keypad as your default keypad. For more information, see the help topic EDT Differences or WPS Differences.
- Initialization files and private defaults. An initialization file sets the characteristics of the editing environment. You can use initialization files to modify the default characteristics for an EVE buffer. Execute initialization files when invoking EVE or at any time during an editing session.
- Wildcard searches. In addition to searching for specific text strings, you can now use wildcards in your search strings. EVE uses both VMS and ULTRIX wildcards.
- New commands to control the cursor, SET CURSOR FREE and SET CURSOR BOUND.
- Improved multiple buffers and multiple windows. See the help topic New Features for more information.

1 Editing Files With EVE

EVE, the Extensible VAX Editor, is an editor built on the VAX Text Processing Utility (VAXTPU), a high performance, programmable text processor. Using EVE, you can create and edit new files or edit existing files. You can add text to a file and modify or format that text. EVE is interactive, so you see the changes to a file as you make them.

Unlike the EDT editor (described in Chapter 2), EVE lets you display more than one buffer on the screen at a time and edit more than one file during the same editing session. EVE is easy to customize or extend using EVE commands and VAXTPU procedures.

1.1 Beginning and Ending an Editing Session

To begin an editing session, invoke EVE with the DCL command EDIT/TPU. In an editing session you can create and edit a new file or edit an existing file. The session ends when you enter the EXIT or QUIT command. Exiting from EVE typically produces a new file or a new version of an existing file.

1.1.1 Invoking EVE

You can start an editing session by creating a new file and inserting text into it during the session. You can also specify an existing file when you invoke EVE.

To begin an EVE editing session, enter the DCL command EDIT/TPU. Specify a file name on the command line if you want to edit an existing file or assign a name to a new file. For example, to invoke EVE and create a new file named NEWFILE.DAT, enter the following command:

```
$ EDIT/TPU NEWFILE.DAT
```

This command produces a screen that appears as follows:

```
[End of file]


Buffer NEWFILE.DAT                          | Insert | Forward

Editing new file: could not find WORKDISK:[USER]NEWFILE.DAT
```

EVE inserts the text of the file you are editing into a temporary holding area called a *buffer*. The contents of the buffer are shown in an area of your screen that is called a *window*. EVE buffers exist only during the editing session. When you end an editing session, you direct EVE to save or discard the contents of a buffer.

The end-of-file marker defines the end of an EVE buffer. It is visible only on the screen and does not become part of your file. When you add text to the buffer, the end-of-file marker moves downward. The marker might not be visible when you are viewing the beginning of a buffer that contains many lines of text.

A highlighted status line appears at the bottom of the EVE window and provides information about the EVE buffer. The status line shows the buffer name, current mode (insert or overstrike), and current direction (forward or reverse).

If you invoke EVE with a file name, an informational message appears in the message window beneath the highlighted status line. The message states either that the file is a new file or that a certain number of lines were read from an existing file. EVE communicates with you throughout the editing session by displaying messages in the message window.

To invoke EVE to edit an existing file named SCHEDULE.DAT, you enter the following command:

```
$ EDIT/TPU SCHEDULE.DAT
```

The contents of the file SCHEDULE.DAT appear in the editing window on your screen.

When you invoke EVE to edit an existing file, you can use the asterisk wildcard character (*) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (%) as a substitute for one character in the file name and file type, and you can use the ellipsis wildcard character ([. . .]) as a substitute for a directory specification. Using wildcards in EVE follows the same rules as using wildcards in DCL. If only one match is made, the file is displayed on your screen. If more than one match is made, EVE displays a list of matching files and prompts you to provide a more complete file specification. If no match is made, the file name that you typed appears in the highlighted status line and the message *Editing new file. Could not find:* is displayed in the message window.

You might want to use a command symbol to invoke the EVE editor. For example, if you place the following statement in your login command file and execute your login command file, you only need to type EVE to invoke the EVE editor:

```
$ EVE == EDIT/TPU
```

Rather than name a file at the beginning of an editing session, you can invoke EVE with the command EDIT/TPU and then enter text into the buffer. You can save the text by writing it to a file, using the WRITE FILE command described in Section 1.7. Alternately, when you finish creating your file, EVE prompts for a file name with the following:

```
Enter a file name to write buffer MAIN; else press RETURN:
```

Type the name of the file and press the RETURN key to write out the buffer to a file.

1.1.2 Ending an Editing Session

Two different commands can end an EVE editing session. EVE produces a new version of the edited file when you end the session with the EXIT command. EVE discards your edits when you end a session with the QUIT command. Any existing versions of the files remain unchanged regardless of how the editing session is ended.

To save your edited text, use the EXIT command. Enter the EXIT command by pressing the F10 key (on VT200-series or VT300-series terminals) or by pressing CTRL/Z.

If you have modified the current buffer, EVE creates a new version of the file with the same file name and file type as the original version, with the version number incremented by 1. For example, if you use the EXIT command after modifying a file named FUN.DAT;1, the output file is named FUN.DAT;2:

```
Buffer FUN.DAT                                  | Insert | Forward

4 lines written to file WORKDISK:[USER]FUN.DAT;2
```

To exit from a session without saving your edits, use the QUIT command. Enter the QUIT command by pressing the DO key (PF4 on VT100-series terminals), typing QUIT at the *Command:* prompt, and pressing RETURN. For example, if you have modified a file named FUN.DAT and enter the QUIT command, the following display appears on your terminal screen:

```
Buffer FUN.DAT                                  | Insert | Forward

Buffer modifications will not be saved, continue quitting (Y or N)?
```

Type Y and press RETURN if you want to quit without saving the edits. If you change your mind and decide to save your edits, type N, press RETURN, and exit from the file using the EXIT command.

If you have modified buffers other than the current one, EVE asks if you want to save the contents of each buffer. If you type Y, EVE creates a new version of an existing file, incrementing the version number by 1. EVE prompts for a file name if no file currently exists.

1.2 Entering EVE Commands

Once you have invoked EVE, you can enter EVE commands to edit and manipulate text. There are two ways to enter EVE commands: by pressing predefined editing keys and by typing the commands themselves.

1.2.1 Using Defined Keys to Enter EVE Commands

EVE defines some editing keys by default. The predefined editing keys on VT200-series and VT300-series terminals include the minikeypad (located between the main keypad and the numeric keypad), certain function keys, and certain control key sequences. On VT100-series terminals, EVE automatically defines most of the numeric keypad keys, the arrow keys, and certain control keys. Each predefined editing key performs one editing command.

Throughout this chapter, EVE editing keys are referred to by their names, rather than by their location on the VT200-series, VT300-series, or VT100-series keyboards. For example, on a VT200-series or VT300-series terminal, the DO key is located at the top of the editing keypad and is labeled DO. On a VT100-series terminal, the DO key is located at the upper right of the numeric keypad and is labeled PF4. Figure 1–1 shows the predefined keys on the VT200-series and VT300-series terminal and Figure 1–2 shows the predefined keys on the VT100-series terminal.

Figure 1–1 Editing Keys—VT200-series and VT300-series Terminals

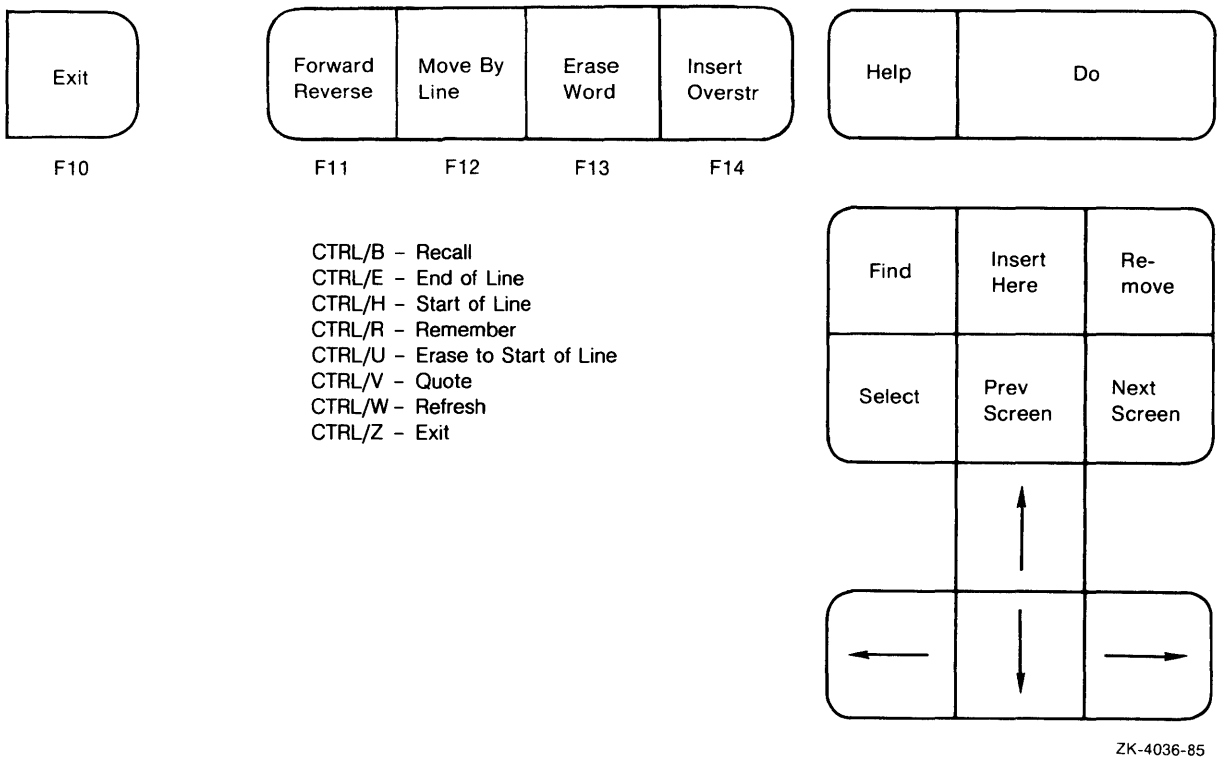
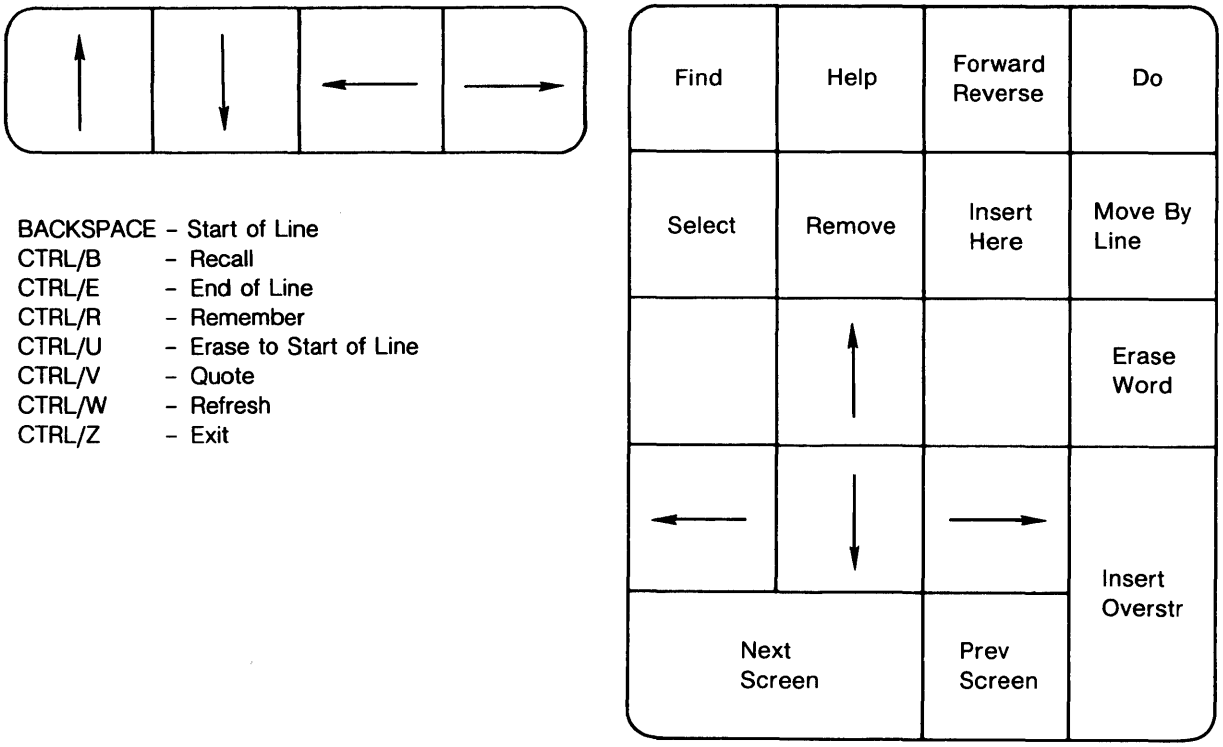


Figure 1–2 Editing Keys—VT100-series Terminals



EVE offers two default keypads in addition to the default EVE keypad. You can set an EDT keypad or a WPS keypad. Although neither fully implements EDT or WPS editing functions, they do provide most keypad functions.

You can define additional editing keys to enter commands or to perform editing operations that you frequently use. See Section 1.9 for information.

1.2.2 Typing EVE Commands

Besides using defined keys to enter commands, you can type out commands at the *Command:* prompt. When you type out EVE commands, you always perform the following three steps:

1. Press the DO key. EVE displays the *Command:* prompt.
2. Type the EVE command after the prompt.
3. Press either the RETURN key or the DO key to enter the command.

You can correct typing mistakes on the EVE command line by pressing DCL line editing keys. For example, use CTRL/U to erase to the beginning of the line, CTRL/E to move to the end of the line, and CTRL/B to recall the last command entered. By default, the editing mode of the EVE command line is the same as the editing mode of your terminal. (You can change the default with the DCL command SET TERMINAL prior to invoking EVE. Once in EVE, you can change the editing mode by pressing CTRL/A.)

To save keystrokes when you are typing EVE commands, you can use CTRL/B, abbreviate commands, use the REPEAT command, or press the DO key. Each method of saving keystrokes is described as follows:

- Press CTRL/B to recall the last EVE command you entered. Pressing CTRL/B again recalls the next to the last command and so forth. Continue pressing CTRL/B until the command you want appears on your screen, and press RETURN to enter it.
- Abbreviate EVE command names, making sure the abbreviation is unambiguous. If you enter an abbreviation that is not unique, EVE displays a list of matching commands and prompts you for a choice. Type enough additional characters to ensure that the abbreviation is unique, and press RETURN to enter the command. You can also abbreviate buffer names, file names, and mark names. Again, EVE provides a list of choices if you do not provide a unique abbreviation.
- Use the REPEAT command to repeat an EVE command or keystroke. Press the DO key, type REPEAT and the number of times it is to be repeated, and press RETURN. EVE repeats the next character or command you enter the specified number of times. For example, to insert the character p in your editing buffer 20 times, press the DO key, type REPEAT 20, and press RETURN. Then type p. EVE inserts p into the current buffer 20 times.
- Pressing the DO key twice activates the last command entered.

1.3 Editing Text

Once you know how to invoke the EVE editor and how to enter commands, you can use EVE commands to edit new and existing files. Different editing keys and commands allow you to position the cursor and perform such typical text editing operations as moving, erasing, and restoring text.

1.3.1 Moving the Cursor

When editing files with EVE, you move the cursor to the place in the text where you want to perform an editing function. Therefore, the more quickly and efficiently you move the cursor through the text, the more time you save in your editing session.

The following tables show the EVE editing keys and commands that move the cursor:

Editing Key	Moves the Cursor to the Following Position
Up arrow	Up one line.
Down arrow	Down one line.
Left arrow	One character or column to the left.
Right arrow	One character or column to the right.
CTRL/E	End of the current line.
CTRL/H	Beginning of the current line.

Editing Key	Moves the Cursor to the Following Position
Move By Line	In forward direction: end of the current line or, if the cursor is already at the end of a line, to the end of the next line. In reverse direction: beginning of the current line or, if the cursor is already at the beginning of a line, to the beginning of the previous line.
Next Screen	Forward in the current buffer. The number of lines the cursor moves depends on the size of the current window.
Previous Screen	Backward in the current buffer. The number of lines the cursor moves depends on the size of the current window.

Editing Command	Moves the Cursor to the Following Position
BOTTOM	End of the current buffer.
BUFFER	Puts the specified buffer in the current window and moves the cursor to the end of the buffer. Creates a new buffer if the specified buffer does not exist.
END OF LINE	End of the current line.
GET FILE	Creates a new buffer containing text of the specified file (or an empty buffer if the file does not exist) and puts the cursor at the beginning of the buffer. If entered a second time with the same file name during an editing session, EVE puts the existing buffer in the current window and positions the cursor at its last location in the buffer.
LINE	Beginning of the specified line in the current buffer.
MOVE BY PAGE	Next or previous page break, depending on the current direction.
MOVE BY WORD	Beginning of the next word, if direction is forward. In reverse direction, cursor moves to the beginning of the current word; if already there, EVE positions cursor at the beginning of the previous word.
NEXT WINDOW	Next window on your screen, assuming another exists. The cursor appears in the last location it occupied in that editing window.
PREVIOUS WINDOW	Previous window on screen, assuming another window exists. The cursor appears in the last location it occupied in that editing window.
SET CURSOR BOUND	Changes cursor mode. Cursor follows the flow of text and cannot be put into an unused portion of the buffer. Similar to cursor behavior in EDT and other editors.
SET CURSOR FREE	The default mode. Cursor is not bound to the flow of the text but can be put anywhere on the screen and text can be entered.
START OF LINE	Beginning of the current line.
TOP	Beginning of the current buffer.

The following example shows how to move the cursor through a buffer.

Invoke EVE and create the file SCHEDULE.DAT with the following command:

```
$ EDIT/TPU SCHEDULE.DAT
```

The command produces the following screen:

```
[End of file]

Buffer SCHEDULE.DAT                          | Insert | Forward

Editing new file: could not find WORKDISK:[USER]SCHEDULE.DAT
```

EVE positions the cursor at the top of the buffer and waits for you to enter text. The [End of file] marker moves down in the buffer as you enter the following text into the file SCHEDULE.DAT:

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer SCHEDULE.DAT                          | Insert | Forward

Editing new file: could not find WORKDISK:[USER]SCHEDULE.DAT
```

Notice that the cursor is positioned at the end of the text you inserted. To move the cursor to the beginning of the file, press the DO key, type TOP, and press RETURN.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer SCHEDULE.DAT                          | Insert | Forward
Command:  TOP
```

Press CTRL/E to move the cursor to the end of the first line of text. CTRL/E works the same way in EVE as it does in DCL.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer SCHEDULE.DAT                          | Insert | Forward

4 lines read from file WORKDISK:[USER]SCHEDULE.DAT
```

Enter the BOTTOM command to move the cursor to the end of the buffer. Press the DO key, type BOTTOM, and press RETURN.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]


Buffer SCHEDULE.DAT                    | Insert | Forward
Command:  BOTTOM
```

Press the Up arrow to move the cursor to the beginning of the fourth line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]


Buffer SCHEDULE.DAT                    | Insert | Forward
```

Press the Forward Reverse key to change the current buffer direction to reverse. Press the Move By Line key to move the cursor to the beginning of the third line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]


Buffer SCHEDULE.DAT                    | Insert | Reverse
```

Press the DO key, type the command LINE 1, and press RETURN to move the cursor to the beginning of the first line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]


Buffer SCHEDULE.DAT                    | Insert | Forward
Command:  LINE 1
```

1.3.2 Inserting Text

You can insert sections of text, entire files, and special nonprinting characters (such as control characters) into whichever buffer you are currently editing.

1.3 Editing Text

The following tables show the editing keys and EVE commands that you use while inserting text:

Editing Key	What It Does
Insert Overstrike	Changes the current editing mode as displayed on the highlighted status line. In insert mode, EVE inserts text at the current character position, moving existing text to accommodate the insertion. In overstrike mode, EVE overwrites text at the current position.
CTRL/A	Same as the Insert Overstrike key.
CTRL/V	Lets you insert nonprinting characters or control codes. You can search for special characters using the Find key. First, press the Find key, then press CTRL/V and the special character to be found, and activate the search by pressing RETURN. The QUOTE command is the same as the CTRL/V sequence.

Command	What It Does
INCLUDE FILE	Inserts the entire contents of the specified file into a buffer at the line before the current cursor location.

Before you begin inserting text into a buffer, look at the highlighted status line to determine whether EVE is in insert or overstrike mode. If EVE is in insert mode, text is inserted at the cursor position, and text that already appears in the file moves to accommodate your insertions. If EVE is in overstrike mode, text that you type at the keyboard is inserted at the cursor position, and the text that already appears in the file is overwritten as the cursor moves through it.

Press CTRL/A or the Insert Overstrike key to change from one mode to the other.

You can add text to your buffer in the following ways:

- Text—You can type characters at the keyboard. EVE adds the characters to the buffer at the current cursor position according to the current mode of the buffer (insert or overstrike).
- Files—You can add entire files by pressing the DO key and entering the EVE command INCLUDE FILE. Type the file specification at the *File to include:* prompt and press RETURN. EVE disregards the current mode (insert or overstrike) of the buffer and inserts the entire contents of the specified file into the buffer just before the line where the cursor currently appears.

  Wildcards are allowed in the file specification. If there is more than one match for a file specification with a wildcard, EVE displays a list of choices and prompts you to provide a more complete file specification. If the specified file does not exist, EVE displays a message stating that it could not include the file.

- Special Nonprinting Characters—You can add special nonprinting characters by pressing CTRL/V followed by the special character. For example, to insert an escape character into the buffer on a VT200-series or VT300-series terminal, press CTRL/V followed by CTRL/3. (On a VT100-series terminal, press CTRL/V and then press CTRL/[.) The special characters are added according to the current mode of the buffer (insert or overstrike).

The following example shows you how to insert text into a file, first in insert mode and then in overstrike mode. Invoke EVE to edit the existing file SCHEDULE.DAT.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer  SCHEDULE.DAT                          | Insert | Forward

4 lines read from file WORKDISK:[USER]SCHEDULE.DAT
```

Check the highlighted status line to ensure that EVE is in insert mode. Press the Insert Overstrike key (or CTRL/A) to change to insert mode if EVE is in overstrike mode. Move the cursor to the letter *s* in the word supervisor, type Engineering, and press the space bar.

The word Engineering is inserted in your text buffer, and the rest of the text on the line shifts to the right.

```
Schedule for 1 July
10:00 AM meeting with Engineering supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer  SCHEDULE.DAT                          | Insert | Forward

4 lines read from file WORKDISK:[USER]SCHEDULE.DAT
```

Now press the Insert Overstrike key to change to overstrike mode. Move the cursor to the letter *D* in the word Donna and type Andrea.

The word Andrea is placed in the buffer, overwriting the word Donna.

```
Schedule for 1 July
10:00 AM meeting with Engineering supervisor
Read and review memo from Andrea
Work on Pascal program
[End of file]

Buffer  SCHEDULE.DAT                          | Overstrike | Forward

4 lines read from file WORKDISK:[USER]SCHEDULE.DAT
```

1.3.3 Erasing and Restoring Text

With the EVE editor, you can easily delete text from a file or correct mistakes made during an editing session. If you erase text by mistake, you can restore the most recently erased text to its former location or, by moving the cursor, to another location. The following table shows the editing keys and EVE commands that erase and restore text:

Editing Key	What It Does
<X]	Erases character to the left of the cursor.
Erase Word	Erases current word or, if the cursor is not on a word, the next word.
CTRL/U	Erases all characters from the current cursor position to the beginning of the line.

Command	What It Does
ERASE CHARACTER	Erases the current character.
ERASE LINE	Erases from the current cursor position to the end of the current line, appending the next line to the end of the current line.
ERASE PREVIOUS WORD	Erases the previous word or the word the text cursor is on. If the cursor is at the start of a line, the carriage return at the end of the previous line is erased and the current line moves up. If the cursor is between words or on the first character of a word, the previous word is erased. If the cursor is in the middle of a word, all of that word is erased (same as ERASE WORD).
RESTORE	Restores, at the current cursor position, the word, sentence, or line that you have erased most recently with an EVE command or editing key. RESTORE does not restore single characters.
RESTORE CHARACTER	Restores, at the current cursor position, the character you have erased most recently with an EVE command or editing key.
RESTORE LINE	Restores, at the current cursor position, the line that you have erased most recently with an EVE command or editing key.
RESTORE WORD	Restores, at the current cursor position, the word that you have erased most recently with an EVE command or editing key.

To erase text from your buffer, move the cursor to the location of the text you want to erase, and press the appropriate editing key or type the appropriate EVE command.

The following example shows you how to erase and restore text. Invoke EVE to create the file RHYMES.DAT and enter the text shown in the following display:

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
```

Move the cursor to the letter *l* in the word *also*. Press the DO key, type the command ERASE LINE, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Commmand:  ERASE LINE
```

EVE erases all characters from the letter *l* to the end of the line and appends the next line to the current line.

```
She rhymes with tree,
a and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
```

Move the cursor to the letter *y* in the word *rhymes*. Press the DO key and enter the command ERASE WORD.

```
She rhymes with tree,
a and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Command:  ERASE WORD
```

EVE erases the word *rhymes* and shifts the remaining text to the left.

```
She with tree,
a and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
```

Move the cursor to the space between the letter *a* and the word *and* on the second line. Press the DO key and enter the command RESTORE LINE.

```
She with tree,
a and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Command:  RESTORE LINE
```

EVE restores the last line that was erased, in this case, *lso with bee,*.

Move the cursor to the letter *w* in the word *with* on the first line. Press the DO key and enter the command RESTORE WORD.

```
She with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                              | Insert | Forward
Command:  RESTORE WORD
```

EVE restores the last word that was erased, in this case, *rhymes*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                              | Insert | Forward
```

Section 1.3.4 describes the functions of the Select and Remove keys, which can be used together to erase text from a buffer.

1.3.4 Moving Text from One Location to Another

The following tables describe the functions of the Select, Remove, and Insert Here keys as well as the STORE TEXT command, which are used to erase text, to move text from one location to another within a buffer in "cut and paste" operations, and to duplicate text. For information on how to move text from one buffer to another, see Section 1.7.

Editing Key	What It Does
Select	Marks text (highlighting it in reverse video) from the initial cursor location to wherever you move the cursor. To cancel the selection, press the Select key again or use RESET.
Remove	Removes the text, which was marked with SELECT or highlighted by FIND, and places it in the Insert Here buffer.
Insert Here	Inserts the text from the Insert Here buffer at the current cursor location.

Command	What it Does
STORE TEXT	Copies text that was marked with SELECT, placing it in the Insert Here buffer. Text that is copied is not removed from its original position.

To erase text when the buffer is set in a forward direction, place the cursor on the first character you want to erase. Press the Select key, and then move the cursor to one character beyond the last character you want to erase. (In reverse direction, move the cursor to the last character, not one beyond.) The text to be erased is highlighted in reverse video. (If you decide not to remove text from the buffer, press the Select key again to cancel the selection.) Press

the Remove key. EVE deletes the highlighted text from your screen and places it in the Insert Here buffer.

You can insert the text at any cursor location by pressing the Insert Here key, or you can erase text permanently from your buffer by leaving it in the Insert Here buffer. You can insert the text contained in the Insert Here buffer any number of times at any cursor location until you select a new section of text and put that new text in the Insert Here buffer using the Remove key or the STORE TEXT command. The Insert Here buffer contains whatever text was last copied or removed.

The following example shows you how to erase and move text from one location to another using the Select, Remove, and Insert Here keys. Invoke EVE to edit the file RHYMES.DAT.

Move the cursor to the beginning of the second line of RHYMES.DAT and press the Select key.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                | Insert | Forward
Selection started. Press the Remove key when finished.
```

Press the Down arrow key once. The second line of text is highlighted. Press the Remove key. The second line of text is removed from the current buffer.

```
She rhymes with tree,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                | Insert | Forward
Remove completed.
```

Press RETURN twice, and then press the Insert Here key. The text in the Insert Here buffer is inserted at the current cursor location.

```
She rhymes with tree,


also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                | Insert | Forward
Remove completed.
```

The STORE TEXT command allows you to duplicate text in a file. Move the cursor to the first line of text and press the Select key. Press CTRL/E to move the cursor to the end of the first line and enter the STORE TEXT command. (Press DO, type STORE TEXT, and press RETURN.) The Insert Here buffer now contains a copy of the selected text. Now move the cursor to the line above *also with bee,* and press the Insert Here key.

```
She rhymes with tree,

She rhymes with tree,
also with bee,
and this one makes three.
[End of file]


Buffer RHYMES.DAT                                  | Insert | Forward

Copy completed.
```

1.3.5 Locating Text

Use the Find key to locate specific text in the editing buffer. Press the Find key (PF1 on VT100-series terminals). Then type the text you want to locate, which is called the *search string*, and press RETURN.

EVE attempts to move the cursor to the beginning of the specified string.

If the search string contains all lowercase letters, EVE disregards the case of letters and locates any occurrence of the string. Thus, *the*, *THe*, and *thE* all match the search string *the*. If the search string contains one or more uppercase letters, EVE locates only the occurrences of the string where the case of letters are exactly the same. Therefore, the only match for the search string *tHis* is tHis.

EVE is sensitive to diacritical (accent) marks and locates only those occurrences of the string where the diacritical marks are exactly the same. For example, in searching for ë, EVE does not locate occurrences of e, é, è, or ê.

The current direction of the buffer determines whether EVE first searches in a forward or reverse direction.

If the editor cannot find the string in the current direction but finds it in the opposite direction, EVE prompts you to change direction. To search in the opposite direction, type Y. EVE moves the cursor to the first occurrence of the string in the opposite direction. The current direction in the highlighted status line is not changed, however.

When EVE finds the search string, the editor highlights it and moves the cursor to the first letter of the string. (You can use any one of the following commands to modify a highlighted search string: REMOVE, FILL RANGE, STORE TEXT, LOWERCASE, UPPERCASE, and CAPITALIZE.) To cancel the highlighting, move the cursor off the search string or enter the RESET command.

If you press the Find key twice, EVE tries to find the next occurrence of the search string.

The following example uses the existing file RHYMES.DAT to illustrate the use of the Find key. When you invoke EVE to edit RHYMES.DAT, the cursor appears on the first letter of the first line of the buffer, and the current direction is forward.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

Press the Find key, type the letters *ree*, and press RETURN. The cursor moves to the letter *r* in the word tree and highlights the letters *ree*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
Forward Find: ree
```

Press Find twice to find the next occurrence of the string *ree*. The cursor moves to the letter *r* in the word three and highlights the letters *ree*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
Finding previous target: ree
```

When a search string is found and highlighted, you can use any command that works on a selected range. For example, enter the UPPERCASE command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
Command:  UPPERCASE
```

The UPPERCASE command changes the case of the highlighted letters from lowercase to uppercase.

```
She rhymes with tree,
also with bee,
and this one makes thREE.
[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
```

You can also use wildcards to search for a text string. EVE can search for text using either of two sets of patterns: VMS or ULTRIX. By default, EVE searches for text using the VMS wildcard patterns. The SHOW WILDCARD command displays wildcard patterns for VMS, which include the asterisk (*) and percent sign (%).

First, position the cursor at the beginning of the buffer. To search for text strings ending in ee, enter the command WILDCARD FIND *ee.

```
She rhymes with tree,
also with bee,
and this one makes thREE.
[End of file]


Buffer RHYMES.DAT                              | Insert | Forward
Command:  WILDCARD FIND *ee
```

EVE positions the cursor at the beginning of the line containing the *r* in *tree*.

The WILDCARD FIND command can also search for a string terminated by the Forward Reverse key. The key resets the initial direction of the search, overriding the current direction for the buffer.

You can specify how the FIND command treats the blank spaces between words such as spaces, tabs, and line breaks. By default, EVE treats the blank spaces, or *whitespace*, in a search string literally and does not match across a line break. However, if you use the SET FIND WHITESPACE command before the search, the FIND command ignores whitespace when searching for a text string.

1.3.6 Marking Locations in Text

The MARK and GO TO commands are very useful when you are editing a large file and know that you want to return to a specific cursor location later in the editing session. The following table describes the MARK and GO TO commands:

Command	What It Does
MARK	Associates a unique and invisible label, consisting of one or more alphanumeric characters, with the current cursor location. The mark exists for the rest of an editing session.
GO TO	Returns the cursor to the location labeled by the MARK command. If the labeled location is contained in another buffer, EVE moves the cursor to the other buffer and places the buffer in the current window.

To mark a cursor location, press the DO key, type MARK label-name, and press RETURN. The label name can be one or more printable characters, including alphanumeric and punctuation characters. To return the cursor to the marked location, press the DO key, type GO TO label-name, and press RETURN.

The following example shows you how to use the MARK and GO TO commands to mark a cursor position with the label name FIRST and how to return to that cursor position.

Move the cursor to the letter *b* in the word bee. Press the DO key, type MARK, and press RETURN. To mark the cursor location with the label FIRST, type FIRST at the *Mark name:* prompt.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Mark name:  FIRST
```

Move the cursor to the letter *t* of the word three.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Current position marked as FIRST
```

Press the DO key, type GO TO FIRST, and press RETURN to return the cursor to the position labeled FIRST.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Going to mark: FIRST
```

1.3.7 Replacing Text

The REPLACE command allows you to replace a text string in the current buffer with another text string. This is especially useful if you have spelled a word incorrectly throughout a long file and you want to fix every occurrence of the misspelled word.

To use the REPLACE command, press the DO key, type REPLACE, and press RETURN. Type the string that you want to replace at the *Old string:* prompt and press RETURN. Type the new string at the *New string:* prompt and press RETURN.

If EVE finds the old string in the current direction, it moves the cursor to the first occurrence of the old string, highlights the string, and provides the following prompt: *Replace? Type yes, no, all, last, or quit:*.

If EVE does not find the string in the current direction but finds it in the opposite direction, EVE provides the following prompt: *Found in 'reverse/forward' direction. Go there? [Y]*. Respond to the prompt by typing a single-character abbreviation of the response and pressing RETURN. (The single-character abbreviation within brackets indicates that simply pressing RETURN is equivalent to typing Y and pressing RETURN.) If you type Y or press RETURN, EVE moves the cursor to the first occurrence of the string in the new current direction, highlights the string, and provides the following prompt: *Replace? Type yes, no, all, last, or quit:*. The current direction of the buffer in the highlighted status line is not changed.

The following table explains the possible responses and actions when you type these responses:

Response	Replacement Procedure
Yes	Replaces the string and attempts to locate another occurrence of the string in the current direction. If found, the cursor moves to the next occurrence of the string, the string is highlighted, and EVE prompts: *Replace? Type yes, no, all, last, or quit:*. If the string is not found in the current direction but is found in the opposite direction, EVE prompts: *Found in 'reverse/forward' direction. Go there? [Y].*
No	Skips the occurrence and searches for another occurrence of the string in the current direction. If found, the cursor moves to the next occurrence of the string, the string is highlighted, and EVE prompts: *Replace? Type yes, no, all, last, or quit:*. If the string is not found in the current direction but is found in the opposite direction, EVE prompts: *Found in 'reverse/forward' direction. Go there? [Y].*
All	Replaces the string and all other occurrences of the string in the current direction. EVE leaves the cursor at the position where the first replacement occurred. After all occurrences of the string in the current direction have been replaced, EVE may inform you: *Found in 'forward/reverse' direction. Go there? [Y].* If you type yes, EVE replaces all occurrences of the string in the new direction.
Last	Replaces this occurrence of the string and stops the REPLACE procedure; the cursor does not move.
Quit	Does not replace this occurrence of the string and stops the REPLACE procedure; the cursor does not move. Pressing CTRL/Z has the same effect.

The REPLACE command is case sensitive. If the old string and the new string are given in all lowercase characters, then EVE matches the case appropriately for each replacement. For example, in replacing the string *parsley* with the string *dill*, EVE replaces a capitalized version of parsley with a capitalized version of dill. If the old string is lowercase, the search is general. However, if the old string is uppercase or mixed case, the search is case exact. If the old and the new string are lowercase, the replacement mirrors the case of the occurrence. Whereas, if the new string is uppercase or mixed case, the replacement is exact.

After finding the old string and prompting for the replacement of all occurrences of the string in both directions, EVE continues to search the buffer for the string. If EVE finds more occurrences, it informs you: *Found in forward/reverse direction. (May have already been replaced.) Go there [N]?* The default answer is No to prevent you from replacing the occurrences a second time.

The following example shows you how to use the REPLACE command to replace every occurrence of the string *ee* with the string *oo*. Move the cursor to the top of the buffer. Press the DO key, type REPLACE, and press RETURN. Type *ee* at the highlighted *Old string:* prompt, press RETURN, and type *oo* at the highlighted *New string:* prompt.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
Replace? Type yes, no, all, last, or quit:
```

The cursor moves to the highlighted string *ee* in the word tree. Type *all*, and press RETURN. All occurrences of the string *ee* are replaced with the string *oo*.

```
She rhymes with troo,
also with boo,
and this one makes throo.
[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
Replace? Type yes, no, all, last, or quit: all
```

1.4 Using the HELP Facility

EVE has an online HELP Facility that supplies information on editing commands and keys without disturbing your work. You can obtain help by entering the HELP command or by pressing the Help key.

To view a list of EVE commands, press the DO key and enter the command HELP. Use the Previous Screen and Next Screen keys (up and down arrow keys on a VT100-series terminal) to scroll through the entire list of EVE commands. To obtain information on a particular command, type a command name after the help prompt and press RETURN. The help text appears on the screen.

If you know the name of a specific command for which you need help, press the DO key, type HELP followed by the name of the command, and press RETURN. The help text for that command appears on the screen. For example, to receive help on the MOVE BY LINE command, enter the command HELP MOVE BY LINE. The following help text appears on your screen:

```
MOVE BY LINE

   Moves the text cursor a line at a time in the current direction (shown in
   the status line).

   o  In FORWARD direction, the cursor moves to the end of the current line
      or if already there, to the end of the next line (if any).

   o  In REVERSE direction, the cursor moves to the start of the current line
      or if already there, to the start of the previous line (if any).

   Keys:  By default, key F12 is defined as MOVE BY LINE.  The VT100 keypad
          defines MINUS on the keypad.

   Related topics:

       END OF LINE   LINE    START OF LINE   WHAT LINE


 Buffer HELP
Type topic name or ? for list. Press Return if done:
```

The HELP Facility also provides information on general topics. For example, if you elect to use the EDT or the WPS keypad, use the command HELP SET KEYPAD EDT or the command HELP SET KEYPAD WPS to obtain information on changing your default keypad. To display a list of all defined keys for the keypad you are using, enter the command HELP KEYS. There is also help on the difference between the EDT and WPS keypads within EVE and the original EDT and WPS keypads (HELP EDT DIFFERENCES or HELP WPS DIFFERENCES).

The Help key produces a keypad diagram for the keypad you are using. The diagram shows both the default editing keys and the keys you have defined for the minikeypad (LK201 keyboard), the main keypad, the keys F10-F14 (LK201 keyboard), and the GOLD key (described in Section 1.9.3).

After you display the keypad, you can obtain help on particular editing keys by pressing the desired key. If you press a key to which you have assigned an EVE command, EVE provides the help text for that command.

1.5 Recovering from System Interruptions

EVE has recovery procedures for two types of system interruptions. You can remove extraneous characters that appear on your screen, and you can recover a "lost" editing session with the journaling facility.

1.5.1 Refreshing the Screen

If extraneous characters, such as an operator message, appear on your terminal screen while you are editing or inserting text, press CTRL/W to refresh the screen. The screen becomes blank, and then all characters are redrawn, minus any extraneous characters.

1.5.2 Using the Journal File

If you are editing a file and a system interruption occurs (that is, a break in communication between your terminal and the computer), you can recover your "lost" editing session. By default, EVE records every keystroke you enter during an editing session in a journal file that has the same file name as the file you are editing and a file type of TJL.

Typically, an editing session ends without interruption, so the system deletes the journal file. When you experience a system interruption, however, the journal file is saved. EVE can use the journal file to reconstruct your editing session so that only the last few keystrokes of your editing session are lost.

To recover an editing session, enter the DCL command you used to invoke EVE plus the /RECOVER qualifier. For example, to recover an editing session you began with the command EDIT/TPU LETTER.RNO, type the following command and file name and press RETURN:

```
$ EDIT/TPU/RECOVER LETTER.RNO
```

You must recover an editing session at a terminal of the same type as the one you used for your editing session. When EVE finishes recovering the session, check to be sure that the last few keystrokes of your editing session were recovered and continue editing the file. If another system interruption occurs before you exit, a journal file containing the keystrokes from both editing sessions is saved.

The journal file is saved in the current default editing directory. However, you can create a journal file in another directory using the /JOURNAL qualifier, as in the following DCL command:

```
$ EDIT/TPU/JOURNAL=[ALEXIS.JOURNEYS]LETTER.TJL LETTER.RNO
```

If you use the /JOURNAL qualifier to create a journal file with a different file name or a different directory, then you must use the /JOURNAL qualifier and the file name when you recover the file. For example, to recover the file LETTER.RNO when the journal file is in directory [ALEXIS.JOURNEYS], enter the following command:

```
$ EDIT/TPU/JOURNAL=[ALEXIS.JOURNEYS]LETTER.TJL/RECOVER LETTER.RNO
```

The journaling facility does have two restrictions.

First, if you use the WRITE FILE command during your editing session to copy the contents of the buffer to another file, you must specify the original version number in order to recover your file. For example, if you are editing an existing file called LETTER.RNO;1 and use the WRITE FILE command, EVE creates LETTER.RNO;2. If you experience a system interruption, you must enter the original version of LETTER.RNO on the EDIT/TPU /RECOVER command line. In this example it would be LETTER.RNO;1. See Section 1.7.4 for more information on the WRITE FILE command.

Second, EVE is usually unable to recover the keystrokes entered after you press CTRL/C. If you press CTRL/C, immediately EXIT from the editing session and invoke EVE again.

1.6 Formatting Text

EVE provides commands that enable you to format your text by setting margins, tabs, screen width, and word wrap. It allows you to center lines, take extra whitespace out of text, and insert page breaks. The following table lists EVE text formatting commands and describes their functions:

Command	What It Does
CAPITALIZE WORD	Capitalizes a single word or each word in the text highlighted by FIND or SELECT.
CENTER LINE	Centers the line of text marked by the cursor between the current left and right margins. The cursor moves with the line, remaining on the same character as the line moves.
FILL	Reformats the current paragraph or selected range according to the margins of the buffer, so the maximum number of words fits on a line.
FILL PARAGRAPH	Reformats the paragraph the cursor identifies according to the margins set for the buffer.
FILL RANGE	Reformats the currently selected range of text (or the current FIND range) according to the current margin settings.

Command	What It Does
INSERT PAGE BREAK	Inserts a form feed character at the current editing position to mark the beginning of a new page. A page break appears as a small double-F and is always on a line by itself. By default, CTRL/L inserts a page break.
LOWERCASE WORD	Changes the currently selected text or the word the cursor is on to lowercase.
SET LEFT MARGIN	Sets the left margin in current buffer. The left margin must be greater than 0 but less than the right margin. By default, the left margin is 1.
SET RIGHT MARGIN	Sets the right margin for the current buffer. The right margin must be greater than the left margin. By default, the right margin is one less than the width. The width is typically 80, so the default margin is typically 79.
SET TABS AT	Sets tab stops at the columns that you specify. Columns are specified as a sequence of positive integers separated by spaces. By default, tab stops are set in every eighth column. This command does not affect the hardware tab settings of your terminal.
SET TABS EVERY	Sets tab stops at the specified interval. By default, tab stops are set in every eighth column. This command does not affect the hardware tab settings of your terminal.
SET TABS INSERT	Turns on tab insert mode, so that EVE sets a tab stop at the column where you press the Tab key, moving over text currently on the screen. SET TABS INSERT is the default mode.
SET TABS SPACES	Changes the tab mode to insert an appropriate number of spaces, rather than a tab character, when the Tab key is pressed. Previously existing tab characters are not affected.
SET TABS MOVEMENT	Changes the tab mode so the Tab key becomes a cursor-control key. Pressing the Tab key moves the cursor to the next tab stop but does not insert a tab character.
SET TABS VISIBLE	Displays tabs as visible characters on the screen.
SET TABS INVISIBLE	Does not display tabs as visible characters on the screen. SET TABS INVISIBLE is the default mode.
SET WIDTH	Sets the width of lines displayed on the screen. Specify width as a positive integer *n*. By default, the screen width is your terminal setting. (It is typically 80 columns.) If *n* is greater than 80, EVE sets the terminal to 132-column mode for the current editing session. When EVE is terminated, the terminal is restored to the default setting. Setting the width changes the display of text in all windows.

Command	What It Does
SET WRAP	Enables word wrapping at the right margin of the buffer. EVE starts new lines without you pressing RETURN or using the FILL command. SET WORD WRAP is the default setting.
SET NOWRAP	Disables word wrapping at the right margin of the buffer. You must start new lines by pressing RETURN or by using the FILL command.
SHIFT LEFT	Moves the current window to the left a specified number of columns. The SHIFT LEFT command can be used only to reverse the effect of the SHIFT RIGHT command.
SHIFT RIGHT	Moves the current window horizontally to the right a specified number of columns, allowing you to view columns of characters that do not currently appear on the terminal screen.
UPPERCASE WORD	Changes the currently selected text or the word the cursor is on to uppercase.

The following example shows you how to use EVE commands to set margins and screen width and to shift the current window. Invoke EVE to edit the existing file RHYMES.DAT. Press the DO key, type SET LEFT MARGIN 20, and press RETURN to set a left margin of 20. The text that already appears in the buffer does not change.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Command:  SET LEFT MARGIN 20
```

Move the cursor to the end of the buffer and type the following new text: *Also with thee, and me.* Then press RETURN. The new text that you enter is inserted at the left margin of 20.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and me.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Left margin set to: 20
```

Reset the left margin to 1. Press the DO key, type SET LEFT MARGIN 1, and press RETURN. Again, the text that already appears in the buffer does not change. When you insert new text, it is inserted at a left margin of 1.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Command:  SET LEFT MARGIN 1
```

Now, set the right margin to 30 by pressing the DO key, typing SET RIGHT MARGIN 30, and pressing RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
Command:  SET RIGHT MARGIN 30
```

Enter the following new text in your file and notice that it wraps automatically to the next line at a right margin of 30.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward

Right margin set to: 30
```

To reset the right margin to 79, press the DO key, type SET RIGHT MARGIN 79, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward

Right margin set to: 79
```

Now, set the width of your text window to 20 by pressing the DO key, typing SET WIDTH 20, and pressing RETURN.

```
She rhymes with tre
also with bee,
and this one makes
And free, and fee,
and brie, and any n
other words.

[End of file]

Buffer RHYMES.DAT
```

The appearance of the current text window changes; all text beyond the twentieth column disappears from the screen. Press the DO key, type SHIFT RIGHT 5, and press RETURN to view five columns of text beyond the right boundary of the window.

The window shifts five columns to the right, and you can see characters that were not visible before the shift operation.

```
hymes with tree,
with bee,
his one makes three.
            Also
ree, and fee, and s
rie, and any number
 words.

[End of file]

Buffer RHYMES.DAT

Window now shifted
```

Shift the window to its original location by pressing the DO key, typing SHIFT LEFT 5, and pressing RETURN.

```
She rhymes with tre
also with bee,
and this one makes
And free, and fee,
and brie, and any n
other words.

[End of file]

Buffer RHYMES.DAT

Window now shifted
```

Set the screen width to 80 by pressing the DO key, typing SET WIDTH 80, and pressing RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
```

EVE sets the width of lines on your screen to 80.

The following example shows how to fill a selected range of text and how to fill a paragraph. Using the existing file RHYMES.DAT, set the left margin to 5 and the right margin to 55.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and
                  me.
And free, and fee, and
see, and brie, and any
number of other words.

[End of file]

Buffer RHYMES.DAT                              | Insert | Forward
Command: SET RIGHT 55
```

Now, fill a range of text by selecting the first three lines of text (with the Select key) and entering the FILL command. Press the DO key, type the command FILL, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and
                  me.
And free, and fee, and
see, and brie, and any
number of other words.

[End of file]

Buffer RHYMES.DAT                              | Insert | Forward
Command:  FILL
```

EVE fills the highlighted text between the left margin of 5 and the right margin of 55.

```
    She rhymes with tree, also with bee, and this one
    makes three.
                  Also with thee, and
                  me.
And free, and fee, and see,
and brie, and any number of
other words.

[End of file]

Buffer RHYMES.DAT                              | Insert | Forward
```

For EVE, a paragraph is bound by blank lines, the top or bottom of the buffer, or page breaks. To compress text in a paragraph, put the cursor anywhere in the text of a paragraph, press the DO key, type the command FILL PARAGRAPH, and press RETURN.

```
   She rhymes with tree, also with bee, and this one
   makes three.
                  Also with thee, and
                  me.
And free, and fee, and
see, and brie, and any
number of other words.

[End of file]

Buffer RHYMES.DAT                                | Insert | Forward
Command:  FILL PARAGRAPH
```

EVE fills the the paragraph according to the margins set for the buffer, in this case 5 and 55.

```
    She rhymes with tree, also with bee, and this one
    makes three. Also with thee, and me. And free, and
    fee, and see, and brie, and any number of other
    words.

[End of file]

Buffer RHYMES.DAT                                | Insert | Forward
```

To center a line of text, put the cursor anywhere on the line you want to center. For example, to center the text *words* in the last line of RHYMES.DAT, put the cursor on the *w*. Press the DO key, type the command CENTER LINE, and press RETURN.

```
    She rhymes with tree, also with bee, and this one
    makes three. Also with thee, and me. And free, and
    fee, and see, and brie, and any number of other
    words.

[End of file]

Buffer RHYMES.DAT                                | Insert | Forward
Command: CENTER LINE
```

The command centers the line of text between the current left and right margins; in this example, the left margin is set at 1 and the right margin is set at 55. The cursor moves with the line, remaining on the character w as the line is centered.

```
    She rhymes with tree, also with bee, and this one
    makes three. Also with thee, and me. And free, and
    fee, and see, and brie, and any number of other
                        words.

[End of file]

Buffer RHYMES.DAT                                | Insert | Forward
```

The EVE commands to change the case of text—CAPITALIZE, UPPERCASE, and LOWERCASE—work either on a selected range of text or on the word on which the cursor rests.

To change the case of the first line of text, move the cursor to the word *she* and press the Select key. To mark the end of the selection, move the cursor to the end of the line, after *one*, press the DO key, and enter the command UPPERCASE.

```
SHE RHYMES WITH TREE, ALSO WITH BEE, AND THIS ONE
makes three. Also with thee, and me. And free, and
fee, and see, and brie, and any number of other
                    words.

[End of file]

Buffer RHYMES.DAT                          | Insert | Forward
```

To change the case of a particular word, position the cursor on the word, press the DO key, and enter the appropriate case-changing command. Once EVE changes the case of the word, the cursor moves to the next word or to the end of the line. If the cursor is between two words, the case of the word to the right of the cursor is changed and the cursor moves to the end of the word.

1.7 Using Buffers

Buffers are storage areas that exist only during an editing session. The following table describes the EVE commands used to create, manipulate, and delete buffers:

Command	What It Does
BUFFER	Puts the specified buffer in the current window and moves the cursor to the last location it occupied in that buffer. Creates a new buffer if the specified buffer does not exist.
DELETE BUFFER	Deletes the buffer you specify.
GET FILE	Creates a new buffer that contains the text of the specified file (or an empty buffer if you specify a file that does not exist); places the new buffer in the current window; and places the cursor at the beginning of the new buffer. If you specify the same file again during an editing session, GET FILE simply places the buffer in the current window. If you specify the same file name and file type with a different device or directory name during an editing session, EVE prompts you for a different buffer name into which to read the file.
GO TO	Returns the cursor to the location labeled by the MARK command. If the labeled location is contained in another buffer, EVE moves the cursor to the other buffer and places the buffer in the current window. (Section 1.7.3 explains how to use multiple buffers in an editing session.)
SHOW	Displays information about the buffers you have created during the editing session. If more than one buffer is active in your editing session, EVE displays information about the buffer you are currently editing. For information on other buffers, press the DO key. To resume editing, press any other key.

Command	What It Does
SHOW BUFFERS	Lists the buffers you have created during an editing session. You can move the cursor through the list and specify a particular buffer for viewing using the Select key.
SHOW SYSTEM BUFFERS	Lists the system buffers created by EVE. You can move the cursor through the list and specify a buffer for viewing using the Select key.
WRITE FILE	Writes the contents of the current buffer to a file. If you do not specify a file name, EVE uses the buffer name as the file name. If you created the current buffer with the BUFFER command, EVE prompts you for a file specification to which it writes the file.

When you invoke EVE to edit an existing file, EVE reads the contents of the file into a buffer. The highlighted status line contains the buffer name, editing mode, and current direction of the buffer.

To display more information about the current buffer, enter the SHOW command. The information displayed includes the buffer name, the name of the input and output files, whether the buffer has been modified, the current mode and direction, the number of lines, the margin and screen-width settings, the tab-stop settings, and the marks that have been defined in the buffer. If more than one buffer is active during an editing session, EVE prompts you to press the DO key to receive information about the other buffers.

To delete a buffer, enter the DELETE BUFFER command, specifying the name of the buffer you want to delete. For example, the command DELETE BUFFER MYFILE.TXT deletes the buffer called MYFILE.TXT. The buffer name must be typed in full; no abbreviations are allowed.

If the buffer (in this example, MYFILE.TXT) has been modified, EVE issues the following prompt to confirm that you want to delete the buffer: *That's a modified buffer. Type delete_only, write_first, or quit:.*

If you are viewing a buffer that you want to delete, EVE replaces the buffer with the oldest buffer existing in the editing session.

1.7.1 Listing Buffers

To display a list of all the buffers you have created during an editing session, enter the SHOW BUFFERS command. You can scroll through the list and specify the buffer you want to view or any buffers you want to delete. To display a buffer in your current window, move the cursor to the buffer name and press the Select key. To delete a buffer, move the cursor to the buffer name and press the Remove key.

These applications of the Select and Remove keys apply only when you are viewing a list of buffers.

To display a list of all buffers that EVE has created, enter the SHOW SYSTEM BUFFERS command. You can scroll through the list and specify the buffer you want to view by moving the cursor to the buffer name and pressing the Select key. EVE puts the buffer in your current window.

Note: Do not delete system buffers because these buffers are necessary for some commands to work properly.

1.7.2 Displaying the Contents of the Messages Buffer

EVE uses the Messages window, which appears at the bottom of the screen, to communicate error and informational messages during an editing session. The Messages window displays the last message in the Messages buffer.

You can display these messages with the BUFFER command. To display the contents of the Messages buffer, enter the command BUFFER MESSAGES. In order to return to the buffer you were editing, enter the BUFFER command followed by the name of the appropriate buffer. For example, to return to the buffer named RHYMES.DAT, enter the command BUFFER RHYMES.DAT. Alternately, you can enter the SHOW BUFFERS command to display the buffers you have created and use the Select key to specify the appropriate buffer.

1.7.3 Editing Two Buffers

During an editing session you can use several buffers if you want to edit more than one file or if you want temporary storage areas for manipulating blocks of text. Multiple buffers are especially useful if you want to copy text from one file to another.

To create a new buffer, enter the GET FILE command and the name of the file you want to copy to the new buffer. You can use the asterisk wildcard character (*) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (%) as a substitute for one character in the file name and file type, and you can use the ellipsis wildcard ([. . .]) as a substitute for a directory specification.

If the specified file exists, EVE reads the contents of the file into a new buffer and displays the buffer in the current window. If there is more than one match for a file specification with a wildcard, EVE displays a list of choices and prompts you to provide a more complete file specification. Otherwise, EVE creates an empty buffer and displays the buffer in the current window.

To change the buffer in the current window, press the DO key, type BUFFER and the name of the buffer you want to display on the screen, and press RETURN. If you forget a buffer name, enter the SHOW BUFFERS command to display the names of active buffers in your editing session and specify a buffer with the Select key.

The following example shows how to use two buffers to edit two files during an EVE editing session. This example assumes that the original versions of files RHYMES.DAT and SCHEDULE.DAT exist in your current default directory. Invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

 Buffer RHYMES.DAT                              | Insert | Forward
 3 lines read from WORKDISK:[USER]RHYMES.DAT
```

Press the DO key, type the command GET FILE SCHEDULE.DAT, and press RETURN to create a new buffer that contains the most recent version of SCHEDULE.DAT.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]


Buffer SCHEDULE.DAT                          | Insert | Forward
```

Now that you have two buffers, practice copying from one buffer to another. First work with buffer SCHEDULE.DAT.

Place the cursor on the letter *R* in the word *Read* and press the Select key. Press the Down arrow once and the third line of SCHEDULE.DAT is highlighted. Press the Remove key to place the selected line in the Insert Here buffer.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]


Buffer SCHEDULE.DAT                          | Insert | Forward

Selection started. Press Remove when finished.
```

The text from SCHEDULE.DAT remains in the Insert Here buffer until you overwrite it with other selected text. Now switch to the other buffer to resume editing RHYMES.DAT. Press the DO key, type the command BUFFER RHYMES.DAT, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                            | Insert | Forward
```

Position the cursor at the top of the buffer and press the Insert Here key. The text from the Insert Here buffer that was removed from the buffer SCHEDULE.DAT is inserted in the top of the buffer RHYMES.DAT.

```
Read and review memo from Donna
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                            | Insert | Forward
```

If you exit from an editing session in which you have modified multiple buffers, EVE writes the contents of the current buffer to a file and then asks you whether you want to write each of the other modified buffers to files.

1.7.4 Reading and Writing Files

There are three ways to read a file into an EVE buffer.

- Invoke EVE with a file specification.
- Enter the INCLUDE FILE command and the name of the file you want to include. EVE reads the entire contents of a file into a buffer just before the line where the cursor is located. Using the INCLUDE FILE command does not change the name of the buffer on the status line.
- Enter the GET FILE command and the name of the file you want to use. This command creates a new buffer and reads the contents of an existing file into the buffer. The name of the buffer on the status line is the same as the file name you specified with the GET FILE command. (See Section 1.7.3.)

To write the contents of the current buffer to a file, enter the WRITE FILE command. You can include a file specification with the WRITE FILE command. If you do not include a file specification, EVE writes the file using the input file specification. If you created the current buffer with the BUFFER command, EVE prompts you for a file specification to which it writes the file.

If you have used the WRITE FILE command in an editing session and you experience a system interruption, see Section 1.5 for information on recovering the editing session.

1.8 Using Windows

During an EVE editing session, the name of the text buffer you are editing is displayed on the screen in a window. A highlighted status line appears at the bottom of a window identifying the name, current editing mode, and current direction of the buffer.

EVE allows you to view more than one window on your terminal screen at the same time. For example, you can have two windows in order to view and edit different sections of the same buffer.

The following table describes the EVE commands used to create and manipulate windows:

Command	What It Does in a Window Environment
SPLIT WINDOW	Splits the window that the cursor is in, forming two smaller windows. Adding an argument to the command allows you to divide the window into more than two parts. For example, SPLIT WINDOW 3 splits the window into 3 windows.
TWO WINDOWS	Synonymous with the SPLIT WINDOW 2 command.
NEXT WINDOW	Puts the text cursor in the next (or other) window.
PREVIOUS WINDOW	Puts the text cursor in the previous (or other) window.
OTHER WINDOW	Synonymous with NEXT WINDOW command.
ONE WINDOW	Restores the current window as a single, large window.

Command	What It Does in a Window Environment
DELETE WINDOW	Deletes the window the text cursor is in, assuming you are using more than one window.
ENLARGE WINDOW	Enlarges the current window by a specified number of lines. For example, ENLARGE WINDOW 5 enlarges the window the text cursor is in by 5 lines. The adjacent window shrinks accordingly.
SHRINK WINDOW	Shrinks the current window by a specified number of lines. For example, SHRINK WINDOW 5 shrinks the window the text cursor is in by 5 lines. The adjacent window is enlarged accordingly.

1.8.1 Editing One Buffer

To view or edit two sections of a file at the same time, use the SPLIT WINDOW command. EVE splits your screen and creates two identical windows. The cursor maintains its position in the buffer but appears only in the bottom window. Notice that the buffer name in each of the status lines is the same.

Now you can edit different sections of a single file. Any edits that you make in one window are made simultaneously in the other window. Unless you are viewing two different sections of a file, you can see EVE incorporate edits simultaneously in the two windows.

Displaying two sections of a long file makes moving text within a file very efficient. You can select and remove text from one part of the file and insert it into the other. To move the cursor from one window to the other, enter the NEXT WINDOW command.

To remove the second window from the screen and expand the current window to occupy the whole editing area, press the DO key, type ONE WINDOW, and press RETURN.

1.8.2 Editing Two Buffers

The following steps describe how to edit two buffers containing different files:

1. Invoke EVE to edit a file.
2. Create two windows on your screen by entering the SPLIT WINDOW command. EVE splits your screen and creates two windows. The cursor maintains its position in the buffer but appears only in the bottom window. Notice that the buffer name in each of the highlighted status lines is the same.
3. Place a second file in the current window using the GET FILE or BUFFER command.

   Use the GET FILE command with a file specification to create a new buffer in the current window.

   To display in the current window a buffer that you created earlier in the editing session, enter the BUFFER command and the name of the buffer you want to display.

4 Your terminal screen now displays two different buffers. You can select and remove text from one buffer and insert it into the other buffer. To move the cursor from one window to the other, enter the command NEXT WINDOW.

The following example shows you how to edit two files and move text from one file to another using two windows. First, invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward

3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

Press the DO key, type the command SPLIT WINDOW, and press RETURN to create two windows on your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward

She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward
```

Press the DO key, type the command GET FILE SCHEDULE.DAT, and press RETURN to create a new buffer containing the text of SCHEDULE.DAT in the bottom window of your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                                  | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer SCHEDULE.DAT                                | Insert | Forward

4 lines read from file WORKDISK:[USER]SCHEDULE.DAT
```

While still in the bottom window, move the cursor to the letter *R* in the word *Read*. Press the Select key, then press the Down arrow twice. The last two lines in SCHEDULE.DAT are highlighted.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer SCHEDULE.DAT                              | Insert | Forward
Move the text cursor to select text.
```

Press the Remove key to place the highlighted text in the Insert Here buffer. Now enter the NEXT WINDOW command to move the cursor to the other window.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]

Buffer SCHEDULE.DAT                              | Insert | Forward
Remove completed.
```

Move the cursor to the bottom of buffer RHYMES.DAT and press the Insert Here key. The text that you removed from SCHEDULE.DAT is inserted into RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
Read and review memo from Donna
Work on Pascal program
[End of file]

 Buffer RHYMES.DAT                               | Insert | Forward
```

```
Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]


Buffer SCHEDULE.DAT                          | Insert | Forward
```

Entering the ONE WINDOW command removes all other windows from the screen, expanding the window containing the cursor to occupy the whole editing area of the screen. Press the DO key, type ONE WINDOW, and press RETURN.

If you exit from the editing session, EVE writes the contents of the current buffer to a file and asks you if you want to write the contents of the other buffer to a file.

1.9 Defining Keys

You can define keys to execute EVE commands or to enter a series of keystrokes called a learn sequence.

EVE does not allow you to define the RETURN key (CTRL/M), the space bar, or any printing characters (such as letters, digits, and punctuation marks) on the main keyboard. In addition, DIGITAL recommends that you do not define the following keys and control key sequences:

DELETE <X]
F6 (VT200- and VT300-series)
Help (PF2 on VT100-series)
CTRL/C
CTRL/I (Tab key)
CTRL/Q
CTRL/R
CTRL/S
CTRL/T
CTRL/U
CTRL/X
CTRL/Y

You can define all other keys, including control keys. You can redefine the DO key, as long as you assign the DO command to another key.

1.9.1 Defining Keys to Execute an EVE Command

The DEFINE KEY command assigns an EVE command to a single key or control key sequence. You can, in effect, create your own editing keys to enter EVE commands that you use frequently. If you are using HELP and press a key to which you have assigned an EVE command, EVE provides the help text for that command. Key definitions are discarded when you terminate an EVE editing session, unless you use the SAVE EXTENDED EVE command (see Section 1.12.2) to save key definitions from one editing session to the next.

To define a key, do the following:

1 Press the DO key, enter the command DEFINE KEY, and press RETURN.

2 Type the EVE command that you want to assign to a key and press RETURN.

3 Press the key to be associated with the EVE command.

The message *Key defined* appears if you have successfully defined a key.

Another way to assign EVE commands to keys is to create an initialization file. The initialization file contains EVE commands that EVE executes when you invoke the editor. Each command line in the initialization file should contain a DEFINE KEY command. The command syntax is as follows:

```
DEFINE KEY [=key-name] command
```

The first parameter is the key to be defined; the second parameter is the command to assign to the key. For example, the following command line assigns the MOVE BY WORD command to keypad key 1:

```
Command: DEFINE KEY=KP1 MOVE BY WORD
```

The following command assigns the FILL command to CTRL/F:

```
Command: DEFINE KEY=CTRL/F FILL
```

You can use three different separators when specifying key names: an underscore, a dash, or a slash. For example, the CTRL/F key can appear as CTRL_F, CTRL-F, or CTRL/F.

To remove a key definition, use the UNDEFINE KEY command.

Section 1.9.3 contains more examples of defining keys to execute EVE commands.

1.9.2 Defining Keys to Enter a Learn Sequence

The LEARN command assigns a sequence of keystrokes, called a *learn sequence*, to a single key or control key. Learn sequences allow you to enter the same series of keystrokes in a buffer any number of times, simply by pressing one key. If you press a key to which you have assigned a learn sequence, EVE provides a HELP message stating that you have defined the key; the HELP diagram labels as *sequence* any keys to which you have assigned a learn sequence. All learn sequences are discarded when you terminate an EVE editing session unless you use the SAVE EXTENDED EVE command (see Section 1.12.2) to save them from one editing session to the next.

Define a learn sequence as follows:

1 Press the DO key, enter the LEARN command, and press RETURN.

2 Type the keystrokes to be remembered.

3 Press CTRL/R followed by the key to be associated with the learn sequence.

The message *Key sequence remembered* appears if you have successfully defined a key.

The following example shows you how to define a learn sequence that inserts a string of text into your file when you press CTRL/F. Invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.

[End of file]

Buffer RHYMES.DAT                              | Insert | Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

First, move to the end of the buffer. To begin the definition of the learn sequence, enter the LEARN command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

 Buffer RHYMES.DAT                             | Insert | Forward
Command:  LEARN
```

Insert the following text, which EVE is to remember, at the end of your file: *And what is a rhyme?*

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
 Buffer RHYMES.DAT                             | Insert | Forward
Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.
```

Press CTRL/R.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
Buffer RHYMES.DAT                              | Insert | Forward
Press the key that you want to use to see what was just learned:
```

Press CTRL/F, the key to which you are assigning the learn sequence.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
Buffer RHYMES.DAT                              | Insert | Forward
Key sequence remembered
```

For the rest of the editing session, simply press CTRL/F and EVE inserts the text *And what is a rhyme?* wherever the cursor is positioned at the time.

1.9.3 Defining a GOLD Key

You can assign two definitions to the same editing key if you create a *GOLD* key. One editing function is performed by pressing the editing key. The other function is performed by first pressing the GOLD key and then pressing the editing key. To define a GOLD key, enter the SET GOLD KEY command and press the key you want to use as the GOLD key. Once defined, the message *GOLD key set.* appears in the Messages buffer.

Once you have defined a GOLD key, you can use the GOLD editing keys predefined by EVE. To see a diagram of these commands, enter the command HELP KEYPAD. The GOLD editing keys appear in the display in reverse video.

You can also create your own key definitions using the GOLD key. The following example demonstrates how to define a GOLD key and assign two commands to a single key. The example defines the number 4 key on the numeric keypad as the GOLD key and then assigns the BOTTOM and TOP commands to the CTRL/G key. Thus, pressing CTRL/G alone enters the BOTTOM command, and pressing the GOLD key followed by CTRL/G enters the TOP command.

Invoke EVE to edit the file RHYMES.DAT. Define a GOLD key by pressing DO, typing SET GOLD KEY, and pressing RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer  RHYMES.DAT                          | Insert | Forward
Command:  SET GOLD KEY
```

Press the number 4 key on the numeric keypad.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer  RHYMES.DAT                          | Insert | Forward
Press the key that you want to use as the GOLD key:
GOLD key set
```

Press the DO key, type DEFINE KEY, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                          | Insert | Forward
Command:  DEFINE KEY
```

Type BOTTOM and press RETURN.

```
And what is a rhyme?
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                          | Insert | Forward
EVE command:  BOTTOM
```

Press CTRL/G.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                          | Insert | Forward
Key defined
```

Now define the GOLD CTRL/G key to enter the TOP command. Press the DO key, type DEFINE KEY, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                          | Insert | Forward
Command:  DEFINE KEY
```

Type TOP and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                          | Insert | Forward
EVE Command:  TOP
```

To assign a key for the TOP command, press the GOLD key (number 4 on the numeric keypad), and then press CTRL/G.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                          | Insert | Forward
Key defined
```

For the rest of your editing session, when you press CTRL/G, EVE executes the BOTTOM command, and when you press the GOLD key (number 4 on the numeric keypad) followed by CTRL/G, EVE executes the TOP command. If you press the GOLD key by mistake, press the Select key to cancel it. Use the SAVE EXTENDED EVE command (see Section 1.12.2) to save key definitions from one editing session to the next.

You cannot define more than one GOLD key at a time. To remove a GOLD key definition, enter the SET NOGOLD KEY command, then press the key you want to undefine. Or, define another GOLD key, which removes the original GOLD key.

Another way of defining a GOLD key is to insert a command in an initialization file, using the following format:

```
SET GOLD KEY  keyname
```

For example, the following command defines keypad key 4 as the GOLD key:

```
SET GOLD KEY KP4
```

1.10 Using the DCL within EVE

You can execute a DCL command from within EVE, or you can use a subprocess to switch between the DCL command level and an EVE editing session very quickly.

1.10.1 Executing a DCL Command

To enter a DCL command from within EVE, press the DO key, type the EVE command DCL and the DCL command you want to execute, and press RETURN. The message *Creating DCL subprocess . . .* appears in the Message buffer.

When the DCL command has executed, EVE creates another window, if necessary, and displays the DCL command and its output in the DCL buffer. (The cursor remains in the buffer it was in before you executed the DCL command.) You can move the cursor to the DCL buffer, select and remove text, and copy it to the editing buffer. Do not enter DCL commands that generate continuous output or run programs that do screen management of their own, such as the Phone Utility. The DCL command is most useful when you want to capture output in an EVE buffer.

1.10.2 Creating a Subprocess

You can create subprocesses to switch between an EVE editing session and DCL command level without terminating your editing session. To create a subprocess, press the DO key, type the SPAWN command, and press RETURN. EVE suspends the current editing session and connects your terminal to a new VMS subprocess. The DCL prompt ($) appears on your terminal screen.

The most common reasons to spawn a subprocess are to invoke the Mail Utility and to run screen-oriented programs, although your subprocess can invoke any VMS utility or execute any DCL command.

To return to your editing session, log out of the subprocess by typing the DCL command LOGOUT and pressing RETURN. EVE resumes the editing session, and the cursor appears in the location it occupied before you spawned the subprocess.

Alternatively, you can supply a DCL command as a parameter to the SPAWN command to create a specific subprocess. For example, to execute the Mail Utility, press DO, type SPAWN MAIL, and press RETURN. The *MAIL>* prompt for the MAIL Utility appears on the screen. When you exit from MAIL, you are automatically logged out of the subprocess and EVE resumes the editing session.

Rather than spawn a process to use the DCL, you can spawn a process for an EVE editing session and then attach back to the parent DCL process to use DCL commands and utilities.

First, create a subprocess for the editing session using the DCL command SPAWN. The SPAWN command creates a subprocess (displayed by the SHOW PROCESS command as "username_1"). At the subprocess level, invoke EVE and conduct the editing session.

When you want to return to the DCL command level, use the EVE command ATTACH to return to the parent process (SHOW PROCESS displays the process as "username").

To resume your editing session, reconnect to the editing subprocess using the DCL command ATTACH with the process name of the subprocess ("username_1"). EVE resumes the editing session and the cursor appears in the location it occupied before you attached to the parent process.

1.11 Using the TPU Command

EVE is an editor built on the VAX Text Processing Utility (VAXTPU), which is a programmable text processor. The TPU command allows you to enter any VAXTPU statement or series of statements that can be expressed on one command line.

To enter a TPU command, press DO, enter the command TPU followed by the VAXTPU statement you want to execute, and press RETURN. For example, to execute the VAXTPU statement APPEND_LINE, which places the current line at the end of the previous line, enter the command TPU APPEND_LINE. For more information on the TPU command, type HELP TPU. See the *VAX Text Processing Utility Manual* for a complete list of VAXTPU statements and procedures.

1.12 Extending the EVE Editor

Since EVE is an editor written in the VAXTPU programming language, you can extend EVE's functions by writing procedures in VAXTPU. This section assumes that you are familiar with the VAXTPU programming language described in the *VAX Text Processing Utility Manual.*

Before you begin writing VAXTPU procedures to modify the EVE editor, DIGITAL recommends that you study the EVE source code, which is stored in SYS$EXAMPLE:EVE$*.TPU. (The wildcard character (*) in the file specification indicates that EVE source code is stored in many files.) These files are put together using EVE$BUILD. Variables and statements in your procedures should be consistent with EVE's variables and statements so that you can avoid making changes that may have an adverse effect on EVE's operations.

1.12.1 Writing and Compiling VAXTPU Procedures

You can write procedures in the VAXTPU programming language that are, in effect, new EVE commands. When you write new EVE command procedures, you should follow these rules:

- Prefix the procedure name with the label *EVE_* so that EVE recognizes the procedure as an EVE command. After you compile the procedure, execute it by pressing the DO key and typing the procedure name without the EVE_ prefix or the underscores. For example, enter SET LEFT MARGIN for the EVE_SET_LEFT_MARGIN procedure. (You may also define a key to execute the new command. See Section 1.9.)
- The words *PROCEDURE* and *ENDPROCEDURE* must start in column 1.
- Initialize all global variables in a single procedure named TPU$LOCAL_INIT or in a module initialization procedure if you build with EVE$BUILD.

If the EVE command you are writing takes any parameters, such as SET LEFT MARGIN 2, where 2 is the parameter, you must define a global variable to associate that parameter with either the data-type string or the data-type integer. When you are using the EVE commands that you have defined, EVE passes the null string ("") if you do not provide a value for a string parameter; it passes the value EVE$K_NO_ARG if you do not provide a value for an integer parameter.

DIGITAL recommends placing global variables in a procedure called TPU$LOCAL_INIT, which is called each time EVE starts. Place all procedures that use parameters in the same file that contains the TPU$LOCAL_INIT procedure. The following are examples of global variable definitions for command parameters:

- The following definition of the global variable eve$arg1_add tells EVE to expect an integer as the first parameter for EVE_ADD:

```
eve$arg1_add := "integer";
```

- The following definition of the global variable eve$arg1_hello tells EVE to expect a string as the first parameter to EVE_HELLO:

```
eve$arg1_hello := "string";
```

The integer variables are required, but the string variables are optional.

In general, each of the global variable names that define command parameters must consist of the following three parts:

- The first part of the variable name must be *eve$*.
- The second part defines the variable's sequence within a procedure. For the first variable, it is *arg1*; for the second, *arg2*; and so on.
- The third part of the variable name is the procedure name without the *EVE_* prefix.

The EXTEND EVE command enables you to compile a VAXTPU procedure without leaving EVE. To compile the procedure that the cursor is in, use the EXTEND THIS command. To compile one procedure, enter the command EXTEND EVE and the name of the procedure you want to compile. To compile all procedures in a file, enter the command EXTEND EVE *. If you

miss a message from the compiler, use the command BUFFER MESSAGES to read the messages stored in the message buffer.

The following example illustrates how to create and compile VAXTPU procedures to define an ADD command, a HELLO command, and the parameters for both commands. Invoke EVE to edit the file MYPROCEDURES.DAT and insert the following text into the file:

```
! Procedure to add two integers and display the result in the message window

procedure eve_add (a1, a2)

local   temp,
        n1,
        n2;

if not eve$prompt_number (a1, n1, "First number to add: ",
                          "No number specified.")
then
    return (FALSE);
endif;

if not eve$prompt_number (a2, n2, "Second number to add: ",
                          "No number specified.")
then
    return (FALSE);
endif;

temp := n1 + n2;
message (str (n1) + " + " + str (n2) + " = " + str (temp));
return (TRUE);

endprocedure;


procedure eve_hello (my_name)
local   the_name;

if eve$prompt_string (my_name, the_name, "Name: ", "We haven't been introduced")
then
    message ("Hello " + the_name);
    return (TRUE);
else
    return (FALSE);
endif;

endprocedure;

Buffer  MYPROCEDURES.DAT                          | Insert | Forward
```

Put the definitions for the three parameter variables in your TPU$LOCAL_INIT procedure, using the syntax shown in file MYPROCEDURES.DAT.

To compile the procedures you have entered into MYPROCEDURES.DAT, press the DO key, type EXTEND EVE *, and press RETURN. If you are going to use the newly compiled commands in the current editing session, you must execute TPU$LOCAL_INIT by entering the command TPU TPU$LOCAL_INIT.

The following section describes how to save the compiled procedure so that you can use the EVE ADD command during an editing session.

1.12.2 Saving VAXTPU Procedures, Key Definitions, and Learn Sequences

The SAVE EXTENDED EVE command saves all currently defined procedures, key definitions, and learn sequences in a section file that you specify. To save the procedures you have compiled and the key definitions and learn sequences you have created during an editing session, enter the SAVE EXTENDED EVE command before you terminate the editing session. Use the following format:

```
SAVE EXTENDED EVE filename.TPU$SECTION
```

The section file is saved in your current default directory, unless you include a device and directory in the file specification. The file type defaults to TPU$SECTION.

You can specify the same file specification each time you execute the SAVE EXTENDED EVE command. By doing this, you add any new key definitions, learn sequences, and procedures to the same section file.

To use this extended version of EVE, you must include the /SECTION qualifier in the command line when you invoke EVE. For example, to invoke EVE to edit the file RHYMES.DAT using the section file WORKDISK:[USER]MYDEFS.TPU$SECTION, enter the following command:

```
$ EDIT/TPU/SECTION=WORKDISK:[USER]MYDEFS.TPU$SECTION RHYMES.DAT
```

Remember, you can define a command symbol to invoke this or any other lengthy command line. You can use all the key definitions, learn sequences, and procedures that you previously defined and saved in WORKDISK:[USER]MYDEFS.TPU$SECTION during your editing session.

For more information on saving key definitions, learn sequences, and VAXTPU procedures, see Appendix A, which describes different types of startup files. Startup files let you tailor the EVE editor to meet your editing requirements.

2 Editing Files with EDT

EDT is an interactive text editor. You can use EDT to edit many kinds of text files—letters, memos, or complex computer programs. With EDT you can create new files, insert text into them, and edit that text. You can also edit text in existing files.

2.1 Introduction

EDT offers many features to make text editing easier and more efficient. These features include the following:

- Three types of editing—keypad mode, line mode, and nokeypad mode. Keypad mode is screen-oriented, which allows you to see several lines of text simultaneously and move the cursor throughout the text in any direction. Line mode enables you to edit text by using line numbers. You can use nokeypad mode commands to define keys. You can move from one mode to another during the same editing session.
- Online HELP. You can use HELP any time during your editing session without affecting your work.
- Journal facility. The journal file protects your editing work in case of a system interruption.
- Access to files and buffers. You can work with as many files and buffers as you need during your EDT session.
- Startup command files. These enable you to personalize the characteristics of your editing sessions.
- Key definition facility. You can define keys to automate your keypad editing work.
- EDT macros. You can create EDT macros to automate your editing work. Macros can be saved for use in later editing sessions.
- Tabbing facility. This feature enables you to create layered text formats.

The discussion of EDT in this guide is designed for the novice as well as the more experienced EDT user. For more detailed information about the EDT editor, see the *VAX EDT Reference Manual.*

2.1.1 Invoking and Terminating EDT

An editing session begins when you invoke EDT using the DCL command EDIT and ends when you terminate EDT with the EXIT or QUIT command. You can start an editing session by creating a file and inserting the text for the file during the course of the session, or you can specify an existing file when you start the session. EDT does not destroy the contents of any existing file that you edit; it simply produces a new version, leaving the old version intact.

Editing Files with EDT

2.1 Introduction

2.1.1.1 Invoking EDT

To invoke EDT, enter the DCL command EDIT. You are prompted for the name of a file. The following example shows how you invoke EDT to edit a file named MEMO.TXT:

```
$ EDIT
_File:MEMO.TXT
```

If you are creating a new file, EDT displays the message *Input file does not exist* followed by the end-of-buffer sign [EOB] and the asterisk prompt (*). If you are editing an existing file, a copy of the first line of text appears on the screen followed by the asterisk prompt.

The following example demonstrates how to invoke EDT to create a new file:

```
$ EDIT NEWFILE.FUN
Input file does not exist
[EOB]
*
```

The following example demonstrates how to invoke EDT to edit a file that already exists:

```
$ EDIT OLDFILE.DAT
     1        This is the first line of the file.
*
```

Once you enter EDT, you can choose between three different modes:

- Line mode—Line mode allows you to use line numbers during your editing session. Line-mode commands are particularly useful for manipulating large blocks of text.
- Keypad mode—Keypad mode allows you to move the cursor to the text you want to edit and perform most editing functions by pressing keypad keys. In keypad mode, you insert text by typing from the main keyboard.
- Nokeypad mode—Nokeypad mode allows you to move the cursor directly to the text you want to modify and to enter nokeypad commands.

One difference between the three modes is that you press keys to edit text in keypad mode, whereas you enter commands to edit text in line mode and in nokeypad mode.

By default, EDT puts you into line mode. You know that you are in line mode when you see the asterisk prompt (*). If you want to enter keypad mode, type the letter C (abbreviation for CHANGE command) when you see the asterisk prompt and press RETURN. If you want to enter line mode from keypad mode, press CTRL/Z. If you want to enter nokeypad mode, enter the SET NOKEYPAD command when you see the asterisk prompt, press RETURN, type the letter C, and press RETURN again.

The following table shows the three EDT modes and the commands you enter to invoke each mode on a video terminal.

Mode	Commands to Invoke Mode
Line	$ EDIT *
Keypad	$ EDIT * C
Nokeypad	$ EDIT * SET NOKEY * C

2.1.1.2 Terminating EDT

Both the EXIT and QUIT commands terminate an editing session; however, only EXIT saves your edits. When you enter the EXIT command at the asterisk prompt, EDT creates an output file containing the edited version of the input file. By default, the output file has the same name and type as the input file. (The version number is incremented by one.) The following example demonstrates how to invoke EDT to edit a file named FUN.DAT and how to terminate the editing session. The output file has the same name as the input file and the version number is increased by one.

```
$ EDIT FUN.DAT
    .
    .
    .
*EXIT
DBA2:[WHITEBRIDGE]FUN.DAT;5  2 lines
```

If you want to override the default and specify a different output file name, enter the EXIT command and specify the new file name as a parameter. In the following example, you invoke EDT to edit a file named FUN.DAT and specify JOKE.DAT as the new file name when you terminate the session.

```
$ EDIT FUN.DAT
    .
    .
    .
*EXIT JOKE.DAT
DBA2:[WHITEBRIDGE]JOKE.DAT;1  2 lines
```

If you do not want to save your edits when you end an editing session, enter the QUIT command. All the edits you made to the file are lost and no output file is created.

The following table shows the three EDT modes and the commands you enter to terminate each mode:

Editing Mode	Commands to Terminate EDT
Line	* EXIT or QUIT $
Keypad	CTRL/Z * EXIT or QUIT $ or GOLD key, COMMAND key, EXIT or QUIT command, ENTER key $
Nokeypad	EX * EXIT or QUIT $ or QUIT $

2.1.2 Using the HELP Facility

EDT's online HELP Facility allows you to get help during your editing session without interrupting your work.

In keypad mode, press the HELP key to get information about keypad functions. For information about line and nokeypad commands, use the line-mode command HELP. The following sections discuss the HELP facility in more detail.

2.1.2.1 Accessing HELP from Line Mode

You can enter the HELP command after the asterisk prompt (*) for information about EDT commands. If you want information about specific commands, type the HELP command followed by the name of the command and press RETURN. For example, if you want to know more about the INSERT command, enter the following:

```
*HELP INSERT
```

If you want information on a qualifier, for example /SAVE, enter the following:

```
*HELP EXIT/SAVE
```

You must include the name of the command, in this case EXIT, and the slash (/) before the qualifier.

EDT can help you get information even when you supply only the first letter or so of the command name. If you enter HELP D, EDT prints the information for both the DEFINE and DELETE commands. HELP CH gives you the CHANGE command, but HELP C supplies information on CHANGE, CLEAR, and COPY.

Use the wildcard character (*) with the HELP command to get information on all subtopics for a command without having to supply the subtopic names. For example, the following command prints information on TYPE/BRIEF and TYPE/STAY:

```
*HELP TYPE *
```

2.1.2.2 Accessing HELP from Keypad Mode

From keypad mode, press the HELP key to get a diagram of the keypad functions. The screen goes blank for a few seconds before the diagram appears.

After EDT displays the diagram, you can press any keypad key to see a description of what that key does. If you press keypad key number 5, for example, descriptions of the BACKUP and TOP keypad functions appear on the screen. At the end of each description are directions for seeing the whole keypad diagram again (by pressing RETURN), reading about other functions (by pressing another keypad key), or returning to your editing session.

For help on a control key sequence such as CTRL/A, press both the control key and the keyboard key. For help on a GOLD key sequence, such as GOLD/A, press only the keyboard key, not the GOLD key.

Note: The HELP diagrams do not display any keyboard or keypad keys that you have redefined.

2.1.2.3 Accessing HELP from Nokeypad Mode

If you are in nokeypad mode and want to get HELP information about nokeypad commands, type EXIT, press RETURN, and enter HELP CHANGE at the asterisk prompt. After the help entry on CHANGE, there are five subtopics:

- ENTITIES
- HARDCOPY
- KEYPAD
- NOKEYPAD
- SUBCOMMANDS

The ENTITIES subtopic contains a list of the nokeypad entity subtopics: character, word, line, range, sentence, page, paragraph, select, vertical, and string. When you want information about one of these entities, include the entity name (for example, sentence) as the last HELP command subtopic:

```
*HELP CHANGE ENTITIES SENTENCE
```

The HARDCOPY, KEYPAD, and SCREEN subtopics provide brief descriptions of hardcopy, change mode, keypad mode, and screen editing. These do not have further subtopics.

Further information on nokeypad commands is contained in the SUBCOMMANDS subtopic. For a complete list of the subtopics covered, enter the following:

```
*HELP CHANGE SUBCOMMANDS
```

To leave HELP and return to your editing session, enter the CHANGE command at the asterisk, as follows:

```
*CHANGE
```

2.1.3 Recovering from System Interruptions

If your EDT editing session is interrupted, you can recover your work by using a journal file. Journal files and the method for recovering from interruptions are discussed in Section 2.7.

2.1.4 Moving from Mode to Mode

Once you become familiar with the three modes in EDT, you can travel from one mode to another to perform your editing tasks. The following table lists the commands you need to know to move from mode to mode:

From	To	Command
Line Mode	Keypad Mode	*CHANGE
Line Mode	Nokeypad Mode	*SETNOKEYPAD *CHANGE
Keypad Mode	Line Mode	CTRL/Z
Keypad Mode	Nokeypad Mode	CTRL/Z *SET NOKEYPAD *CHANGE
Nokeypad Mode	Line Mode	EX *
Nokeypad Mode	Keypad Mode	EX *SET KEYPAD *CHANGE

The following example demonstrates how to invoke EDT to create a new file named FUN.FUN, move to line mode (by default), and then move to keypad mode:

```
$ EDIT FUN.FUN ❶
Input file does not exist
[EOB]
*CHANGE ❷
```

❶ The EDIT command invokes EDT.

❷ The asterisk prompt (*) indicates that you are in line mode and the CHANGE command invokes keypad mode.

The following example demonstrates how to invoke EDT to create a new file named WHY.NOT, enter line mode (by default), enter keypad mode, return to line mode, and, finally, enter nokeypad mode.

```
$ EDIT WHY.NOT ❶
Input file does not exist
[EOB]
*CHANGE ❷
[EOB]
CTRL/Z ❸
*SET NOKEYPAD ❹
*CHANGE ❺
```

❶ The EDIT command invokes EDT.

❷ The asterisk prompt (*) indicates that you are in line mode and the CHANGE command invokes keypad mode.

❸ CTRL/Z returns you to line mode.

❹ The asterisk prompt indicates line mode and the SET NOKEYPAD command invokes nokeypad mode.

❺ The CHANGE command invokes nokeypad mode (only after you enter the SET NOKEYPAD command).

2.2 Using Keypad Mode

Keypad editing is available on VT300-series, VT200-series, VT100-series, and VT52 terminals. In keypad editing, the contents of a file are displayed on the screen as you edit. You can see the changes you make to a file as they take place.

In keypad editing, you press keys to perform editing functions rather than type commands as is done in line and nokeypad editing.

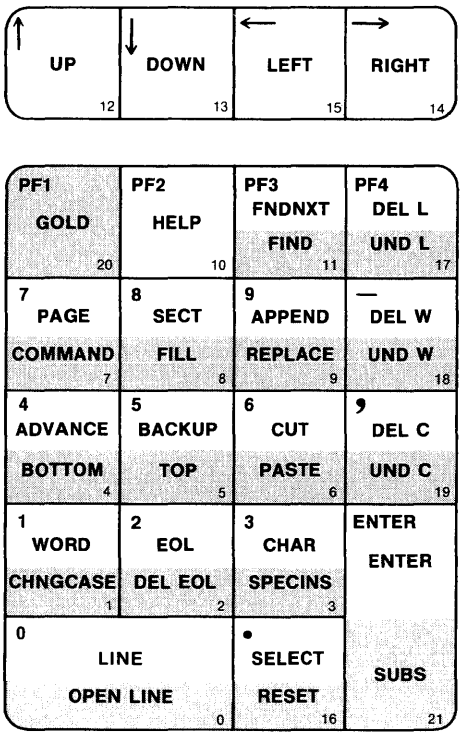
2.2.1 Terminal Keypads

Figure 2–1 shows all the keypad functions available on VT52, VT100-series, and LK201 keyboards. (An LK201 keyboard is used with VT200-series and VT300-series terminals.) Each key in the keypad performs at least one editing command; many of the keys perform two. You can use the standard function (the upper half of the key) by pressing the key. You can use the alternate function (the lower, shaded half of the key) by pressing the GOLD key before pressing the function key.

Notice that the LK201 keyboard includes a numeric keypad (like the keypad on the VT100-series terminal) and a six-key editing keypad with four arrow keys. (Two of the twenty available function keys, HELP and DO, are also displayed.) See the *VAX EDT Reference Manual* for more detailed information about the LK201 function keys.

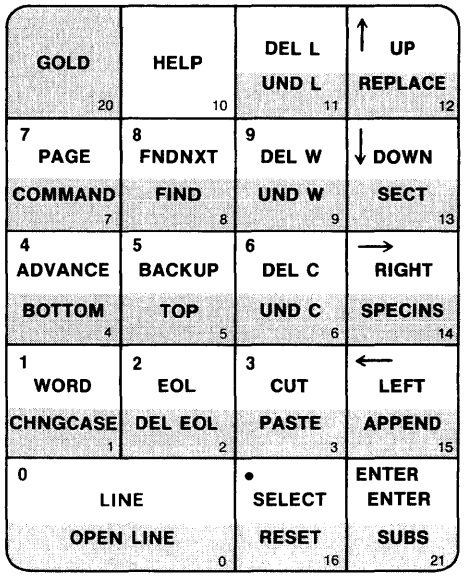
Figure 2–1 VT100, VT52, and LK201 Keypads

Keypad Editing Keys – VT100 Terminals



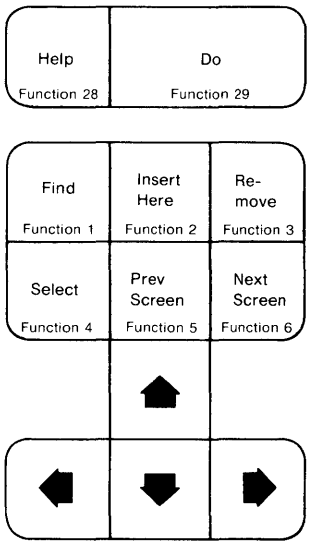
VT100

Keypad Editing Keys – VT52 Terminals

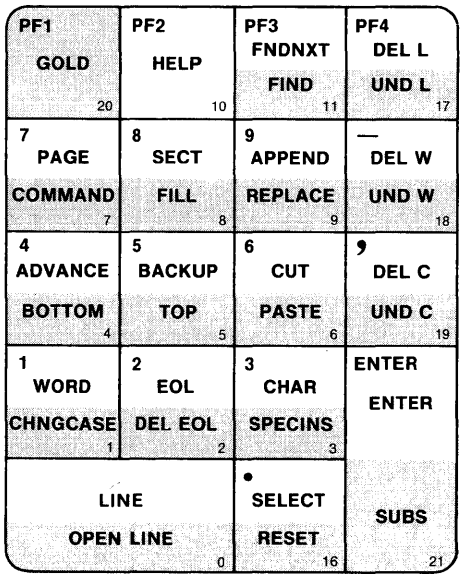


VT52

Keypad Editing Keys – LK201 Keyboard



VT200 Series



ZK-1605-84

2.2.2 Using the GOLD Key

You can use the GOLD key for two purposes:

- To use the alternate of two functions on a keypad key
- To execute a function a specified number of times

To use an alternate function, press the GOLD key followed by the desired key. (You do not need to hold down the GOLD key.) For example, to use UND C, you press the GOLD key first, then the UND C key.

To execute a function a specified number of times, press the GOLD key, type a number (on the main keyboard), and press the function you want repeated. Try it and see how much time you save. The following steps show how to repeat any keypad function a specified number of times:

1 Press the GOLD key.

2 Type a number.

3 Press the desired function.

2.2.3 Inserting Text

Once you have entered keypad mode, you can insert text by typing it on the keyboard. No command is required. Whatever you type becomes part of the file. Your insertion appears on the screen as you type it, and the surrounding text moves as necessary to accommodate it. The cursor marks the starting location for the insertion. When you want to begin a new line, press RETURN to move the cursor to the beginning of the next line and continue typing your text.

As you type new text, you may notice errors in surrounding text. You can move the cursor to these errors and correct them at any time and then move the cursor back and continue to insert text.

2.2.4 Moving the Cursor

There are many ways to move the cursor in keypad mode. The following table lists several keypad functions that move the cursor:

Function Key	Destination of Cursor
TOP	Beginning-of-Buffer
BOTTOM	End-of-Buffer
Left arrow	One character to the left
Up arrow	Up one character
Right arrow	One character to the right
Down arrow	Down one character

The ADVANCE and BACKUP keypad functions cause the cursor to move in a forward or backward direction.

The following example demonstrates how to move the cursor to the beginning (or top) and to the end (or bottom) of the buffer, using the TOP and BOTTOM keypad functions.

Insert the following three lines of text:

```
Five golden rings,
four calling birds,
three french hens,
```

1. Press the GOLD key followed by TOP to move the cursor to the *F* in the word *Five*.
2. Press the GOLD key followed by BOTTOM. Notice the cursor moving to the left bracket of the end-of-buffer sign [EOB].
3. Press the GOLD key followed again by TOP to move the cursor back to the *F*.

Repeat these steps to become familiar with the TOP and BOTTOM keypad functions.

The following example demonstrates how to move the cursor using the four arrow keys: Up, Down, Left, Right.

```
Under a toadstool crept a wee elf,
Out of the rain to shelter himself.
Under a toadstool all in a heap,
Sat a big doormouse, fast asleep.
```

1. Press the TOP keypad function (the GOLD key followed by TOP) to move the cursor to the letter *U* in the word *Under*.
2. Press the Right arrow 9 times, then press the Down arrow once, moving the cursor to the *e* in the word *the*.
3. Press the Left arrow 5 times, then press the Down arrow once, moving the cursor to the *r* in the word *under*.
4. Press the Right arrow 8 times, then press the Down arrow once, moving the cursor to the second *o* in the word *doormouse*.
5. Press the Left arrow 6 times, then press the Up arrow 3 times, moving the cursor to the word *a*.

The next example demonstrates the keypad functions for TOP, LINE, EOL, WORD, and CHAR. Enter the following four lines of text:

```
ONE TWO THREE FOUR FIVE
SIX SEVEN EIGHT NINE
TEN ELEVEN TWELVE THIRTEEN
FOURTEEN
```

1. Press TOP to move the cursor to the first letter of the word *ONE*.
2. Press LINE once, moving the cursor to the *S* in the word *SIX*.
3. Press WORD twice, moving the cursor to the *E* in the word *EIGHT*. Then press EOL. The cursor moves to the end of the line.
4. Press WORD three times, moving the cursor to the *T* in the word *TWELVE*. Then press EOL. The cursor moves to the end of the line.

5 If you are using a VT100-series, a VT200-series, or a VT300-series terminal, press CHAR four times. The cursor moves to the *R* in the word *FOURTEEN*.

The following figure shows the approximate distances that some keypad commands move the cursor.

Keypad Function	Distance Keypad Function Moves Cursor
CHAR	#
WORD	######
EOL	###########
LINE	##
SECT	## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ##

Keypad Function	Distance Keypad Function Moves Cursor
PAGE	##

Note that the PREV SCREEN and NEXT SCREEN keys (on an LK201 keyboard) work like the SECT keypad function, moving the cursor back or ahead by 16 lines.

2.2.4.1 How EDT Views Words

You can determine the way that EDT recognizes a word, sentence, paragraph, or page. By default, EDT identifies a word as any group of contiguous characters starting at the current cursor position and continuing in the current direction until one of the following characters is encountered:

- Space ( )
- Horizontal tab (TAB)
- Linefeed (LF)
- Vertical tab (VT)
- Form feed (FF)

- Carriage return (CR)
- Line terminator

For example, enter the following text and move the cursor throughout the paragraph by pressing the WORD key. Notice that the cursor stops on the first letter of each word.

```
This is the first line of text.
Is this the second?
This must be the third.
How about a fourth line?
And, a fifth, perhaps?
Will anyone take a sixth?
Number seven, please.
```

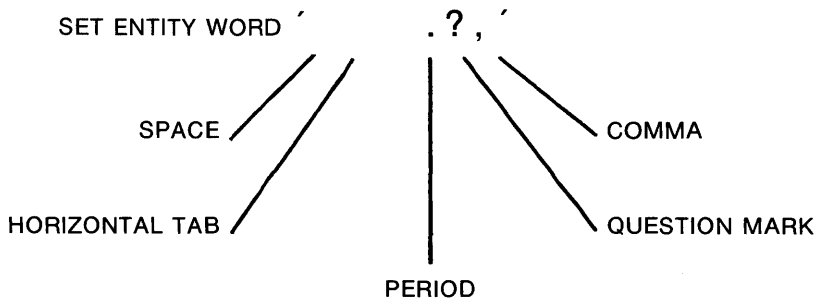
You can use the SET ENTITY WORD command to change the way EDT views words. If you want EDT to recognize periods, question marks, and commas, enter the following command line in your startup command file (for information about startup command files, see Section 2.11):

```
SET ENTITY WORD ".?,"
```

When you use the WORD key to move throughout the text again, notice that the cursor stops on all periods, question marks, and commas. For more information about the SET ENTITY command, see the *VAX EDT Reference Manual.*

Figure 2–2 shows the characters that can be used in the SET ENTITY command.

Figure 2–2 Using the SET ENTITY WORD Command



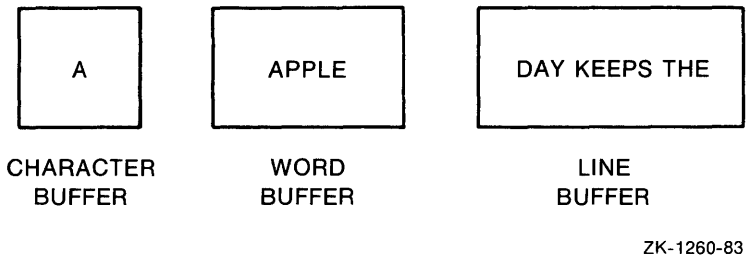
2.2.5 Deleting and Undeleting Text

You can delete text by character, word, and line. The DELETE and DEL C keys delete characters. DEL W and LINEFEED delete words or parts of words. DEL L, DEL EOL, and CTRL/U delete single lines or parts of lines. The deleted text is stored in one of three buffers so that you can restore it using the UND commands (UND C, UND W, and UND L). A buffer is a temporary holding area for text. The character buffer contains the last character deleted; the word buffer contains the last word deleted; and the line buffer contains the last line deleted. Only the most recent deletion can be

restored; you can restore this unit of text numerous times to any location. See Section 2.6 for more information on buffers.

Figure 2–3 shows the three buffers that EDT fills when you use the delete and undelete keys.

Figure 2–3 Three EDT Buffers Used for Deleting and Undeleting Text



Use the following keypad functions to delete text:

Keypad Function	What It Does
DELETE	Deletes the preceding character
DEL C	Deletes the current character
DEL W	Deletes to the end of the word
LINE FEED	Deletes to the beginning of the previous word
DEL L	Deletes to the next line
CTRL/U	Deletes from the beginning of the line to the cursor
DEL EOL	Deletes from the cursor to the end or beginning of the line, depending on the current cursor direction
CUT	Deletes the selected string from the current file and stores it in the PASTE buffer. The selected string is all the text between the cursor location when you pressed the SELECT keypad function and the position to which you moved the cursor. (Note that the REMOVE key on the LK201 keypad works like the CUT keypad function.)

Enter the following three lines of text:

```
CHARACTER

WORD WORD WORD WORD WORD

LINE LINE LINE LINE LINE LINE LINE LINE
```

The following steps show how to delete and restore characters:

1. Move the cursor to the first C in the word *character*. Press DEL C. The C disappears.
2. Press the GOLD key followed by UND C. The C reappears.
3. Move the cursor to the *H*. Press DEL C to make it disappear.

4 Then press the GOLD key followed by UND C. The H is restored.

5 Press the GOLD key followed by UND C again, another H appears.

Notice that the DEL C keypad function deletes the character directly at the cursor and the DELETE key (on the main keyboard) deletes the character immediately to the left of the cursor. You can use the UND C keypad function to restore the most recently deleted character in either case. Press the DELETE key and the DEL C keypad function to see the difference between the two.

Use the following steps to delete and restore words:

1 Move the cursor to the *W* in *WORD*. Press DEL W. The word *WORD* disappears.

2 Press the GOLD key followed by UND W to restore the word.

3 Press the GOLD key followed by UND W again, another *WORD* appears.

Notice that the DEL W keypad function deletes to the end of the current word, and the LINE FEED key (on the main keyboard) deletes to the beginning of the preceding word. Press the LINE FEED key to see the difference between the DEL W keypad function and the LINE FEED key. Again, you can use the UND W keypad function to restore the deleted word in either case.

To delete and restore entire lines of text, do the following:

1 Move the cursor to the beginning of the line of LINES. Press the DEL L keypad function. The entire line disappears.

2 Press the GOLD key followed by UND L to restore the line.

3 Press the GOLD key and UND L several times to create multiple lines.

Press CTRL/U to delete text from the cursor to the beginning of the line. Notice that the cursor moves to the beginning of the line. Press UND L to restore the line.

2.2.6 Locating Text

Use the FIND and FNDNXT keypad functions to locate strings of text. When you press the GOLD key followed by FIND, the following prompt appears at the bottom of the screen:

```
Search for:
```

Type the string of text you are looking for and press one of the following keys:

- ADVANCE
- BACKUP
- ENTER
- DO

If you press ADVANCE, EDT searches in a forward direction for the specified string. If you press BACKUP, EDT searches in a backward direction. If you press ENTER or DO, EDT searches in the direction already set.

Note: The ADVANCE and BACKUP keys also set the direction for subsequent EDT commands.

Enter the following text:

```
Ella will not be able to attend the concert
tonight. She has a sore throat.  Perhaps you could give
the ticket to somebody in your music class.  Ella wants
to see the program when she is feeling better.
```

Press the GOLD key followed by the FIND keypad function. When you are prompted for a search string, enter the word *ticket*.

```
Search for: ticket
```

Now press BACKUP. The search starts at the end of the text (where the cursor is located) and continues towards the beginning of the text until it finds the word *ticket*.

Next search for the word *program*. Press the GOLD key followed by FIND. When you are prompted for the search string, enter the word *program*.

```
Search for: program
```

If you press BACKUP again, you see the response *String was not found* because the word *program* is located after the word *ticket*. Now press ADVANCE followed by FNDNXT, and EDT finds the string.

Remember the following three items when you are entering a search string:

1. EDT ignores diacritical marks and the case of letters while making searches (unless you enter the SET SEARCH EXACT command).
2. DELETE is the only key you can use to edit an incorrectly typed search string.
3. To cancel a search string, press CTRL/U.

If you want to find the next occurrence of the string you are searching for, use the FNDNXT keypad function. EDT searches in the direction already set. If EDT cannot find the string, it displays the message *String was not found.* You can reverse the direction of the search by pressing either ADVANCE or BACKUP before pressing FNDNXT.

Enter the following text, then press GOLD and FIND, and enter the search string *little* when prompted:

```
One little, two little, three little chickadees,
four little, five little, six little chickadees,
seven little, eight little, nine little chickadees,
ten little chickadees feeding.
```

```
Search for: little
```

Because the cursor is at the end of the text, you must press the BACKUP keypad function to set the search in a backward direction. Now press FNDNXT nine times. EDT finds each occurrence of the search string. When the cursor arrives at the first *little*, press the ADVANCE keypad function to reverse the direction of the search. Every time you press FNDNXT, the cursor moves forward to the next occurrence of the word *little*.

2.2.7 Moving Text

You can move text using three different groups of keypad functions:

- RETURN and OPENLINE—Pressing RETURN causes both text and cursor to move down the screen line by line. If you press OPENLINE, the text moves down the screen line by line, but the cursor remains in the same place.
- DEL L, UND L, DEL W, UND W, DEL C, and UND C—Deleting a unit of text (character, word, or line) from one location, moving the cursor to another location, and then undeleting the text is a way to move small portions of text. (See Section 2.2 for more information about deleting and undeleting text.)
- CUT and PASTE (or REMOVE and INSERT HERE)—Cutting (or removing) a unit of text and pasting (or inserting) it in another location is the method you use for moving large portions of text. This method is described in the rest of this section.

Use the following three keypad functions to move text:

- SELECT
- CUT (or REMOVE)
- PASTE (or INSERT HERE)

Press SELECT to mark one end of the string of text that you want to delete or move. Then move the cursor either backward or forward to the other end of the string, and press CUT. Before you press CUT, you can correct a mistake by pressing the GOLD key followed by RESET to cancel the select range and start over. The selected text is highlighted in reverse video.

Press CUT to delete the selected string of text from the current file and store it in the PASTE buffer. (See Section 2.6 for more information about the PASTE buffer.) The selected text is all the text between the cursor location when you pressed SELECT and the position to which you have moved the cursor.

If you press CUT before you press SELECT, you see the message *No select range active.*

To insert the contents of the PASTE buffer to the left of the cursor position, press GOLD followed by PASTE.

To summarize, use the following steps to move text:

1. Place the cursor at the beginning of the text you want to move.
2. Press SELECT to mark the location.
3. Move the cursor to the end of the select range.
4. Press CUT to delete the text from its current position.
5. Move the cursor to the character just beyond where you want the text inserted.
6. Press the GOLD key followed by PASTE.

Note: If you want to restore the select string to its original location after you perform Step 4 above, press the GOLD key followed by PASTE.

Enter the following text:

```
january february march april
may june july november
december august september october
```

In order to move the string *august september october* after *july*, move the cursor to the *a* in *august*. Press SELECT. Then move the cursor to the *r* in *october*. Press CUT. The selected string disappears into the PASTE buffer. Now move the cursor after the word *july* and press GOLD followed by PASTE (GOLD/PASTE). A copy of the selected string in the PASTE buffer appears on the screen between the words *july* and *november*.

If you press GOLD/PASTE again, you get another copy of the contents of the PASTE buffer. Every time you press GOLD/PASTE, you get another copy of the string *august september october*. You can always get as many copies of the PASTE buffer as you want by pressing GOLD/PASTE. However, once you perform another SELECT and CUT operation specifying a different string, the contents of the PASTE buffer change.

If you want to add to the text in the PASTE buffer, retaining the text already there, use APPEND. APPEND deletes the select range from the current buffer and adds it to the end of the PASTE buffer.

For example, if your PASTE buffer contains the text *Wolfgang Amadeus* and you want to add the text *Mozart*, follow the steps below:

1. Press SELECT.
2. Type the word *Mozart*. (Precede the word with a space.)
3. Press APPEND.

Now the PASTE buffer contains the text:

```
Wolfgang Amadeus Mozart
```

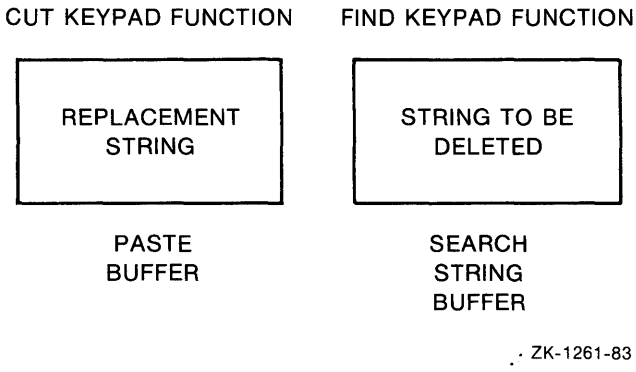
2.2.8 Substituting Text

You can use the SUBS keypad function or the REPLACE keypad function to replace one string of text with another.

When you make substitutions using SUBS, you are using both the PASTE buffer and the search string buffer. You use CUT to put the replacement string into the PASTE buffer and FIND to put the string you want to find and delete into the search string buffer. Figure 2–4 shows both buffers. The following five steps demonstrate how to use SUBS to make substitutions:

1. Press SELECT and enter the replacement text.
2. Press CUT. (The select range disappears into the PASTE buffer.)
3. Press the GOLD key followed by FIND and enter the search string (the text you want to replace).
4. Press either ADVANCE or BACKUP to find the first occurrence of the search string in the direction you choose.
5. Press the GOLD key followed by SUBS to exchange the existing text for the replacement text.

Figure 2–4 Two EDT Buffers Used for Substituting Text



When you use SUBS, EDT performs the substitution first and then moves to the next occurrence of the search string. If EDT cannot find another occurrence of the search string, it prints the message "String was not found." If you do not want to make a particular substitution, press the FNDNXT keypad function, and the cursor moves to the next occurrence of the search string. If you want to change that one, press SUBS again. Thus, you can move through the buffer pressing SUBS each time you want to make a replacement and pressing FNDNXT when you want EDT to leave the search string alone.

Note: You must use the FIND keypad function when you are making substitutions with SUBS because you are replacing text that matches the search string with the contents of the PASTE buffer. SUBS does not work correctly if you do not use FIND.

Enter the following text:

```
Susanne grabbed the red apple and gobbled it down.
Suddenly her face turned quite red.  Glancing towards
the red house she saw the huge tree bathed in red
leaves.
```

Now perform the following steps to substitute the word *green* for the word *red*:

1. Press SELECT and enter the word *green*.
2. Press CUT. (The word *green* disappears into the PASTE buffer.)
3. Press GOLD followed by FIND and enter the word *red*.
4. Press either ADVANCE or BACKUP to find the first occurrence of *red* in the direction you choose.
5. Press SUBS to substitute the word *green* for the word *red*.
6. Press SUBS three more times.

Your resulting text should look like the following:

```
Susanne grabbed the green apple and gobbled it down.  Suddenly
her face turned quite green.  Glancing towards the green house
she saw the huge tree bathed in green leaves.
```

REPLACE deletes the text in the select range and replaces it with the contents of the PASTE buffer. For example, if your PASTE buffer contains the words *paste paste paste,* and your select range contains the words *select select select,* press REPLACE to make your select range contain the words *paste paste paste,* as the following steps demonstrate:

1. Press SELECT.
2. Type the words *paste paste paste.*
3. Press CUT. Now your PASTE buffer contains the words *paste paste paste.*
4. Press SELECT again.
5. Type the words *select select select.*
6. Press REPLACE. Now your select range contains the words *paste paste paste.*

2.2.9 Five More Keys to Use with the GOLD Key

Five keypad functions associated with the GOLD key are summarized in the following table:

Keypad Function	What It Does
COMMAND	Enables you to enter a line-mode command from keypad mode.
CHNGCASE	Reverses the case of letters in your text. Uppercase letters become lowercase; lowercase letters become uppercase.
FILL	Reorganizes the select range so that the maximum number of whole words can fit within the current line width.
RESET	Changes the following conditions of your editing session: • Cancels an active select range. • Empties the search buffer so that there is no current search string. • Sets EDT's current direction to ADVANCE. • Sets EDT to the default DMOV state. (DMOV is a Nokeypad command that returns your editing session to EDT's default state, where EDT does not alter the case of letters during move operations.)
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. For information about the DEC Multinational Character Set, see the *VAX EDT Reference Manual.*

For more information about all the available keypad keys, see the *VAX EDT Reference Manual.*

2.3 Using Line Mode

You can use EDT's line editing facility with any interactive terminal—hardcopy or screen. Line editing uses the line as its point of reference. EDT moves through the text line by line, not character by character as in the two other editing modes. Line editing commands are particularly useful for manipulating large blocks of text.

2.3.1 Line Numbers

To help you locate and edit text, EDT assigns line numbers. These line numbers are not part of the text and are not kept when you end an editing session. To see the line numbers of an already existing file, enter the TYPE WHOLE command after the asterisk prompt (*). Notice that the text you enter is indented 12 spaces.

The following example demonstrates how to display your line numbers:

```
*TYPE WHOLE
    1           oneoneoneoneoneoneone
    2           twotwotwotwotwo
    3           threethreethree
    4           fourfourfourfour
    5           fivefivefivefive
[EOB]
*
```

Line numbers have the following characteristics:

- They are assigned to every line in every buffer in every editing session, including newly inserted lines and text added with the INCLUDE command.
- They start with 1 and increment by 1, unless otherwise specified with the RESEQUENCE/SEQUENCE command.
- They are decimal numbers if newly inserted.
- They are removed by the EXIT command, unless otherwise specified.
- They can be renumbered in increments of 1 or more with the RESEQUENCE command.

When you insert new text, EDT numbers the lines using decimal numbers. For example, if you add a line of text between lines 13 and 14, it is numbered 13.1. To avoid any visual confusion when working with decimal numbers, enter the RESEQUENCE command after the asterisk prompt to change the sequence of line numbers. When you enter RESEQUENCE, EDT renumbers all the lines from the cursor to the end of the buffer in increments of 1. You can specify a range of lines within the file by entering a range after typing RESEQUENCE.

If you insert text after the fourth line in the following example and enter the TYPE WHOLE command again, you will see the newly inserted text preceded by decimal numbers. To change the sequence of line numbers, enter the RESEQUENCE command.

```
*TYPE WHOLE
    1           oneoneoneoneoneoneone
    2           twotwotwotwotwo
    3           threethreethree
    4           fourfourfourfour
    4.1         four_point_one
    4.2         four_point_two
    4.3         four_point_three
    5           fivefivefivefive
[EOB]
*RESEQUENCE
    1           oneoneoneoneoneoneone
    2           twotwotwotwotwo
    3           threethreethree
    4           fourfourfourfour
    5           four_point_one
    6           four_point_two
    7           four_point_three
    8           fivefivefivefive
[EOB]
*
```

The /SEQUENCE qualifier determines the increments EDT uses to number lines. When you enter the following command and include two numbers separated by a colon (n:m), EDT numbers the lines in the buffer, assigning number *n* to the first line and incrementing by number *m*:

```
RESEQUENCE/SEQUENCE:n:m
```

The following example resequences the contents of the buffer, assigning number 6 to the first line and incrementing by 4:

```
*RESEQUENCE/SEQUENCE:6:4 [RET]
3 lines resequenced
*TYPE WHOLE [RET]
    6           Washington, Maine
    10          Vermont, New York
    14          Ohio, New Mexico
    18          Colorado, Virginia
    22          Florida, Arizona
    26          Alabama, Oregon
    30          Minnesota, New Hampshire
    34          Texas, California
    38          Delaware, Michigan
[EOB]
*
```

2.3.2 Inserting Text

When you want to insert text, enter the INSERT command after the asterisk prompt (*). The cursor indents 16 spaces and waits for you to start typing. (Indenting is on the screen or paper only and does not become part of the edited text.) You can enter as many lines as you want. Each time you come to the end of a line, press RETURN to move to the beginning of the next line.

As you enter a line, you can delete characters by pressing the DELETE key. You can delete a portion of a line, from the cursor to the left margin, by pressing CTRL/U. However, when you end a line with RETURN, you can no longer delete characters from that line in insert mode. (See Section 2.3 for more information about deleting text.) When you finish entering text, press CTRL/Z. EDT echoes ^Z on the screen and returns you to the asterisk prompt.

To display the text you just inserted, enter the TYPE WHOLE command after the prompt.

Note: EDT, which inserts text in front of the current line, is different from many other text editors that insert text following the current line.

The following example demonstrates how to insert text:

```
$ EDIT NEWFILE.DAT
Input file does not exist
[EOB]
*INSERT
                This is the first line that I would like
                to type.  At the moment there is no file
                named newfile.dat in my directory.  When
                I finish my editing session and type the
                EXIT command, EDT will save a copy of
                this text in my directory under the name
                newfile.dat.
[EOB]
*EXIT
DBA0:[GLOVER]NEWFILE.DAT;1 5 lines
```

2.3.3 Ranges

When you want a line-mode command to affect a specific part of the buffer, you must enter a range. Ranges can specify one or more lines. In addition, the multiple-line ranges can be contiguous or noncontiguous.

The following table describes the different ranges you can specify when editing in line mode:

Range Type	Description
period(.)	Current line
number	EDT line number
'string'	Next line containing the quoted string
BEGIN	First line of the buffer
END	After the last line in the buffer ([EOB])
LAST	Last line EDT was at in the previous buffer
WHOLE	Entire buffer
BEFORE	All lines in the buffer before the current line
REST	All lines in the buffer starting with the current line and ending with the last line

The following examples demonstrate each range type just listed:

EXAMPLES

1 `TYPE .`

This command displays the current line.

2 `TYPE 35`

This command displays line 35.

3 TYPE *December*

This command displays the first line it encounters containing *December*.

4 TYPE BEGIN

This command displays the first line of the current buffer.

5 TYPE END

This command displays the [EOB] mark.

6 TYPE LAST

This command displays the most recent line of text you displayed. The LAST specifier can only be used by itself; EDT does not accept LAST with other range specifiers or symbols or with buffer names.

7 TYPE WHOLE

This command displays every line in the current buffer.

8 TYPE BEFORE

This command displays the group of lines in the current buffer starting with the first line and ending with the line just before the current line.

9 TYPE REST

This command displays the group of lines in the current buffer starting with the current line and ending with the last line in the buffer.

You can use the following symbols and words with the range types listed above while editing in line mode:

Symbol/Word	Description
, or AND	Used to join noncontiguous ranges in a list; only single lines can be joined in this way
: or THRU	Indicates a group of lines starting with the first range specifier and ending with the second
n	Indicates the number of lines from the current line
# n or FOR n	Indicates the next "n" number of lines
+ "string" or "n"	Indicates that "string" or "n" refers to a line or lines after the current line
– "string" or "n"	Indicates that "string" or "n" refers to a line or lines before the current line
ALL "string" or "n"	Indicates that the command applies to all lines containing "string"

The following examples demonstrate the symbols and words just listed:

EXAMPLES

1 `TYPE 3,6,8`

This command displays lines 3, 6, and 8.

2 `DELETE 4 AND 13`

This command deletes lines 4 and 13.

3 `TYPE 25:`*`Hartford`*

This command displays the group of lines starting with line number 25 and ending with the line containing *Hartford*.

4 `TYPE . THRU 150`

This command displays all the lines in the group starting with the current line and stopping with line number 150.

5 `DELETE 4#3`

This command deletes line 4 and the three lines following line 4.

6 `TYPE . FOR 10`

This command displays the current line and the next ten lines; the number 10 refers to the tenth line after the current line.

7 `TYPE BEGIN +6`

This command displays the seventh line of the current buffer.

8 `DELETE -3 THRU .`

This command deletes the current line and the three lines preceding it; the number –3 refers to the third line before the current line.

9 `TYPE "Dear"+2 THRU "Sincerely"-2`

This command displays the body of the letter, starting with the first paragraph and ending with the last.

10 `TYPE . THRU 1000 ALL "Date:"`

This command displays every line containing the string "Date:" that appears in the group of lines starting with the current line and ending with line number 1000.

2.3.4 Deleting Text

You can delete individual lines or groups of lines by using the DELETE command. After a delete operation, EDT displays the line following the last line deleted; this line is the new current line.

Use the following syntax:

```
DELETE range
```

The following example demonstrates how to delete line 2 in a file named FUN.DAT:

```
$ EDIT FUN.DAT
*   1           This is the first line of a file named FUN.DAT.
*TYPE WHOLE
    1           This is the first line of a file named FUN.DAT.
    2           This is the second.
    3           This is the third,
    4           and the fourth.
*DELETE 2
1 line deleted
    3           This is the third,
*TYPE WHOLE
    1           This is the first line of a file named FUN.DAT.
    3           This is the third,
    4           and the fourth.
*
```

If you enter the DELETE command and do not specify a range, EDT deletes the current line.

When you delete a range of lines, EDT deletes the lines and displays a message stating the number of lines deleted. The message is followed by a display of the next line in the text buffer.

You can use the /QUERY qualifier if you want EDT to prompt you before deleting each line of a specified range. The prompt is a question mark. Answer the prompt with one of the following four responses:

- Y (Yes) Delete this line.
- N (No) Do not delete this line.
- A (All) Delete all remaining lines in the specified range.
- Q (Quit) Quit the delete operation.

You must enter one of these responses before EDT continues to the next line or exits from the delete operation. The following example demonstrates the /QUERY qualifier with the DELETE command:

```
*DELETE 7 THRU 13/QUERY [RET]
    7           This is the first line of the range.
?N [RET]
    8           This is the second line of the range.
?Y [RET]
1 line deleted
    9           This is the third line.
?A [RET]
5 lines deleted
    14          This line is not in the range.
*
```

2.3.5 Substituting Text

When you want to substitute one string for another, use either the SUBSTITUTE or SUBSTITUTE NEXT command. These are the only line editing commands that can alter text within a line, as opposed to changing the entire line. The SUBSTITUTE command operates on the current line or on a specified range. It has the capacity to make a global substitution; that is, you can replace every occurrence of one string with another by using only one line-mode command. The SUBSTITUTE NEXT command makes a substitution at the next occurrence of the string within the buffer.

The syntax for the SUBSTITUTE command is as follows:

```
SUBSTITUTE /old-string/new-string/[range] [/QUERY]
```

To substitute the string GOOD for BAD throughout a buffer, enter the line-mode command SUBSTITUTE, followed by the old string and the new string. Separate all three with the same delimiter. (A delimiter is a character that marks the beginning or end of a string.) To apply the command to the entire buffer, specify WHOLE as the range parameter. When the operation completes, EDT supplies a message indicating how many substitutions were made.

Enter the following text:

```
He felt bad.  He had had a bad time in a bad part of
the city. How did he know that all the stores would
have bad merchandise. So, he had bad luck trying to
find the perfect gift.  His stomach felt bad after a
bad lunch.
```

Enter the following command:

```
*SUBSTITUTE/BAD/GOOD/WHOLE
```

The resulting text will look like this:

```
He felt GOOD.  He had had a GOOD time in a GOOD part of
the city. How did he know that all the stores would
have GOOD merchandise. So, he had GOOD luck trying to
find the perfect gift.  His stomach felt GOOD after a
GOOD lunch.
```

Slashes are not the only characters you can use to delimit strings. Most nonalphanumeric characters will work, as long as the delimiters are matched and do not occur in either string. For example, the following command substitutes the string A/3 for A/2 in the current line, using dollar signs ($) as delimiters:

```
*SUBSTITUTE$A/2$A/3$
    25          SIZE = A/3;
1 substitution
*
```

Note that a global substitution replaces all occurrences of the string, regardless of case, diacritical marks, or surrounding characters. If you want EDT to search for exact comparisons of case, enter the SET SEARCH command and specify the parameter EXACT. (For information about the SET SEARCH command, see the *VAX EDT Reference Manual*.)

If the search string occurs in the middle of a longer string, the substitution will still be made. For instance, a global substitution of IN for AT would change all words containing the string AT. (LATER would become LINER, THAT would become THIN, SAT would become SIN, and so on.) To get EDT to prompt you before each substitution, use the /QUERY qualifier with the SUBSTITUTE command:

```
*SUBSTITUTE/ in / at /WHOLE/QUERY
    1       He was in the point of no return.
?y
    1       He was at the point of no return.
1 substitution
```

EDT prompts you for one of the following responses before each occurrence of the search string:

Y—Yes, do the substitution
N—No, do not do the substitution
Q—Quit, terminate the command
A—All, do the rest of the substitutions without query

2.3.6 Moving Text from One Location to Another

You can use the MOVE and COPY commands to move one or more lines of text from one place to another. The effect of these commands is similar; the only difference is that the COPY command does not delete the text from its original location, whereas the MOVE command does.

The syntax for the MOVE command is as follows:

```
MOVE first range TO second range/QUERY
```

When you use the MOVE command, EDT moves the lines in the first range above the line in the second range. EDT deletes the original copy of the text. For example, if you want to move lines 43 through 56 to above line 13, enter the following command:

```
*MOVE 43 THRU 56 TO 13 [RET]
```

In the following example, EDT moves the current line to above line 65:

```
*MOVE TO 65 [RET]
```

The second range always refers to a single line. EDT inserts the line or lines specified by the first range just above the line specified by the second range. The lines that are moved are renumbered to be consistent with their new location.

You can use the /QUERY qualifier with the MOVE and COPY commands to verify each line to be inserted in the range. (See Section 2.3 for more information about the /QUERY qualifier.)

Use the following COPY command to copy text without deleting the original text:

```
COPY first-range TO second-range/QUERY/DUPLICATE:n
```

When you use the COPY command, EDT copies the lines in the first range to a location above the line in the second range. For example, if you want to copy lines 16 through 24 to above line 3, enter the following command:

```
*COPY 16 THRU 24 TO 3
```

In the following example, EDT copies lines 15 through 60 to the position just before line 95.

```
*COPY 15 THRU 60 TO 95
```

You can use the /DUPLICATE qualifier with the COPY command if you want to insert the range of text more than once. In the following example, EDT copies the first line and places it just above line 65. This operation is repeated 9 times.

```
*COPY BEGIN TO 65/DUPLICATE:9 [RET]
1 line copied 9 times
*
```

2.3.7 Replacing Text

The REPLACE command combines the DELETE and INSERT functions in one command. You can use REPLACE when you need to delete a block of text and want to type new text in that same location. The syntax of the REPLACE command is as follows:

```
REPLACE range [RET] text [CTRL/Z]
```

When you use the REPLACE command, EDT deletes the line or lines specified by the range. As with the DELETE command, EDT prints a message telling you how many lines you have removed.

If you supply no specifiers with the command, EDT deletes the current line. As soon as EDT finishes deleting the specified line or lines, it shifts to the insert state. (The cursor moves to the right, just as it does when you enter the INSERT command.)

Anything you type after pressing RETURN is inserted. When you finish typing the new text, press CTRL/Z to signal EDT that you want to return to the asterisk prompt (*).

The following REPLACE command deletes the current line and shifts to the insert state:

```
*TYPE .
   17      This is the current line.
*REPLACE
1 line deleted
           I have just deleted the current line and
           am adding new lines to my text. <CTRL/Z>

   18      This is the next line.
*TYPE 16 THRU 18
   16      This is the line before the current line.
   16.1    I have just deleted the current line and
   16.2    am adding new lines to my text.
   18      This is the next line.
```

2.4 Using Nokeypad Mode

You can use nokeypad editing on VT300-series, VT200-series, VT100-series, and VT52 terminals.

The commands you use in nokeypad editing are English words or abbreviations that you enter from the main keyboard to move the cursor and edit text. As you type nokeypad commands, they are displayed on the lower left portion of the screen. As in line mode, you press RETURN to process the commands. You can join several commands on a single line and process all of them by pressing RETURN. Note the following characteristics of the cursor in nokeypad mode:

- The cursor is at the top of the screen when you enter nokeypad mode.
- You can move the cursor by using the arrow keys.
- When you do not specify a range with your nokeypad command, you can affect the text at the cursor.

When you type two commands on the same line, such as 3W D-C, you have the option of separating them with a space (for the sake of clarity), even though EDT does not require one.

Nokeypad commands can be used to define keys for keypad mode editing.

See the *VAX EDT Reference Manual* for information on using nokeypad mode.

2.5 Modifying an EDT Environment

EDT provides many tools for modifying your own EDT environment. You can change the appearance of your screen display, the behavior of lines of text as you insert them, and the mode in which you are working. The following section discusses some of the SET commands provided by EDT. See the *VAX EDT Reference Manual* for descriptions of all the available SET commands.

2.5.1 Using SET Commands

You can use the SET commands to change the way EDT works. Some SET commands affect the display of text on your screen. For example, you can specify the number of lines you want displayed (SET LINES number) and whether or not the lines have line numbers (SET [NO]NUMBERS). You can also specify that EDT make an exact search (SET SEARCH EXACT).

By default, your screen contains 22 lines. If you want to decrease the screen display to 5 lines, enter the command SET LINES 5. If you are editing at slow data rates, you can increase your editing speed by decreasing the number of lines displayed on your screen.

When you are working in line mode, EDT displays line numbers by default. If you do not want the numbers to appear on the screen or paper, enter the command SET NONUMBERS.

By default, when you press the GOLD key followed by the FIND key, EDT searches for the specified string, disregarding the case of letters. For example, if you enter the search string *course*, EDT finds every occurrence of the word (for example, *course, Course, COURSE*). But, when you enter SET SEARCH

EXACT, EDT only finds occurrences of the word that exactly match the specified string (*course*).

You can use the following line-mode commands to enter one of the other modes:

```
SET KEYPAD
SET NOKEYPAD
```

In a startup command file, you would use the following SET commands to determine your own default editing mode:

```
SET MODE LINE
SET MODE CHANGE
```

For example, to enter nokeypad mode from keypad mode, enter the following command after pressing GOLD/COMMAND:

```
Command: SET NOKEYPAD
```

One SET command, SET QUIET, even controls the sound of your terminal. You can use this command to suppress the terminal bell that signals an error.

2.5.2 Using SHOW Commands to See What Is Set

EDT provides the SHOW commands to enable you to see what is set and what is not set. For every SET command there is a SHOW command. If you want to see the number of lines on your screen, enter the SHOW LINES command. If you want to see whether EDT is performing an exact search, enter the SHOW SEARCH command.

For example, if you want to check the number of lines displayed on your screen, enter the following command:

```
*SHOW LINES
22
```

Or, if you want to see whether line numbers will be displayed in line mode, enter the following command:

```
*SHOW NUMBERS
numbers
```

The following table lists the SET commands discussed so far along with their corresponding SHOW commands:

SET Command	SHOW Command
SET KEYPAD SET NOKEYPAD	SHOW KEYPAD
SET LINES	SHOW LINES
SET MODE LINE SET MODE CHANGE	SHOW MODE
SET NUMBERS SET NONUMBERS	SHOW NUMBERS

SET Command	SHOW Command
SET QUIET SET NOQUIET	SHOW QUIET
SET SCREEN	SHOW SCREEN
SET TRUNCATE SET NOTRUNCATE	SHOW TRUNCATE
SET WRAP SET NOWRAP	SHOW WRAP

For a complete list of the SHOW commands provided by EDT, see the *VAX EDT Reference Manual.*

2.6 Using Buffers

Buffers are temporary holding areas for text. They enable you to work with more than one file at a time. You can use the buffers available in EDT to do the following:

- Divide one or more files into sections
- Move part or all of another file into your editing session
- Create a file from part or all of the text in a buffer

When you start an editing session, EDT automatically provides a buffer called MAIN. The MAIN buffer serves as the work area for text you insert and edit. If you are editing a file that already exists, EDT puts a copy of the contents of the file into this MAIN buffer.

The MAIN buffer has the following characteristics:

- MAIN is provided automatically when you invoke EDT.
- MAIN exists during your entire editing session.
- MAIN cannot be deleted. (Only the contents, not the name, of this buffer can be deleted.)

The other buffer that EDT provides automatically is called the PASTE buffer. When you use the CUT keypad function, the deleted text goes into the PASTE buffer. Every time you perform a new CUT operation, EDT clears the PASTE buffer and replaces its contents with the newly deleted text.

The PASTE buffer has the following characteristics:

- The PASTE buffer is provided automatically when you begin your editing session.
- The PASTE buffer can be edited.
- The PASTE buffer can be loaded with the contents of another file. (Use the INCLUDE command.)
- The PASTE buffer cannot be deleted. (Only the contents, not the name, of this buffer can be deleted.)

2.6.1 Viewing Existing Buffers

If you want to see a list of the buffers in your editing session, enter the line-mode command SHOW BUFFER. If you are working in your MAIN buffer and you have not used the CUT command or created any new buffers, you get the following display:

```
=MAIN    23      lines
 PASTE   no      lines
```

(The number of lines, 23, used in this example is arbitrary.)

These two buffers, MAIN and PASTE, are always displayed when you enter the SHOW BUFFER command because they always exist and you cannot delete them.

If you have just used the CUT and PASTE commands to move three lines of text and you enter the SHOW BUFFER command again, the display indicates those three lines in the PASTE buffer:

```
=MAIN    20      lines
 PASTE   3       lines
```

The current buffer (the buffer in which you are working) is preceded by an equal sign. In the example above, MAIN is the current buffer. If you cannot remember which buffer you are working in, enter the SHOW BUFFER command.

Sometimes you will notice an asterisk (*) following the line count for the MAIN buffer. This asterisk indicates that EDT has not had a chance to read through the entire input file and has seen only as many lines as are indicated.

2.6.2 Creating Buffers

To create a buffer from keypad mode, press the GOLD key, followed by the COMMAND function. When EDT prompts you with *Command:*, enter the FIND command, then an equal sign followed immediately by the buffer name of your choice. A buffer name can contain any alphanumeric characters, but it must begin with a letter. The only punctuation character you can use in a buffer name is an underscore.

To create a buffer named BIRD, enter the following line after the command prompt:

```
Command: FIND=BIRD
```

When you enter this command, the screen clears except for the [EOB] symbol, indicating that the current buffer, BIRD, is empty. After you press RETURN, you can insert and edit text just as you would in the MAIN buffer.

To create a buffer from line mode, enter the FIND command, an equal sign, and the buffer name after the asterisk prompt. For example:

```
*FIND=BIRD
```

To return to the MAIN buffer, type FIND=MAIN. The buffer named BIRD remains intact until you exit from the file, when only the MAIN buffer is saved.

You can also create a buffer during a MOVE operation. For example, to move lines 1 through 17 to a new buffer named AINGE, enter the following line-mode command:

```
*MOVE 1:17 TO =AINGE
```

2.6.3 Deleting Buffers

Use the line-mode command CLEAR to delete buffers during an editing session. For example, to delete a buffer named WALTON, type the following command:

```
*CLEAR WALTON
```

You cannot delete the buffers MAIN and PASTE, only their contents. If you enter the command CLEAR MAIN, the contents of your MAIN buffer disappear, but the buffer name remains. When you exit from EDT, the contents of all the existing buffers, except MAIN, are deleted.

2.6.4 Copying Text from One Buffer to Another Buffer

To copy text from one buffer to another, enter the COPY command. For example, to copy the contents of a buffer named BIRD to a buffer named PARISH, enter the following command:

```
*COPY =BIRD TO =PARISH
```

When you complete this operation, the current buffer will be PARISH.

2.6.5 Copying Text from a File Into a Buffer

To copy text from a file outside of EDT into a buffer, use the INCLUDE command. For example, to copy the contents of a file named JOHNSON.DAT to your MAIN buffer, type the following command:

```
*INCLUDE JOHNSON.DAT =MAIN
```

2.6.6 Copying Text from a Buffer to a File

To copy text from a buffer to a file, use the WRITE command. You can use this command to copy text to a new file without affecting your editing session. For example, the following command puts a copy of lines 23 through the end of the current buffer into a file that you name MCHALE.DAT:

```
*WRITE MCHALE.DAT 23:END
```

The following table lists the three commands you need to copy text between buffers and external files:

Direction	Command
One buffer to another	* COPY =other_buffer TO =buffer
External file to buffer	* INCLUDE alien_file.typ =buffer
Buffer to external file	* WRITE alien_file.typ n:n

2.7 Recovering a Lost Editing Session

While you are editing or inserting text, EDT is keeping track of every keystroke you enter at your terminal. EDT records this information in a file called a journal file. Unless you specify otherwise, this journal file is deleted as soon as you enter the EXIT or QUIT command. However, when you experience a system interruption, the journal file is not deleted.

The journal file does not contain a version of your text. Rather, it contains a record of the keystrokes you entered during the session. By combining the journal file with the text that you had at the beginning of your session, you can recover your session to a point just before the interruption.

Note: Sometimes, the last few keystrokes are missing. This is normal. No work from earlier in your session will be omitted.

The following command line demonstrates how to recover a file named TRAVEL.SDML after the editing session has been interrupted:

```
$ EDIT/RECOVER TRAVEL.SDML
```

When you work with journal files, you will notice that they have a file type of JOU. The file name is the same as the file you were editing. The journal file for TRAVEL.SDML is TRAVEL.JOU. And, the journal file for ASCONA.LIS is ASCONA.JOU.

To recover an editing session you had begun with the command EDIT JONES.DAT, enter the following command line:

```
$ EDIT/RECOVER JONES.DAT
```

Notice that you enter the name of the file you were editing, not the name of the journal file. If you enter the JOU file type, you edit the journal file. After you enter the EDIT/RECOVER command, EDT replays the editing session. When it finishes, you can continue editing.

You can start an editing session and interrupt it to see how the /RECOVER qualifier works with the following steps:

1. Invoke EDT to create a file named NBA.FUN.
2. Insert text into NBA.FUN and perform a number of edits.
3. Press CTRL/Y to end the editing session abruptly. (You will be at the DCL command level at this point.)
4. Enter the DCL command DIRECTORY to see the newly created journal file (NBA.JOU).

5 Invoke EDT again, using the /RECOVER qualifier to recover your lost session, by entering the command line:

```
$ EDIT/RECOVER NBA.FUN
```

6 Be sure that your last few keystrokes were recovered and continue to edit the file. Then enter the EXIT or QUIT command to terminate your editing session.

7 Enter the DCL command DIRECTORY again. The journal file, NBA.JOU, is not there because you have exited normally from EDT.

2.8 Creating Columns and Layered Text

You can use EDT's tabbing facility in keypad mode to format text in two ways: in columns of eight and in layers.

2.8.1 Creating Columns of Eight

EDT has automatic tabs set for columns eight characters wide. Therefore, when you press TAB, your text moves to the nearest preset tab stop: column 9, 17, 25, 33, 41, 49, and so on. TAB always moves your text to the right regardless of EDT's current direction. If you want to move text back toward the left margin, use the DELETE key.

The following example demonstrates how to use TAB. Enter the following lines of text:

```
Tab each word in this sentence?
Tab each word in this sentence?
Tab each word in this sentence?
```

Move the cursor to the first letter of the first word and press TAB. Then move the cursor to the first letter of the second word and press TAB; then, to the third word and press TAB, and so on. When you are done, the sentences look as follows:

```
Tab     each    word    in      this    sentence?
Tab     each    word    in      this    sentence?
Tab     each    word    in      this    sentence?
```

The words are organized in columns of eight characters. As long as you want to arrange text in columns of eight, you can use TAB.

2.8.2 Creating Layers of Text

If you have layered text, such as an outline, you can use the line mode command SET TAB along with the TAB key to format the lines. When you enter the SET TAB command and specify a value (for example, 15), you can tab your first item over 15 spaces by pressing TAB. The following example shows a list of colors:

```
red
green
beige
blue
white
```

After you enter the following line mode command, move the cursor to the first letter of each word in your list of colors and press TAB:

```
*SET TAB 15
```

Your tabbed list is shown below:

```
red
green
beige
blue
white
```

The two most important things to remember when you are using SET TAB are the following:

- The SET TAB value only affects the first indentation on a line.
- The cursor must be at the left edge of the screen on the line being moved.

To format an outline, use the line-mode command SET TAB along with the keys CTRL/E, CTRL/D, and TAB to change the indentation level of your lines.

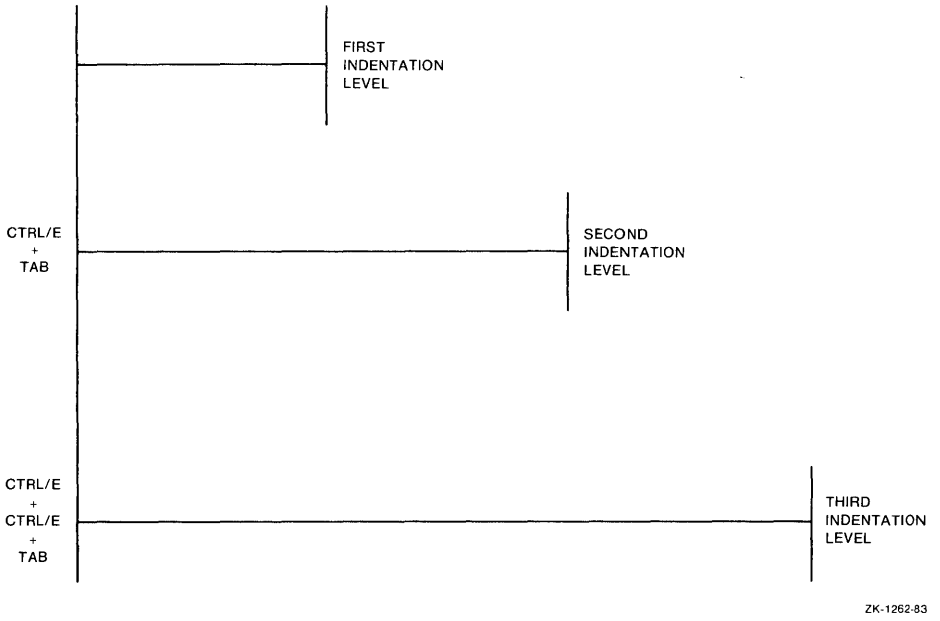
Enter the SET TAB command to set the indentation level. After you enter the SET TAB command, you can move the first line to be indented to the first indentation level by pressing TAB. To move a line of text to the second indentation level, press CTRL/E. Then press TAB to actually move the line of text. You increase the indentation level of the text by pressing CTRL/E, and you move the text to the desired level by pressing TAB.

For example, enter the command SET TAB 10, press TAB, and type the word TEN. Press RETURN. Then press CTRL/E, followed by TAB, and type the word TWENTY. Press RETURN. Next, press CTRL/E again, TAB, and type the word THIRTY. Press RETURN. Finally, press CTRL/E, TAB, and type the word FORTY. Your display will look as follows:

```
TEN
          TWENTY
                    THIRTY
                              FORTY
```

Figure 2–5 shows the keys TAB and CTRL/E with their corresponding indentation levels.

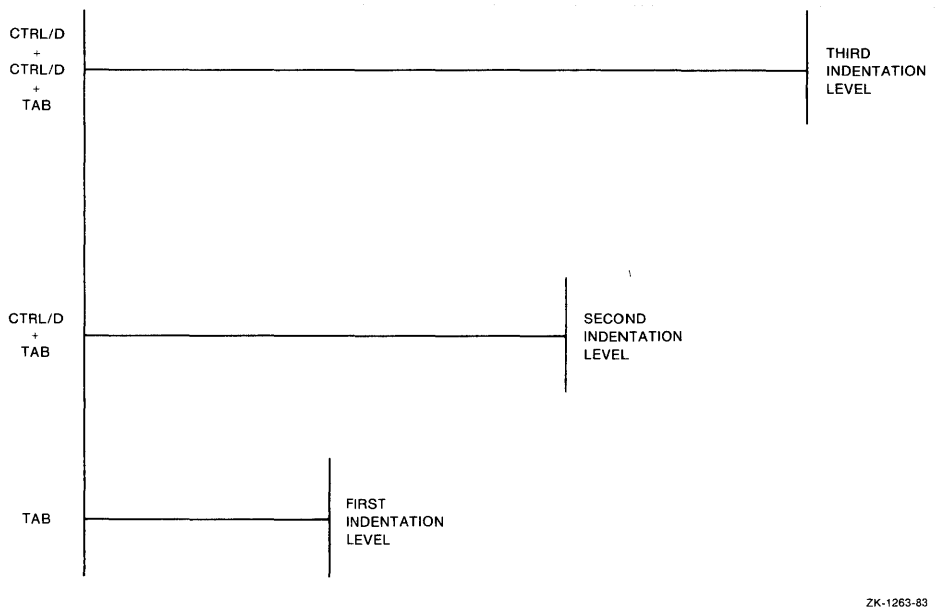
Figure 2–5 Using CTRL/E to Increase the Indentation Level



Use CTRL/D to select the next lower indentation level. CTRL/D does the opposite of CTRL/E. It lowers the indentation level count by the value of SET TAB. It does not move text back toward the left margin; rather, it reduces the number of spaces that text is moved to the right.

Figure 2–6 shows the keys TAB and CTRL/D with their corresponding indentation levels.

Figure 2–6 Using CTRL/D to Decrease the Indentation Level



The following sampling shows part of an outline before you indent the lines:

```
I.
A.
B.
1.
C.
II.
```

Now perform the following steps:

1. Choose the number of spaces you want to indent (for example, 5).
2. Enter the SET TAB command followed by that number.
3. Move the cursor to the Roman numeral I and press TAB.
4. Move the cursor to the letter A, press CTRL/E, and press TAB.
5. Move the cursor to the letter B and press TAB.
6. Move the cursor to the number 1, press CTRL/E, and press TAB.
7. Move the cursor to the letter C, press CTRL/D, and press TAB.
8. Move the cursor to the Roman numeral II, press CTRL/D, and press TAB.

The outline is now formatted as follows:

```
I.
     A.
     B.
          1.
     C.
II.
```

The following shows a longer outline and the tabbing keys you press to create it:

```
                    Malcolm's Family Tree

TAB
     I. Malcolm
CTRL/E + TAB
               A. Ian
TAB
               B. Tatiana
TAB
               C. Lars
CTRL/E + TAB
                    1.  Stephanie
TAB
                    2.  Brian

CTRL/D + CTRL/D + TAB
     II.Frederick
CTRL/E + TAB
               A. Louis
CTRL/E + TAB
                    1.  Elsa

CTRL/D + CTRL/D + TAB
     III.Louisa
CTRL/E + TAB
               A. Jeffrey
TAB
               B. Andrew

CTRL/D + TAB
     IV.Sonya
CTRL/E + TAB
               A. Beth
TAB
               B. Celia
TAB
               C. Franklin
CTRL/E + TAB
                    1. Martin
CTRL/E + TAB
                         a) Stephan
CTRL/D + CTRL/D + TAB
               D. Martha
```

Note: Only two keypad tabbing functions actually move text: TAB and CTRL/T. The remaining three functions (CTRL/A, CTRL/D, and CTRL/E) determine the effect that the first two have on your text. CTRL/A and CTRL/T are discussed in the following sections.

2.8.3 Using CTRL/A to Indent Text

You can use CTRL/A to move a line of text to more than one indentation level without having to enter repeated tab increments (CTRL/E). There are three steps to the process:

1. Enter a SET TAB command, specifying a value.
2. Move the cursor to one of the indentation levels. The indentation level must be a character position that is a multiple of the value you specified with SET TAB. For example, you could move the cursor to 20 or 25 if the tab size is 5.
3. Press CTRL/A.

Once you choose the indentation level you want, and establish the position with CTRL/A, you can use TAB to indent the line of text to that place.

If you want to indent text to a different indentation level after establishing a position with CTRL/A, you can either use CTRL/A again to reset the position or use CTRL/E and CTRL/D to adjust the indentation level up or down.

Note: If the cursor position is not evenly divisible by the SET TAB value when you press CTRL/A, EDT displays the message *Could not align tabs with cursor.*

2.8.4 Using CTRL/T to Indent Groups of Lines of Text

Use CTRL/T to indent groups of lines using the current SET TAB value. The CTRL/T function is similar to CTRL/A, CTRL/D, and CTRL/E in the following ways:

- CTRL/T has no effect unless you have specified a value with the SET TAB command.
- CTRL/T moves entire lines of text.

On the other hand, CTRL/T differs from these other CTRL tabbing functions in the following ways:

- When you press CTRL/T, you do not press TAB to make the block of text move to the right. The text moves automatically.
- The block of text is indented only the amount of the value established with the SET TAB command, without regard to the current indentation level.

To allow CTRL/T to work correctly in EDT, you must disable DCL control over CTRL/T. By default, DCL displays process statistics when you press CTRL/T. To disable DCL control, enter the following command at the DCL command level:

```
$ SET NOCONTROL=T
```

To indent text using CTRL/T, use the SELECT command to choose a range of lines, and then press CTRL/T. EDT indents the range of lines only to the current SET TAB value. To indent text by multiples of the SET TAB value, use the GOLD repeat function described in Section 2.2.2.

Note: All the functions, except for TAB, require that a SET TAB value be established for your editing session. EDT's default is SET NOTAB. As long as NOTAB is in effect, pressing CTRL/A, CTRL/D, CTRL/E, or CTRL/T has no effect on the text you are editing.

2.8.5 Looking at the Indentation Level

Use the SHOW TAB command to see the value specified with the SET TAB command and the current indentation level. For example, if you set your tab size to 15 and press CTRL/E twice, EDT displays the following information when you enter the SHOW TAB command:

```
*SHOW TAB
tab size 15; tab level 3
```

For more information on the SHOW TAB command, see the *VAX EDT Reference Manual.*

2.9 Defining Keys

When you define or redefine a key, you change the function of that key. You can use combinations of nokeypad commands or existing keypad functions to dictate what a key does.

Some advantages of defining keys are as follows:

- You can take advantage of different entities available in nokeypad mode. For example, you can define keys to delete sentences, paragraphs, and pages.
- You can combine several nokeypad commands into one key definition. For example, you can define a key to find a word, delete it, and substitute several words in its place.
- You can access nokeypad commands that do not exist in keypad editing, such as SHL, SHR, CHGL, CHGU, and DATE.

2.9.1 Definition Procedure

EDT provides two commands to define keys: CTRL/K and DEFINE KEY. The line-mode command DEFINE KEY allows you to put key definitions in startup command files and EDT macros, as well as to create key definitions during your EDT session. You can only use CTRL/K when working in keypad mode.

2.9.1.1 Using CTRL/K to Define a Key

To define a key in keypad mode, do the following:

1. Press CTRL/K.
2. Press the key (or key sequence) you want to define (for example, GOLD/A or CTRL/H).
3. Type in the key definition. (Either enter a string of nokeypad commands or press a sequence of keypad function keys, or both.)

4 Type a period.

5 Press the ENTER key.

To define keys, you need to be familiar with nokeypad commands. See the *VAX EDT Reference Manual* for more information.

The following example uses D and SEN. D is the delete command and SEN refers to a string of characters (a sentence) enclosed by delimiters. This example defines CTRL/A to delete the sentence at your cursor.

```
CTRL/K
Press the key you wish to define
CTRL/A
Now enter the definition terminated by ENTER
DSEN.ENTER
```

The following example demonstrates how to define a key in keypad mode to draw a vertical line:

```
CTRL/K
Press the key you wish to define
CTRL/V
Now enter the definition terminated by ENTER
22(+VI|^Z).ENTER
```

In this example, CTRL/K tells EDT to define the key combination CTRL/V. The number 22 tells EDT to execute the commands in the parentheses 22 times. The definition for the Down arrow is +V. The command I| tells EDT to put the vertical bar character into your text. The insert command is terminated by ^Z. The period ensures that the command will take effect as soon as you press the key.

Note: The ^Z is typed as a circumflex followed by the letter Z.

2.9.1.2 Using the DEFINE KEY Command

The DEFINE KEY command allows you to assign a different command to a key or assign a string of text to a key. To assign a command to a key, use the following syntax with the DEFINE KEY command:

```
DEFINE KEY key name AS "command(s)."
```

Key name is the name of the key or its keypad number, for example, DEFINE KEY GOLD F or DEFINE KEY 3. (See Figure 2–1 for keypad diagrams showing key names and keypad numbers. Note that you use the small number in the lower right corner on the keypad diagram. For example, you would use the number 11 to redefine keypad key PF3.) Type the *command(s),* which is the actual definition, completely in nokeypad syntax. Follow the nokeypad commands with a period and enclose the definition (and the period) in quotation marks or apostrophes.

Note: To use a key you have defined with the DEFINE KEY command, you must be in keypad mode. To enter keypad mode, see Section 2.1.4.

The steps for defining a key using the DEFINE KEY command are outlined below:

1 Enter line mode. (See Section 2.1.4 if you need help.)

2 Enter the DEFINE KEY command.

3 Type the key name you want to define followed by the word *AS* (for example, DEFINE KEY CONTROL B AS . . .).

4 Enter a string of nokeypad commands followed by a period. Enclose the string and the period in quotation marks (for example, DEFINE KEY CONTROL B AS *D+NL.*).

5 Press RETURN.

6 Exit line mode. (See Section 2.1.4 if you need help.)

To assign a string of text to a key, use the following syntax with the DEFINE KEY command:

```
DEFINE KEY key-name AS "IInsert your text here. ^Z."
```

In this example, the command to insert text is enclosed in quotation marks. The following components are in the quotation marks:

- The INSERT command (I).
- Text to be inserted.
- ^Z (to complete the insertion). Remember to enter the ^Z by typing the circumflex followed by the letter Z.
- A period to enable the command to take effect as soon as you press the key.

When you use the DEFINE KEY command to define CTRL keys, for example CTRL/A, type the word CONTROL:

```
DEFINE KEY CONTROL A AS "command."
```

When you want to define the GOLD key, for example GOLD 5, type the word GOLD:

```
DEFINE KEY GOLD 5 AS "command."
```

When you want to redefine a function key on the LK201 keyboard, type the word FUNCTION, as in the following example:

```
DEFINE KEY FUNCTION 29 AS "command."
```

To create key definitions, you can consolidate several keypad functions into one key definition. The following table lists the key names you use when you define keys:

Key name	VT100	VT52	LK201
20	PF1	red key	PF1
10	PF2	blue key	PF2
11	PF3	black key	PF3
17	PF4	-	PF4
0-9	keypad 0-9	keypad 0-9	keypad 0-9
16	period key	period key	period key
19	comma key	comma key	comma key
18	minus key	minus key	minus key

Key name	VT100	VT52	LK201
21	ENTER key	ENTER key	ENTER key
FUNCTION 1	-	-	Find
FUNCTION 2	-	-	Insert Here
FUNCTION 3	-	-	Remove
FUNCTION 4	-	-	Select
FUNCTION 5	-	-	Previous Screen
FUNCTION 6	-	-	Next Screen
FUNCTION 28	-	-	Help
FUNCTION 29	-	-	Do

You can combine keypad functions to correct typing errors. For example, to correct reversed letters, you delete a character, press the Right arrow key, and undelete the character. Suppose you type the word *social* as *oscial*. You would move the cursor to the *o*, delete it, press the Right arrow key to move the cursor to the *c*, and undelete the *o*. The following table displays these steps translated into nokeypad commands:

Function	Nokeypad Commands
delete a character	D+C
move cursor one character to the right	+C
undelete a character	UNDC

To define CTRL/R to do all three commands at once (thus saving you keystrokes), enter the following line:

```
*DEFINE KEY CONTROL R AS "D+C +C UNDC."
```

Now, when you press CTRL/R, EDT will reverse the two characters at the cursor.

2.9.2 Which Keys Can Be Defined

You can define all keypad keys, all function keys, and GOLD with either a keypad key or a function key.

```
DEFINE KEY 8 AS "command."
DEFINE KEY GOLD 8 AS "command."
DEFINE KEY FUNCTION 34 AS "command."
```

You can define GOLD/keyboard keys. A GOLD/keyboard key refers to the combination of GOLD with a key on the main keyboard.

```
DEFINE KEY GOLD A AS "command."
```

When you define a GOLD/keyboard key sequence, enclose the following symbols within quotation marks (except for quotation marks, which you must enclose within apostrophes):

exclamation point (!)
percent sign (%)

apostrophe (')
quotation marks (")

For example, you can define the following GOLD sequence to exit from EDT in one step, without having to press several keys and type EXIT:

```
*DEFINE KEY GOLD "!" AS "EXT EXIT"
```

You can also define keys using the control key (CTRL) with letter keys as well as with brackets ([ ]), a backslash (\), a tilde (~), a circumflex (^), and an underscore (_).

```
DEFINE KEY CONTROL A AS "command."
DEFINE KEY CONTROL ] AS "command."
```

You can also define GOLD CONTROL keys (for example, DEFINE KEY GOLD CONTROL AS *command*). When you use GOLD and CONTROL, press the GOLD key first, then hold down the CTRL key while you press the keyboard key.

Do not define the following key combinations:

CTRL/C
CTRL/O
CTRL/Q
CTRL/S
CTRL/T (unless you SET NOCONTROL=T)
CTRL/X
CTRL/Y
GOLD/CTRL/C
GOLD/CTRL/O
GOLD/CTRL/Q
GOLD/CTRL/S
GOLD/CTRL/X
GOLD/CTRL/Y
GOLD/digit (keyboard keys 0 through 9)
GOLD/minus sign (-)

You can redefine the following key combinations, but be aware of their default functions:

CTRL/H (BACKSPACE)
CTRL/I (TAB)
CTRL/J (LINEFEED)
CTRL/M (RETURN)

2.9.3 Saving Key Definitions

Once you define a key, it stays defined throughout your editing session until you EXIT, QUIT, or redefine the key. You can save key definitions, however, by including them in your startup command file or in a macro file. The following is a sample section from a startup command file:

```
DEFINE KEY CONTROL R AS "EXT INCLUDE ?'INCLUDE FILE: '."❶
DEFINE KEY CONTROL G AS "EXT FIND=MAIN."❷
DEFINE KEY CONTROL L AS "EXT EXIT."❸
DEFINE KEY CONTROL N AS "EXT QUIT."❹
```

When you press:

❶ CTRL/R, EDT prompts you for a file to include.

❷ CTRL/G, EDT puts you at the top of the MAIN buffer.

❸ CTRL/L, EDT exits your editing session.

❹ CTRL/N, EDT quits your editing session.

See Section 2.11 for more information about saving key definitions in startup command files.

2.10 Using Macros

EDT allows you to define a series of line-mode commands as a macro. The following sections explain how and why you would want to do this.

2.10.1 Introduction

A macro is a sequence of line-mode commands that you can execute when you type the macro name. You use the DEFINE MACRO command to add the macro name to the list of valid line-mode commands for the duration of your editing session.

2.10.2 Creating a Macro

To create a macro, assign a name—for example, CONCERTS—with the DEFINE MACRO command:

```
*DEFINE MACRO CONCERTS
```

Next, enter a buffer of the same name:

```
*FIND=CONCERTS
```

Then, type the list of line-mode commands; in this case, use the INSERT command to create a list of dates. Type the INSERT command followed by the semicolon, one space, and then the text to be inserted. End the list with CTRL/Z to invoke the line-mode prompt. (To return to the current line in the MAIN buffer, enter the FIND=MAIN command.)

```
*INSERT
            INSERT; February 11, 1988
            INSERT; May 2, 1988
            INSERT; December 9, 1988
            INSERT; January 20, 1989
            INSERT; March 20, 1989
            INSERT; April 7, 1989
            INSERT; August 25, 1990
^Z
[EOB]
*FIND=MAIN
```

EDT executes that series of commands whenever you type the name of the macro as a line command, as shown in the following example:

```
*CONCERTS
*TYPE WHOLE

    1       February 11, 1988
    2       May 2, 1988
    3       December 9, 1988
    4       January 20, 1989
    5       March 20, 1989
    6       April 7, 1989
    7       August 25, 1990
```

2.10.3 Macro Functions

Although macros can contain only line-mode commands, they are able to perform a variety of functions, including the following:

- Macros can contain SET commands to tailor your editing session.
- Macros can contain a series of line-mode commands.
- Macros can contain a series of key definitions.
- Macros can define other macros or call up macros from external files.

2.10.4 Comparing Macros to Startup Command Files

Both macros and startup command files contain line-mode commands, but macros are more flexible. A startup command file (see Section 2.11) is executed once at the beginning of an editing session. But, you can create and access many different macros containing different sets of commands during one editing session just by typing the macro name.

2.10.5 Saving Macros

Macros can be created at any time during your editing session and can be used for the remainder of the session. Since they are located in buffers, they disappear as soon as you leave EDT. If you create a macro that you want to save for other EDT sessions, you can use the EDT command WRITE to put a copy of the macro in an external file. When you need that macro again, use the INCLUDE command to copy the macro into a buffer and then establish the macro name as a command with the DEFINE MACRO command.

The following example illustrates this process:

```
$ EDIT FUN.FUN
*DEFINE MACRO HEADING ❶
*FIND=HEADING ❷
*INSERT
INSERT;  Name: ❸
INSERT;  Address:
INSERT;  Number:
INSERT;  Date:
^Z
[EOB]
*WRITE HEADING.MAC ❹
DISK$:[HARTWELL]HEADING.MAC;1 4 lines
*EXIT
   .
   .
   .
$ EDIT WHY.NOT
*HEADING ❺
 ^
Unrecognized command
*FIND=HEADING
*INCLUDE HEADING.MAC ❻
*DEFINE MACRO HEADING ❼
*FIND=MAIN ❽
*HEADING ❾
*TYPE WHOLE

    Name:
    Address:
    Number:
    Date:
```

❶ The DEFINE MACRO command makes HEADING a valid line-mode command.

❷ You enter a buffer named HEADING.

❸ You use four INSERT commands to create the text you want to use.

❹ The WRITE command puts a copy of the macro in an external file.

❺ In a new editing session, EDT does not recognize the macro named HEADING until you define the macro with the DEFINE MACRO command.

❻ The INCLUDE command brings the macro in from an outside file and puts it in the buffer named HEADING.

❼ The DEFINE MACRO command makes HEADING a valid line-mode command.

❽ You return to the MAIN buffer. (You must leave the buffer containing the macro before you use the macro as a line-mode command.)

❾ The HEADING command now functions as a line-mode command.

Note: Macros override line-mode commands. If you create a macro with the same name as an existing line-mode command, EDT performs the macro, not the line-mode command. As long as the macro is in effect, it overrides the default line-mode command. If you need to reestablish the default use of the line-mode command, use the CLEAR command to eliminate the buffer with the same name.

2.10.6 Including Specifiers in a Macro

When you use line-mode commands that take specifiers, you must include the appropriate specifier in the macro text. For example, if you create a macro that includes a substitute command, you must supply the strings for EDT to use with that command.

You cannot use the SUBSTITUTE command in a macro and expect to enter the strings after typing the macro name.

The same rule applies when you use a command like TYPE in your macro. If you want to have a command that automatically types the current line and the next 9 lines, you can create the macro XTYPE to be:

```
TYPE . THRU +9
```

You cannot, however, create a command that lets you choose the number of lines to type and allows you to enter the final digit from the terminal.

2.11 Startup Command Files

A startup command file contains EDT line-mode commands that are executed when you invoke EDT—before you receive control of the editor. You can use startup command files to customize your EDT sessions. You can create startup command files in either your current or main directory. Some of the line-mode commands that a startup command file might contain are:

- DEFINE KEY commands—To redefine the function invoked by a function key, a keypad key, or a control character while you are editing in keypad mode.
- DEFINE MACRO commands—To associate a name with a sequence of line editing commands stored in a text buffer. You can then invoke the sequence by entering the macro name in response to the line editing asterisk prompt.
- INCLUDE commands—To bring text from a file into a text buffer. You might use them to load macros into a buffer, or to fill a buffer with text that you often use.
- SET commands—To establish EDT operating parameters. For example, SET TAB establishes the increment for structured tabs, and SET MODE CHANGE invokes keypad mode.

When you begin an editing session, EDT automatically searches your current directory for a startup command file named EDTINI.EDT. EDT then processes the commands in that file before giving control to you. For example, if you create an EDTINI.EDT file and include the following line-mode commands, EDT automatically executes these commands the next time you invoke EDT:

```
$ EDIT EDTINI.EDT [RET] ❶
Input file does not exist
[EOB]
*INSERT [RET]
        SET QUIET [RET] ❷
        SET NONUMBERS [RET] ❸
[CTRL/Z]
*EXIT
$ EDIT LIST.DAT ❹
```

❶ EDT is invoked to create the file named EDTINI.EDT.

❷ SET QUIET suppresses the bell sound when EDT prints an error message (in keypad mode).

❸ SET NONUMBERS suppresses line numbers in line mode.

❹ Now when you invoke EDT to edit a file named LIST.DAT, the commands in EDTINI.EDT execute before you gain control of the editor.

Because EDTINI.EDT is the default startup command file, you do not need to name it in the command line. 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.

The following example demonstrates how to create a two-line startup command file:

```
$ EDIT SETUP.EDT
Input file does not exist
[EOB]
*INSERT
        SET MODE CHANGE
        SET LINES 5
CTRL/Z
[EOB]
*EXIT
```

In this example, the name of your startup command file is SETUP.EDT. It contains two SET commands: SET MODE and SET LINES. SET MODE CHANGE causes EDT to begin your editing session in keypad mode, and SET LINES 5 limits the display of text on your screen to five lines.

Use the qualifier /COMMAND= with the EDIT command to indicate the startup command file you want EDT to use. The following command line tells EDT to execute the line-mode commands in the startup command file named SETUP.EDT to edit a file named FUN.FUN:

```
$ EDIT/COMMAND=SETUP.EDT FUN.FUN
```

The following startup command file contains two SET commands, three DEFINE KEY commands, and one DEFINE MACRO command:

```
SET WRAP 60 ❶
SET SEARCH EXACT ❷
DEFINE KEY GOLD W AS "CHGUW." ❸
DEFINE KEY CONTROL B AS "EXT INCLUDE ?'INCLUDE FILE: '." ❹
DEFINE KEY CONTROL G AS "EXT FIND=MAIN." ❺
FIND=GENERAL ❻
INSERT ;SET SEARCH GENERAL ❼
DEFINE MACRO GENERAL ❽
FIND=MAIN ❾
```

❶ Limits the line length for inserting text in keypad mode to 60 characters.

❷ Causes EDT to match exactly the case and diacritical marks (for example, grave accent or circumflex) of letters in search strings.

❸ Defines GOLD/W to change all lowercase letters in a word to uppercase.

❹ Defines CTRL/B to extend the INCLUDE line command to keypad mode and issue a prompt.

❺ Defines CTRL/G to return to the MAIN buffer.

❻ Creates a buffer named GENERAL and moves there.

❼ Inserts the line-mode command SET SEARCH GENERAL.

❽ Adds GENERAL to the list of valid line mode commands.

❾ Returns to the first line of the MAIN buffer.

3 Formatting Files with DSR

DIGITAL Standard Runoff (DSR) is a program that formats text. DSR commands and flags, which are entered into text, allow you to specify the final appearance of a document. Neither the DSR commands nor flags appear in your final document.

3.1 Introduction

You can use DSR to determine your page size, create justified or unjustified right margins, place space between lines, and form lists. DSR can center titles, supply page numbers, and organize your text into sections and chapters. You can also use DSR to create a table of contents and an index. For a complete list of all the available DSR commands, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.1.1 Using DSR Defaults

When you process a file using DSR, a certain set of DSR commands affect your file by default. You do not need to add these DSR commands to your file to achieve the desired effect because DSR does it automatically. If you do not want your file to be affected by these DSR default commands, you must disable them. See the *VAX DIGITAL Standard Runoff Reference Manual* for a complete list of the default commands provided by DSR and the commands you need to disable them. The most common of the defaults are discussed here.

When you use DSR to process a file, your output file looks different from your input file because DSR provides the following standard format defaults:

- A text width of 70 characters and a text length of 58 lines.
- Sequential page numbering (omitting a page number for the first page).
- A left margin setting of 0 and a right margin setting of 70.
- Single spacing.
- Tab stops at character positions 9, 17, 25, 33, and so on.
- Filling: DSR fills each line with as many words as possible until the addition of another complete word would exceed the right margin.
- Justification: DSR adds spaces between words to expand each line exactly to the right margin, making the right margin even, or justified.

For example, if you create a file containing the text in the next example and then process the file using DSR, the resulting output file will be 70 characters wide (beginning at character position 0 and ending at character position 70), single spaced, filled, and justified.

Formatting Files with DSR

3.1 Introduction

Input File:

```
Children learn to speak through imitation.

Researchers find that infants first make random sounds.
Naturally, their parents praise them for sounds that are
culturally meaningful, so children begin to
repeat a subset of sounds. In time, babies deliberately
imitate the sounds made by their parents.
As they grow, children expand their language beyond the
family. They incorporate language learned
from friends, neighbors, television, radio, and books.
```

Output File:

```
Children learn to speak through imitation. Researchers find that
infants first make random sounds. Naturally, their parents praise
them for sounds that are culturally meaningful, so children begin to
repeat a subset of sounds. In time, babies deliberately imitate the
sounds made by their parents. As they grow, children expand their
language beyond the family. They incorporate language learned from
friends, neighbors, television, radio, and books.
```

3.1.2 Including DSR Commands

To use DSR, you create a file, insert text, and include DSR commands in the appropriate places. DSR creates an output file with a file type of MEM (for example, TEXT.MEM). The MEM file contains the formatted text, which you can then print or display on your terminal.

Use the following steps to format text using DSR:

What You Do	How You Do It
Create a file	$ EDIT list.rno Input file does not exist [EOB] *
Insert text with DSR commands	.LIST .LIST ELEMENT;apples .LIST ELEMENT;plums .LIST ELEMENT;oranges .LIST ELEMENT;bananas .END LIST
Process file	$ RUNOFF list.rno
Display formatted file	$ Type LIST.MEM **1** apples **2** plums **3** oranges **4** bananas

3.1.3 Looking at DSR Commands

All DSR commands are English words preceded by a control flag, which is often a period (.), as in the following examples:

```
.BLANK
.CENTER
```

The .BLANK command inserts a blank line. The .CENTER command centers the text immediately following it on the same line. You can abbreviate all DSR commands. The abbreviations for .BLANK and .CENTER are .B and .C, respectively. For a complete list of valid DSR command abbreviations, see the *VAX DIGITAL Standard Runoff Reference Manual.*

You end a DSR command with a terminator. Commands are most commonly terminated by the end of a line, but you can also use a semicolon (;) as a terminator, or you can terminate a command and begin another one with a period. You can also combine terminators. For example, terminate a command with a semicolon and another command.

DSR commands can be typed in uppercase, lowercase, or a combination of uppercase and lowercase.

The following table shows four ways to terminate a DSR command:

DSR Command	Terminator
.BLANK2	(end of line)
.BLANK2;	semicolon
.BLANK2.CENTER;text	another command
.BLANK2;.CENTER;text	semicolon and another command

On any line containing a DSR command, the command must begin in column 1, unless it follows other commands on the same line. Depending on the particular command, a line containing a command can contain additional commands or text.

The following example demonstrates how to use the .BLANK and .CENTER commands to format text in a file called DIETING.RNO:

```
.CENTER;Twelve Days of Dieting ❶
.BLANK3 ❷
On the twelfth  day of dieting, Millitsa gave to me,
.BLANK ❸
Twelve hot fudge sundaes,
.BLANK
Eleven gooey doughnuts,
.BLANK
Ten cherry cheese cakes,
.BLANK2 ❹
Nine lady fingers,
.BLANK ❺
Eight date nut muffins,
.BLANK
Seven oatmeal cookies,
.BLANK
Six bags of corn chips,
.BLANK2
.CENTER;FIVE COFFEE RINGS, ❻
.BLANK3
Four sticky buns,
.BLANK
Three candy bars,
.BLANK
Two marbled cakes,
.BLANK
And a pizza with pepperoni.
```

❶ The .CENTER command terminated by a semicolon centers the title.

❷ The .BLANK command followed by the number 3 creates three blank lines.

❸ The .BLANK command creates one blank line.

❹ The .BLANK command followed by the number 2 creates two blank lines.

❺ The .BLANK command creates one blank line.

❻ The .CENTER command terminated by a semicolon centers the line of text.

When you enter the command RUNOFF DIETING.RNO, DSR formats your text to look like this:

```
                    Twelve Days of Dieting

On the twelfth day of dieting, Millitsa gave to me,

Twelve hot fudge sundaes,

Eleven gooey doughnuts,

Ten cherry cheese cakes,

Nine lady fingers,

Eight date nut muffins,
```

```
Seven oatmeal cookies,
Six bags of corn chips,
                                FIVE COFFEE RINGS,

Four sticky buns,
Three candy bars,
Two marbled cakes,
And a pizza with pepperoni.
```

3.1.4 Running DSR to Process Your Files

After you add DSR commands to your file and exit from the editor, you are ready to run DSR. To run DSR, you enter the RUNOFF command followed by the name of the file you want to process. For example, to process a file named FUN.FUN, enter the following command line:

```
$ RUNOFF FUN.FUN
```

If you process a file with a file type of RNO, you only need to enter the file name, not the file type. For example, to process a file named FUN.RNO, enter the following command line:

```
$ RUNOFF FUN
```

3.1.4.1 Using Qualifiers with the RUNOFF Command

DSR provides twenty different qualifiers that you can use with the RUNOFF command. These qualifiers are as follows:

/BACKSPACE
/BOLD
/CHANGE_BARS
/CONTENTS
/DEBUG
/DOWN
/FORM_SIZE
/INDEX
/LOG
/MESSAGES
/NONSPACING_UNDERLINE
/OUTPUT
/PAGES
/PAUSE
/RIGHT
/SEPARATE_UNDERLINE
/SEQUENCE
/SIMULATE
/UNDERLINE_CHARACTER
/VARIANT

Section 3.13 explains how to create a table of contents and an index by using the qualifiers /CONTENTS and /INDEX. All the qualifiers are discussed in detail in the *VAX DIGITAL Standard Runoff Reference Manual*. The next section discusses how to use one of the qualifiers, /MESSAGES.

Using the /MESSAGES Qualifier

If DSR finds any errors during processing, it prints an error message (both on the screen and in the MEM file) telling you what the error is and where it is located in both the input and the output files. You can then correct the RNO file (by using a text editor) and reprocess the file.

You can use the /MESSAGES qualifier to specify where you want DSR to display error messages. The options are the following:

OUTPUT	Sends error messages only to the output file (MEM file)
USER	Sends error messages only to the terminal (screen)

The default is /MESSAGES=(OUTPUT,USER), which sends messages to the output file and displays them on the terminal. You can prevent error messages from going either to the output file or to the terminal, but you cannot suppress them entirely.

The following command causes DSR to display error messages in the MEM file, not at the terminal:

```
$ RUNOFF FUN/MESSAGES=OUTPUT
```

The following command causes DSR to display error messages at the terminal, not in the MEM file:

```
$ RUNOFF FUN/MESSAGES=USER
```

3.1.5 Stripping MEM Files of Carriage-Return/Line-Feed Symbols

When you edit a MEM file produced by DSR, you will notice that the file contains carriage-return/line-feed symbols. The format of the symbols is dependent on the text editor you use. You can also have a file containing these symbols when you add a MEM file to an RNO file. In either case, you may want to strip the MEM file of these symbols using a text editor. Removing the symbols does not break your file. See the documentation for your text editor for instructions on taking carriage-return/line-feed symbols from the file.

3.2 Formatting Lists

You need to know only the following three DSR commands to format a numbered list:

- .LIST
- .LIST ELEMENT
- .END LIST

Begin your list with the .LIST command. The .LIST command causes DSR to start a list. When DSR formats a list, it indents the left margin, puts a blank line before the first list item, after the last item, and between each item in the list. DSR also numbers (or otherwise marks) the items.

Precede each item in your list by the .LIST ELEMENT command. Use a semicolon to separate the list item from the command or put the list item on the following line. End your list with the .END LIST command.

The following example shows how to format a list using DSR commands:

Input file (LIST.RNO)

```
.LIST
.LIST ELEMENT;grosbeak
.LIST ELEMENT;gold finch
.LIST ELEMENT;redpoll
.LIST ELEMENT;sparrow
.END LIST
```

Output file (LIST.MEM)

```
1. grosbeak
2. gold finch
3. redpoll
4. sparrow
```

The following sections describe other kinds of lists and the DSR commands you need to create them.

3.2.1 Creating Bulleted Lists

By default, DSR creates a numbered list. If you do not want your list to be numbered, you can substitute bullets or other characters for the numbers. For example, if you want a bulleted list, enter the .LIST command followed by a lowercase o, or bullet, in quotation marks ("o"):

```
.LIST "o"
```

The following example shows how to make a bulleted list:

Input file (LIST.RNO)

```
.LIST "o"
.LIST ELEMENT;ferret
.LIST ELEMENT;mink
.LIST ELEMENT;rabbit
.LIST ELEMENT;sable
.LIST ELEMENT;raccoon
.END LIST
```

Output file (LIST.MEM)

```
o ferret
o mink
o rabbit
o sable
o raccoon
```

3.2.2 Creating Lists Using Any Symbol

If you do not want numbered or bulleted lists, you can choose another character or symbol and enclose it in quotation marks following the .LIST command. Asterisks (*), dollar signs ($), and at signs (@) are used in the following examples:

Input file (LIST.RNO)

```
.LIST "*"
.LIST ELEMENT;jupiter
.LIST ELEMENT;mars
.LIST ELEMENT;venus
.END LIST
```

Output file (LIST.MEM)

```
* jupiter

* mars

* venus
```

Input file (LIST.RNO)

```
.LIST "$"
.LIST ELEMENT;mark
.LIST ELEMENT;yen
.LIST ELEMENT;buck
.LIST ELEMENT;pound
.END LIST
```

Output file (LIST.MEM)

```
$ mark

$ yen

$ buck

$ pound
```

Input file (LIST.RNO)

```
.LIST "@"
.LIST ELEMENT;why
.LIST ELEMENT;not
.LIST ELEMENT;?
.END LIST
```

Output file (LIST.MEM)

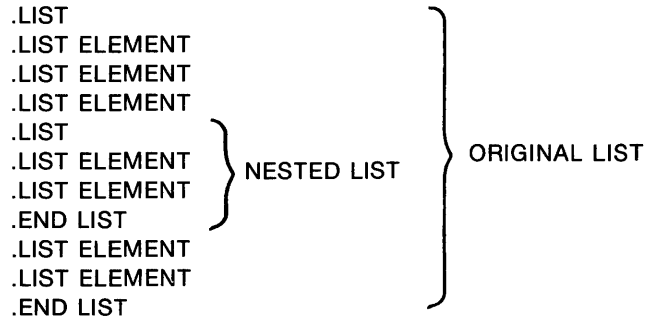
```
@ why

@ not

@ ?
```

3.2.3 Creating Nested Lists

You can also make a nested list—that is, a list indented within a list—by entering another .LIST command between the original .LIST and .END LIST commands. The new .LIST command temporarily suspends the characteristics of the original list. The next .END LIST command ends only the new .LIST command and restores the characteristics of the previous one. You must terminate each list with the .END LIST command.

Figure 3–1 displays the original list containing five elements and the nested list containing three elements.

Figure 3–1 Creating a Nested List



ZK-1264-83

Notice that the nested list in the following example has bullets, unlike the original numbered list:

Input file (LIST.RNO)

```
.LIST
.LIST ELEMENT;German
.LIST ELEMENT;Russian
.LIST ELEMENT;Swedish
.LIST ELEMENT;Yugoslavian
.LIST "o"
.LIST ELEMENT;Serbian
.LIST ELEMENT;Croatian
.LIST ELEMENT;Macedonian
.END LIST
.LIST ELEMENT;Turkish
.LIST ELEMENT;Scottish
.LIST ELEMENT;Irish
.END LIST
```

Output file (LIST.MEM)

1. German
2. Russian
3. Swedish

```
4.  Yugoslavian
      o  Serbian
      o  Croatian
      o  Macedonian
5.  Turkish
6.  Scottish
7.  Irish
```

3.2.4 Creating Lists with Letters and Roman Numerals

By default, DSR numbers lists with decimal numbers. To list with letters or Roman numerals, enter the .DISPLAY ELEMENTS command after the .LIST command. Use the following syntax:

```
.DISPLAY ELEMENTS x
```

The argument *x* is a one- or two-letter code that specifies the form the list numbers will take. The codes and their explanations are shown in the following table:

Code	Form of Sequence and Case
D	Decimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
LU	Uppercase Letters
LL	Lowercase Letters

The following examples show lowercase Roman numerals and letters. Notice that the .DISPLAY ELEMENTS command appears between the .LIST command and the first .LIST ELEMENT command.

Input file (LIST.RNO)

```
.LIST
.DISPLAY ELEMENTS RL
.LIST ELEMENT;tan
.LIST ELEMENT;beige
.LIST ELEMENT;rust
.LIST ELEMENT;brown
.END LIST
```

Output file (LIST.MEM)

```
    i.  tan
   ii.  beige
  iii.  rust
   iv.  brown
```

Input file (LIST.RNO)

```
.LIST
.DISPLAY ELEMENTS LL
.LIST ELEMENT;January
.LIST ELEMENT;February
.LIST ELEMENT;March
.END LIST
```

Output file (LIST.MEM)

```
    a.  January

    b.  February

    c.  March
```

In the following example, notice that each .END LIST command ends the sequence that you specified with .DISPLAY ELEMENTS. You must enter the .DISPLAY ELEMENTS command each time you start a list in which you want to change the numbers or letters. If you want each list to be marked by the same characters, you still must enter a .DISPLAY ELEMENTS command for every .LIST command.

Input file (LIST.RNO)

```
.LIST
.DISPLAY ELEMENTS RL
.LIST ELEMENT;marble
.LIST ELEMENT;maple
.LIST ELEMENT;alabaster
.LIST ELEMENT;oak
.LIST
.DISPLAY ELEMENTS LL
.LIST ELEMENT;light
.LIST ELEMENT;dark
.END LIST
.LIST ELEMENT;glass
.LIST ELEMENT;plastic
.LIST ELEMENT;mahogany
.END LIST
```

Output file (LIST.MEM)

```
     i.  marble

    ii.  maple

   iii.  alabaster

    iv.  oak

           a.  light

           b.  dark

     v.  glass

    vi.  plastic

   vii.  mahogany
```

3.3 Formatting Memos

You can use the following DSR commands to format a memo:

- .BLANK
- .BREAK
- .TAB STOPS

You can organize these three DSR commands into a memo skeleton, like the following one, and fill in the variable information every time you need to send a memo. (Notice that you must press the TAB key to activate the .TAB STOPS command.)

Input file (SKELETON.RNO)

```
.TAB STOPS 30
[TAB]   DATE:
.BREAK
[TAB]   FROM:
.BREAK
[TAB]   DEPT:
.BREAK
[TAB]   EXT:
.BREAK
[TAB]   LOC:
.BLANK 2
TO:
.BLANK
SUBJECT:
.BLANK 2
(Enter text of memo here.)
```

Output file (SKELETON.MEM)

```
                                DATE:
                                FROM:
                                DEPT:
                                EXT:
                                LOC:

TO:

SUBJECT:

(Enter text of memo here.)
```

The .BREAK command ends the current line immediately, without filling or justification. The .TAB STOPS *n* command changes the current position of the tab stops. The value of *n* specifies the character position for a tab stop.

The next memo example uses the following DSR commands:

```
.BLANK
.BREAK
.LEFT MARGIN
.LIST
.LIST ELEMENT
.END LIST
.LITERAL
.END LITERAL
.TAB STOPS
```

Input file (SPY.RNO)

```
.TAB STOPS 40 ❶
[tab]DATE: 13-Nov-88
.BREAK ❷
[tab]FROM: A. Gent
.BREAK
[tab]DEPT: Topspy
.BREAK
[tab]EXT: 007
.BREAK
[tab]LOC: Swiss Branch
.BLANK 2
TO: Underlings
.BLANK 2
SUBJECT: New Equipment

.BLANK 2
The following items will be discussed in our
next meeting:
.LIST ❸
.LIST ELEMENT;super slim tie tack monitor
.LIST ELEMENT;wired socks
.LIST ELEMENT;mini diamond stud earring camera
.END LIST
Please attempt to decode the following message
for your next assignment:
.LITERAL ❹
                que estap dyi
                    fho
                         syto
              szru
                      fsohg
                     wquip
                  dwd
.END LITERAL ❺
```

❶ The .TAB STOPS 40 command sets the first tab stop to character position 40.

❷ The .BREAK command ends the current line immediately and starts a new line without adding a blank line.

❸ The .LIST command begins a list.

❹ The .LITERAL command displays the text literally, exactly as it is entered.

❺ The .END LITERAL command stops displaying the text literally.

Output file (SPY.MEM)

```
                                        DATE: 13-Nov-88
                                        FROM: A.  Gent
                                        DEPT: Topspy
                                        EXT: 007
                                        LOC: Swiss Branch

TO: Underlings

SUBJECT: New Equipment

The following items will be discussed in our next meeting:

     1.  super slim tie tack monitor

     2.  wired socks

     3.  mini diamond stud earring camera
Please attempt to decode the following message for your next
assignment:

               que estap dyi

                   fho
                        syto
             szru
                     fsohg
                    wquip
                dwd
```

3.4 Filling and Justifying Text

By default, DSR automatically fills and justifies text (.FILL and .JUSTIFY commands) when you process a file. The .FILL command causes DSR to fill each line with as many words as possible until the addition of another word would exceed the right margin. The .JUSTIFY command causes DSR to add spaces between words to expand each line exactly to the right margin, making the margin even, or justified. If you do not want DSR to fill or justify the text in your file, you must disable these commands by entering the .NO FILL and .NO JUSTIFY commands.

3.4.1 DSR Commands .FILL and .JUSTIFY (defaults)

The following example demonstrates how DSR fills and justifies text by default:

Input file (VERSE.RNO)

```
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For it so falls out, That what  we  have
we prize not to the worth while we enjoy
it; but being lacked, lacked  and  lost,
Why, then we rack the value.
```

3.4.2 DSR Commands .NO FILL and .JUSTIFY

When you do not want each line of your text to fill with words, enter the .NO FILL command. The .NO FILL command also turns off justification. To obtain an even right margin, include the .JUSTIFY command after the .NO FILL command. Entering the .NO FILL command with the .JUSTIFY command, produces the following variation of the original example. Notice how the .JUSTIFY command spaces the words on each line to create an even right margin.

Input file (VERSE.RNO)

```
.NO FILL
.JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For      it      so      falls      out,
That        what         we         have
we               prize                not
to               the                worth
while        we        enjoy          it;
but            being              lacked,
lacked             and              lost,
Why,
then     we     rack     the      value.
```

3.4.3 DSR Commands .FILL and .NO JUSTIFY

When you want each line of text to fill with words but do not want an even right margin, use the .FILL and .NO JUSTIFY commands. Notice the ragged right margins in the following example:

Input file (VERSE.RNO)

```
.FILL
.NO JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For it so falls out, That what we have
we prize not to the worth while we enjoy
it; but being lacked, lacked and lost,
Why, then we rack the value.
```

3.4.4 DSR Commands .NO FILL and .NO JUSTIFY

When you want your text to stay the way you type it, that is, without filling and justification, enter the .NO FILL and .NO JUSTIFY commands.

Input file (VERSE.RNO)

```
.NO FILL
.NO JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

3.5 Adjusting the Text Display

By default, DSR provides a page length of 58 lines and a page width of 70 characters. When you want to change the size of your page, use the .PAGE SIZE and .RIGHT MARGIN commands. The .PAGE SIZE command specifies the number of lines of text on a page and the page width for the running heads. The syntax for these commands is as follows:

```
.PAGE SIZE n
.RIGHT MARGIN n
```

The parameter *n* with the .PAGE SIZE command indicates the maximum number of lines on a page. It cannot be smaller than 13. The parameter *n* with the .RIGHT MARGIN command indicates the maximum number of characters on a line. It cannot be larger than 150.

The following example shows how to create two different page sizes by entering the .PAGE SIZE command along with the .RIGHT MARGIN command:

Input file (PAGESIZE.RNO)

```
.PAGE SIZE 15
.RIGHT MARGIN 20
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
```

Output file (PAGESIZE.MEM)

```
page size page  size
15   lines  long  20
characters wide page
size  page  size  15
lines    long     20
characters wide page
size  page  size  15
lines     long    20
characters wide page
size  page  size  15
lines    long     20
characters wide page

                Page 2

size  page  size  15
lines     long    20
characters wide page
size  page  size  15
lines    long     20
characters wide
```

Input file (PAGE.RNO)

```
.PAGE SIZE 25
.RIGHT MARGIN 35
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
```

Output file (PAGE.MEM)

```
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide  page  size page size 25 lines
long 35 characters wide  page  size
page   size   25   lines   long  35
characters wide page size page size
25  lines  long  35 characters wide
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide  page  size page size 25 lines
long 35 characters wide  page  size
page   size   25   lines   long  35
characters wide page size page size
25  lines  long  35 characters wide
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide  page  size page size 25 lines
long 35 characters wide  page  size
page   size   25   lines   long  35
```



```
characters wide page size page size
25  lines  long  35 characters wide
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide
```

3.5.1 Indenting Text

By default, when you use the .INDENT command, DSR indents by five spaces the first line of text following the command. The syntax for the indent command is as follows:

```
.INDENT n
```

The parameter *n* specifies the number of character positions to the right of the .LEFT MARGIN setting that the line of text is indented. When you specify a negative number (*-n*), DSR moves the line of text *n* number of character positions to the left of the .LEFT MARGIN setting.

The following example demonstrates how to use the .INDENT command:

Input file (INDENT.RNO)

```
.LEFT MARGIN 10
.RIGHT MARGIN 60
The left margin setting for this text is 10 and the right margin
setting is 60.  If you want to indent text, use the .INDENT command.
.BLANK
.INDENT 20
This line of text is indented 20 spaces.
Every line you type from now on will be flush with the left
margin until you enter the .INDENT command again.
.BLANK
.INDENT 10
This line of text is indented 10 spaces.
If you want a line of text to begin to the left of the
left margin setting, specify a negative number (--n) with
the .INDENT command.
.BLANK
.INDENT -10
This line of text is 10 spaces to the left of the left margin.
This line of text begins at the left margin setting.  Any text
you enter at this point will remain flush with the left
margin until you enter the .INDENT command again.  If you
want to indent two or three lines of text, you must enter
the .INDENT command on each line you want to indent.
.BLANK
.INDENT 15
This is the first of three indented lines.
.INDENT 15
This is the second of three indented lines.
.INDENT 15
This is the third.
This line of text is back at the left margin again.
```

Output file (INDENT.MEM)

```
The left margin setting for this text  is  10  and
the  right  margin  setting is 60.  If you want to
indent text, use the .INDENT command.
                    This line of text is  indented
20  spaces.   Every line you type from now on will
be flush with the left margin until you enter  the
.INDENT command again.
          This line of text is indented 10 spaces.
If you want a line of text to begin to the left of
the left margin setting, specify a negative number
(-n) with the .INDENT command.
```

```
This line of text is 10 spaces to the left of the left
         margin.  This line of text begins at the left
         margin setting.  Any text you enter at this point
         will remain flush with the left margin until you
         enter the .INDENT command again.  If you want to
         indent two or three lines of text, you must enter
         the .INDENT command on each line you want to
         indent.
                        This is the first of three indented
         lines.
                        This is the second of three
         indented lines.
                        This is the third.  This line of
         text is back at the left margin again.
```

3.5.2 Placing a Single Line of Text Relative to the Right Margin

You can use the .RIGHT command to position a single line of text relative to the right margin. The syntax is as follows:

```
.RIGHT n;text
```

The number substituted for variable *n* can be either positive or negative. A positive number specifies how many character positions to the left of the right margin setting the line is indented. A negative number specifies the number of character positions to the right of the right margin setting that the line extends to. *Text* indicates the text to be positioned relative to the right margin. No other DSR command can follow this text on a line.

The following example shows the .RIGHT command being used to notate various directions in the script of a play. (A positive number is specified for *n*.)

Input file (PLAY.RNO)

```
.LEFT MARGIN 0
.RIGHT MARGIN 50
Celia: What did you want me to do?  I really tried to
understand what was going on, but I was scared!
.RIGHT 25;(sob)
.BLANK 1
Bert: Maybe some sensitivity would help?  You never
think of anyone but yourself!  Did you ever stop to
consider that maybe he was just as scared as you were?
No,
.RIGHT 25;(shake head)
all you can think about is yourself.
.BLANK 1
Celia: Maybe I could apologize...
.RIGHT 25;(turn)
make amends...
.BLANK 1
Bert: You had better figure out something because it
is your problem now.
.RIGHT 25;(open door)
```

Output file (PLAY.MEM)

```
Celia:  What did you want  me  to  do?   I  really
tried  to  understand what was going on, but I was
scared!
                        (sob)
Bert:  Maybe some  sensitivity  would  help?   You
never  think of anyone but yourself!  Did you ever
stop to consider that maybe he was just as  scared
as you were?  No,
                (shake head)
all you can think about is yourself.
Celia:  Maybe I could apologize...
                       (turn)
make amends...
Bert:  You had better figure out something because
it is your problem now.
                 (open door)
```

The following example is identical to the previous example, except that a negative number is specified for *n*:

Input file (PLAY.RNO)

```
.LEFT MARGIN 0
.RIGHT MARGIN 50
Celia: What did you want me to do?  I really tried to
understand what was going on, but I was scared!
.RIGHT -5;(sob)
.BLANK 1
Bert: Maybe some sensitivity would help?  You never
think of anyone but yourself!  Did you ever stop to
consider that maybe he was just as scared as you were?
No,
.RIGHT -5;(shake head)
all you can think about is yourself.
.BLANK 1
Celia: Maybe I could apologize...
.RIGHT -5;(turn)
make amends...
.BLANK 1
Bert: You had better figure out something because it
is your problem now.
.RIGHT -5;(open door)
```

Output file (PLAY.MEM)

```
Celia:  What did you want  me  to  do?   I  really
tried  to  understand what was going on, but I was
scared!
                                                (sob)
Bert:  Maybe some  sensitivity  would  help?   You
never  think of anyone but yourself!  Did you ever
stop to consider that maybe he was just as  scared
as you were?  No,
                                          (shake head)
all you can think about is yourself.
Celia:  Maybe I could apologize...
                                                 (turn)
make amends...
Bert:  You had better figure out something because
it is your problem now.
```

3.6 Creating Space on a Page

If you want an example or figure to accompany your text, you need to create some space on the page for it. You can use several different DSR commands to create that necessary space. Some of these commands are as follows:

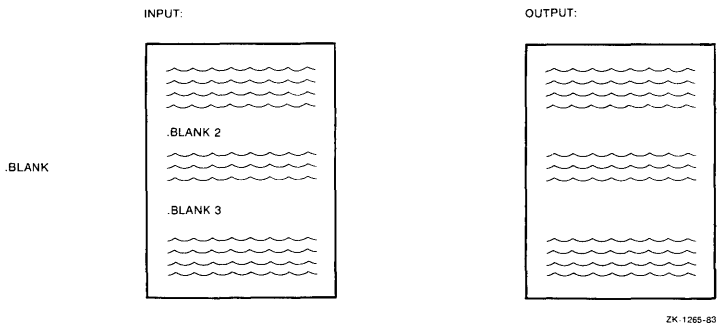
- .BLANK
- .FIGURE
- .FIGURE DEFERRED
- .LITERAL

3.6.1 Separating Sections with Blank Lines

If you use the .BLANK command, DSR leaves the specified number of lines blank. For example, .BLANK 2 produces two blank lines and .BLANK 10 produces ten blank lines. By default, .BLANK produces one blank line. The .BLANK command is useful for creating one or two blank lines between sections of text. (.BLANK does not work correctly when you use it at the top of a page.) If you enter the .BLANK 10 command near the bottom of the page, there may not be enough room on the page for all 10 lines, resulting in your space being split between pages. To avoid possible space splits, use either the .FIGURE command or the .FIGURE DEFERRED command.

Figure 3–2 shows an input file (TEXT.RNO) containing the .BLANK command and the resulting output file (TEXT.MEM) with the blank line.

Figure 3–2 Using the .BLANK Command



3.6.2 Creating Uninterrupted Space

The .FIGURE command leaves space for the specified number of blank lines on the output page at the point in the input file where the command is entered. In the following example, the command .FIGURE 5 adds enough space for 5 lines after the words "entitle them":

Input file DECLARATION.RNO

```
.PAGE SIZE 25
.RIGHT MARGIN 40
When in the Course of human events it becomes necessary
for one people to dissolve the political bands which
have connected them with another, and to assume among
the powers of the earth, the separate and equal station
to which the Laws of Nature and of Nature's God entitle
them,
.FIGURE 5
a decent respect to the opinions of mankind
requires that they should declare the causes which
impel them to the separation.
```

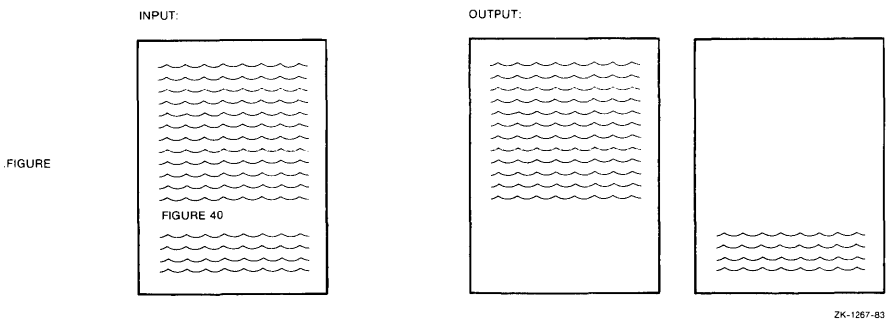
Output file DECLARATION.MEM

```
When in the Course of  human  events  it
becomes  necessary  for  one  people  to
dissolve the political bands which  have
connected  them  with  another,  and  to
assume among the powers  of  the  earth,
the  separate and equal station to which
the Laws of Nature and of  Nature's  God
entitle them,



a decent  respect  to  the  opinions  of
mankind  requires  that  they  should
declare the causes which impel  them  to
the separation.
```

If there is not enough space on the output page for the specified number of blank lines, DSR ends the page immediately and puts the requested number of blank lines at the top of the next page.

Figure 3–3 shows this circumstance with an input file (TEXT.RNO) containing the .FIGURE command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 3–3 Using the .FIGURE Command

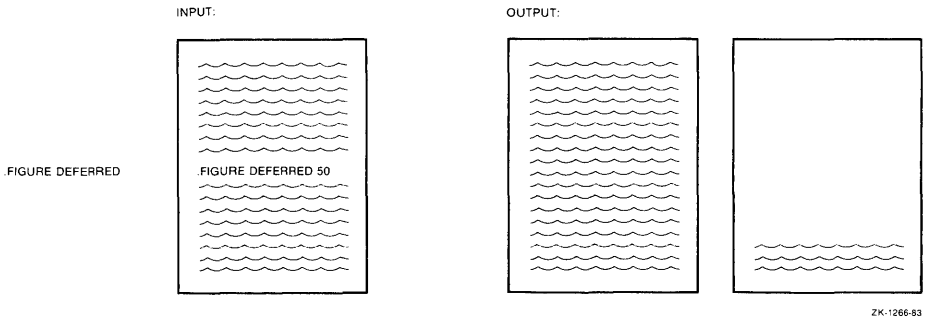


The .FIGURE DEFERRED command leaves enough space on the current page for the specified number of lines. If all the lines do not fit, .FIGURE DEFERRED fills the current page with text and creates space for the specified number of lines at the top of the next page.

For example, if you are about to add a 40-line figure to your file (using the .FIGURE 40 command) but you realize only 25 lines are left on the current output page, add the .FIGURE DEFERRED 40 command to the file instead of the .FIGURE 40 command. This causes DSR to fill the current output page with text and leave a 40-line space at the top of the next page for your figure.

Figure 3–4 shows an input file (TEXT.RNO) containing the .FIGURE DEFERRED command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 3–4 Using the .FIGURE DEFERRED Command

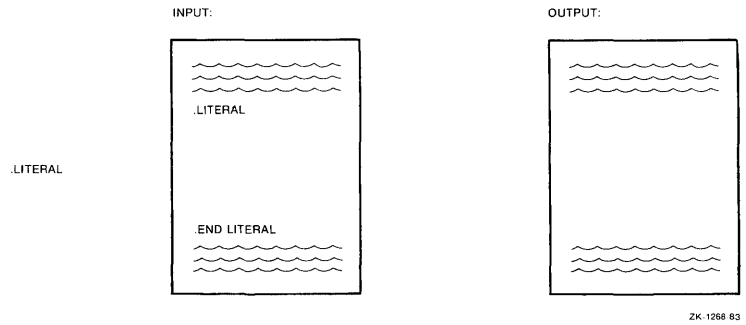


3.6.3 Seeing the Space You Create

You can also use the .LITERAL command to create space. If you want to see the amount of space you are creating, enter the .LITERAL command and press RETURN until the screen has enough blank space. Then, enter the .END LITERAL command. You might want to use the .LITERAL command if you have a picture to paste directly onto your text and want to see how it looks with the blank space you create.

Figure 3–5 shows an input file (TEXT.RNO) containing the .LITERAL command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 3–5 Using the .LITERAL Command



3.6.4 Example of Creating Space

The following example uses the .BLANK, .FIGURE DEFERRED, .FIGURE, and .LITERAL commands in a single file to show the different output that each produces:

Input file (SPACE.RNO)

```
.page size 20
.right margin 40
This is the first line on the page.
Each page will be 20 lines long and 40
characters wide.  If you enter
the .BLANK 2 command here, DSR
creates two blank lines.
.BLANK 2
There is not enough space on this page
for a figure that is 15 lines long, so
if you enter the .FIGURE DEFERRED 15
command here,
.FIGURE DEFERRED 15
DSR will fill up the rest of this page
with text, then create 15 lines of
space at the top of the next page.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .FIGURE 10 command here,
.FIGURE 10
DSR will create 10 lines of space.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
```

```
But, if you enter the .FIGURE 10 command again,
and there is not enough space left on this
page for the entire figure (in this case 10),
DSR stops displaying text immediately
.FIGURE 10
and moves to the top of the next page, creating
the specified space (10 lines).  Then DSR
resumes the display of text.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .LITERAL command and press
RET three times,
.LITERAL


.END LITERAL
you will create three blank lines.  Remember to
enter the .END LITERAL command when you are
done.
```

Output file (SPACE.MEM)

```
This is the first line on the page.
Each page will be 20 lines long and 40
characters wide.  If you enter the
.BLANK 2 command here, DSR creates two
blank lines.

There is not enough space on this page
for a figure that is 15 lines long, so
if you enter the .FIGURE DEFERRED 15
command here,
DSR will fill up the rest of this page
with text, then create 15 lines of space
at the top of the next page.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
                                    Page 2



texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
```

Page 3

texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .FIGURE 10 command here,

DSR will create 10 lines of space.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext

Page 4

texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
But, if you enter the .FIGURE 10 command again, and there is not enough space left on this page for the entire figure (in this case 10), DSR stops displaying text immediately

Page 5

and moves to the top of the next page, creating the specified space (10 lines). Then DSR resumes the display of text.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .LITERAL command and

Page 6

press [RET] three times,

you will create three blank lines. Remember to enter the .END LITERAL command when you are done.

3.7 Formatting Sections

When you want to organize text into sections, use the .HEADER LEVEL command. The syntax for the command is as follows:

```
.HEADER LEVEL n title
```

The value of *n* indicates the specific header level. For example, .HEADER LEVEL 1 creates a section beginning with 1. If you enter a series of .HEADER LEVEL 1 commands, DSR produces section numbers starting with 1 and increasing by 1, as in the following example:

Input file (HEAD1.RNO)

```
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
```

Output file (HEAD1.MEM)

```
1

2

3

4

5
```

The .HEADER LEVEL 2 command affects the second digit. If you enter a series of .HEADER LEVEL 2 commands, the second digit changes, as in the following example:

Input file (HEAD2.RNO)

```
.HEADER LEVEL 2
.HEADER LEVEL 2
.HEADER LEVEL 2
.HEADER LEVEL 2
```

Output file (HEAD2.MEM)

```
0.1

0.2

0.3

0.4
```

The .HEADER LEVEL 3 command affects the third digit. An example of a series of .HEADER LEVEL 3 commands follows:

Input file (HEAD3.RNO)

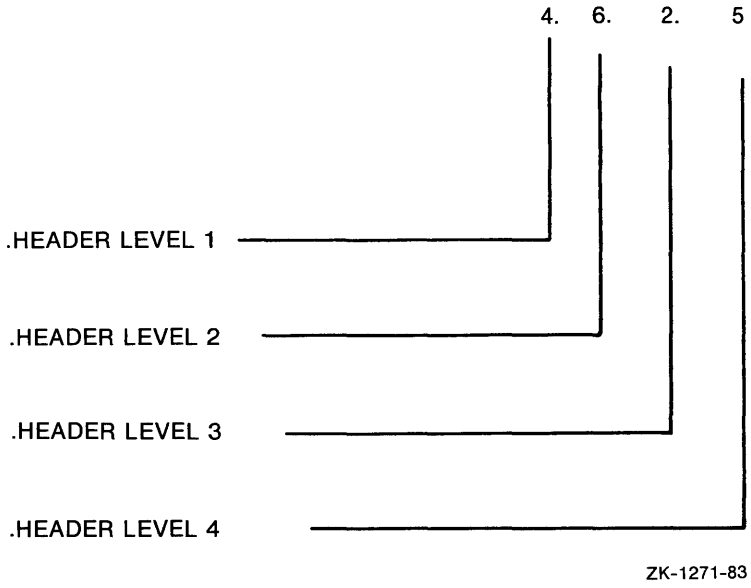
```
.HEADER LEVEL 3
.HEADER LEVEL 3
.HEADER LEVEL 3
.HEADER LEVEL 3
```

Output file (HEAD3.MEM)

```
0.0.1
0.0.2
0.0.3
0.0.4
```

Figure 3–6 shows one section number (4.6.2.5) and the various header levels corresponding to each part of the number.

Figure 3–6 Looking at Header Levels



3.7.1 Specifying a Title

You can follow the .HEADER LEVEL *n* command with a title. DSR uses uppercase for all the letters in a title of header level 1. A title of header level 2 has only initial caps. Header level 3 titles have initial caps followed by a hyphen to separate the title from the text that follows on the same line.

The following example contains a series of .HEADER LEVEL commands:

Input file (LEVEL.RNO)

```
.HEADER LEVEL 1this is a header level 1 title
.HEADER LEVEL 2this is a header level 2 title
.HEADER LEVEL 3this is a header level 3 title
This is the first line of text.
.HEADER LEVEL 1this is the next header level 1 title
```

Output file (LEVEL.MEM)

```
1  THIS IS A HEADER LEVEL 1 TITLE

1.1  This Is A Header Level 2 Title

1.1.1  This Is A Header Level 3 Title - This  is  the  first  line  of
text.


2  THIS IS THE NEXT HEADER LEVEL 1 TITLE
```

The following example demonstrates how to organize recipes using the .HEADER LEVEL command:

Input file (RECIPES.RNO)

```
.HEADER LEVEL 1pies
An introduction to pies goes here.
.HEADER LEVEL 2grasshopper pie
The origin of the Grasshopper Pie goes here.
.HEADER LEVEL3chocolate wafer crust
This section explains how to make the crust.
.HEADER LEVEL3filling
This section explains how to make the filling.
.HEADER LEVEL2lemon meringue pie
Various lemon pies are discussed here.
.HEADER LEVEL3flaky pie crust
The recipe for a good, flaky crust goes here.
.HEADER LEVEL3lemon filling
This section describes how to make the filling.
.HEADER LEVEL3meringue topping
Meringues are mentioned here.
.HEADER LEVEL1layered cakes
Layered cakes are described here.
.HEADER LEVEL2millie's magnificent torte
Tortes are described here.
.HEADER LEVEL3walnut layers
A recipe for the walnut layers goes here.
.HEADER LEVEL3coffee frosting
Coffee frosting is explained here.
```

Output file (RECIPES.MEM)

```
1  PIES

An introduction to pies goes here.


1.1  Grasshopper Pie

The origin of the Grasshopper Pie goes here.


1.1.1  Chocolate Wafer Crust - This section explains how  to
make the crust.


1.1.2  Filling - This  section  explains  how  to  make  the
filling.


1.2  Lemon Meringue Pie

Various lemon pies are discussed here.


1.2.1  Flaky Pie Crust - The recipe for a good, flaky  crust
goes here.
```

```
1.2.2  Lemon Filling - This section describes  how  to  make
the filling.

1.2.3  Meringue Topping - Meringues are mentioned here.

2  LAYERED CAKES

Layered cakes are described here.

2.2  Millie's Magnificent Torte

Tortes are described here.

2.2.1  Walnut Layers - A recipe for the walnut  layers  goes
here.

2.2.2  Coffee Frosting - Coffee frosting is explained here.
```

3.7.2 Using Roman Numerals or Letters

By default, DSR displays decimal numbers when you use the .HEADER LEVEL command. If you want to specify Roman numerals or letters, use the .DISPLAY LEVELS command. The syntax for this command follows:

```
.DISPLAY LEVELS y1,y2,y3...y6
```

The letter *y* is a one- or two-letter code. Several available codes follow:

Code	Form of Sequence and Case
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
LU	Letters, Uppercase
LL	Letters, Lowercase

For more information and a complete list of available codes, see the *VAX DIGITAL Standard Runoff Reference Manual.*

If you want the numbers in the first level heads to be uppercase Roman numerals, the second level to be uppercase letters, and the third level to be lowercase Roman numerals, enter the .DISPLAY LEVELS RU,LU,RL command. The resulting sections look as follows:

```
I  ALL HEADER LEVEL 1 TITLES ARE UPPERCASED

The .HEADER LEVEL 1 command produced this title.  The  code
RU  tells  DSR  to  make  all  level 1 heads uppercase Roman
numerals.

I.A  Header Level 2 Titles Have Initial Uppercase Letters

The .HEADER LEVEL 2 command produced this title.  The  code
LU tells DSR to make all level 2 heads uppercase letters.

I.A.i  Header Level 3 Titles Are Followed By A  Hyphen  -  The
.HEADER  LEVEL  3  command produced this title.  The code RL
tells  DSR  to  make  all  level  3  heads  lowercase  Roman
numerals.
```

3.8 Formatting Chapters

When you want to organize text into chapters, use the .CHAPTER command. The .CHAPTER command specifies the beginning of a chapter, numbers it, and allows you to supply a title. The syntax for the command is as follows:

```
.CHAPTER title
```

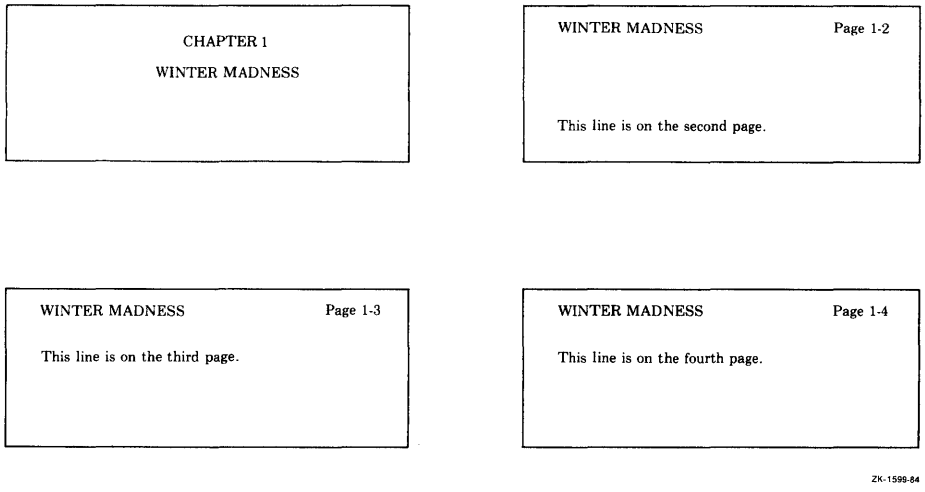
"Winter Madness" is the chapter title in the following example:

Input file (CHAP.RNO)

```
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 20
This line is on the third page.
.BLANK 22
This line is on the fourth page.
```

Figure 3–7 shows the output file (CHAP.MEM).

Figure 3–7 Using the .CHAPTER Command



3.8.1 Numbering Chapters

By default, the .CHAPTER command produces decimal chapter numbers. If you want Roman numerals or letters instead of decimal numbers, enter the .DISPLAY CHAPTER command before the .CHAPTER command you want to affect. The syntax for this command follows:

```
.DISPLAY CHAPTER y
```

The letter *y* is one of the codes discussed in Section 3.7.2.

For more information about all the available codes, see the *VAX DIGITAL Standard Runoff Reference Manual.*

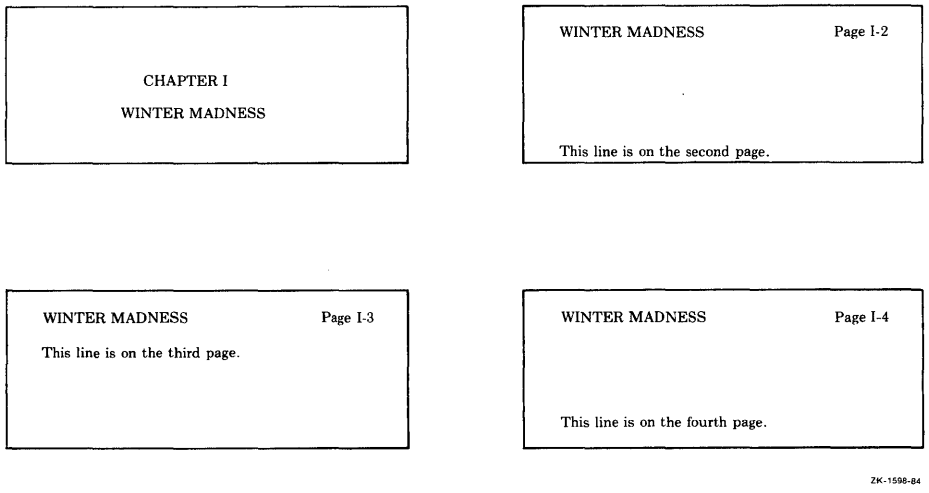
The following example shows how to create chapters numbered with uppercase Roman numerals by entering the .DISPLAY CHAPTER command with the code RU. Notice the uppercase Roman numeral page numbers.

Input file (CHAP.RNO)

```
.DISPLAY CHAPTER RU
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 20
This line is on the third page.
.BLANK 22
This line is on the fourth page.
```

Figure 3–8 shows the output file (CHAP.MEM).

Figure 3–8 Using the .DISPLAY CHAPTER Command



3.8.2 Changing the Way Pages Are Numbered

By default, DSR numbers pages using decimal numbers. If you want your pages lettered or numbered with Roman numerals, use the .DISPLAY NUMBER command before the page you want to affect. The syntax for this command follows:

```
.DISPLAY NUMBER y
```

The letter *y* is one of the codes discussed in Section 3.7.2.

The next example shows the commands you enter to specify chapter and page numbers.

Input file (CHAP.RNO)

```
.DISPLAY CHAPTER RU ❶
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 20
.DISPLAY NUMBER LL ❷
This line is on the third page.
.BLANK 20
.DISPLAY NUMBER RL ❸
This line is on the fourth page.
```

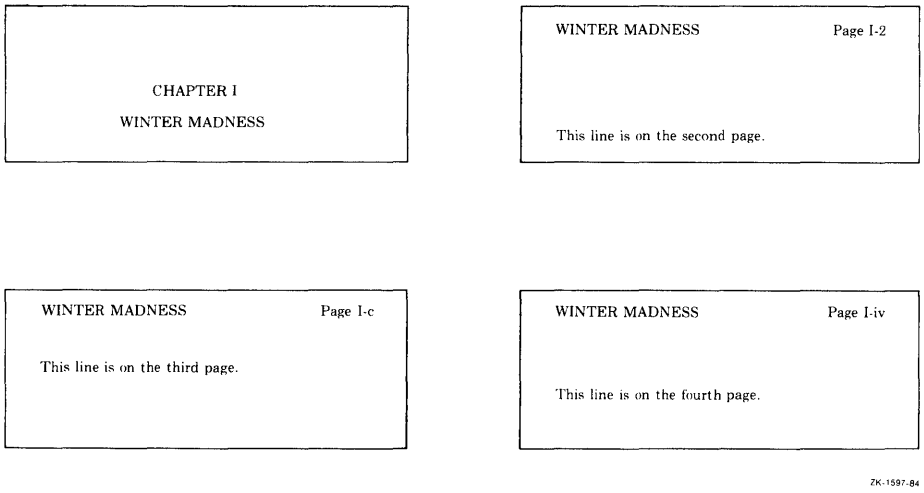
❶ This command numbers chapters with uppercase Roman numerals (RU).

❷ This command changes page numbers from decimal numbers to lowercase letters (LL).

❸ This command changes page numbers from lowercase letters (LL) to lowercase Roman numerals (RL).

Figure 3–9 shows the output file (CHAP.MEM).

Figure 3–9 Using the .DISPLAY NUMBER Command



3.9 Creating an Appendix

You can use the .APPENDIX command to specify the beginning of an appendix. DSR assigns an identifying letter to it and allows you to supply a title. Successive .APPENDIX commands assign identifying letters in alphabetical order.

The syntax for the .APPENDIX command is as follows:

```
.APPENDIX text
```

Text indicates the title you give the appendix.

The following example shows how to create three consecutive appendixes:

Input file (APPENDIX.RNO)

```
This is the last line of text before the appendixes.
.APPENDIX First Title
.APPENDIX Second Title
.APPENDIX Third Title
```

Output file (APPENDIX.MEM)

```
This is the last line of text before the appendixes.
```

APPENDIX A

FIRST TITLE

APPENDIX B

SECOND TITLE

APPENDIX C

THIRD TITLE

3.10 Creating Running Heads

By default, DSR provides page numbers at the top right of every page except the first. The .HEADERS ON command controls DSR's ability to create running page numbers. Running page numbers, which are provided by default, are just part of the information that DSR is able to display about the contents of a page. By using the DSR commands discussed in the following sections, you can create one or two lines of information (running heads) at the top of each page.

If you do not want page numbers, you can disable the page numbering default with the .NO NUMBER command.

The following example shows how DSR formats pages, providing running page numbers by default:

Input file (RUNNING.RNO)

```
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word page at the top
right.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number.
```

Output file (RUNNING.MEM)

```
This line of text is on the first page.  The first page does
not have a page number.
                                                        Page 2

This line of text is on the second page.  From this point
on, every page has a number preceded by the word page at the
top right.
                                                        Page 3

This line of text is on the third page.  Notice the position
of the page number.
```

3.10.1 Specifying a Title

If you want your running head information to contain a title, use the .TITLE command. By default, DSR displays this title at the top left of every page except the first.

The following example shows how DSR formats pages when you enter the .TITLE command with running page numbers and a running title:

Input file (RUNNING.RNO)

```
.TITLE Whispering Willows
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number or a title.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word page at the top
right and a title at the top left.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number and the title.
```

Output file (RUNNING.MEM)

```
This line of text is on the first page.  The first page does
not have a page number or a title.
Whispering Willows                                    Page 2

This line of text is on the second page.  From this point
on, every page has a number preceded by the word page at the
top right and a title at the top left.
Whispering Willows                                    Page 3

This line of text is on the third page.  Notice the position
of the page number and the title.
```

3.10.2 Specifying the Date

When you want the current date to appear in running heads, use the .DATE command. The date appears in the format dd mm yy, for example, 09 December 1988, on the right side of the subtitle line. The .SUBTITLE command must be included for the .DATE command to work.

The following example shows how DSR formats pages when you enter the .DATE command with the .TITLE command:

Input file (RUNNING.RNO)

```
.TITLE Whispering Willows
.SUBTITLE
.DATE
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number, a title, or the date.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word page at the top
right, a title at the top left, and the date.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number, the title, and the date.
```

Output file (RUNNING.MEM)

```
This line of text is on the first page.  The first page does
not have a page number, a title, or the date.

Whispering Willows                                    Page 2
                                            14 December 1986

This line of text is on the second page.  From this point
on, every page has a number preceded by the word page at the
top right, a title at the top left, and the date.

Whispering Willows                                    Page 3
                                            14 December 1986

This line of text is on the third page.  Notice the position
of the page number, the title, and the date.
```

3.10.3 Specifying a Subtitle

To specify a subtitle, enter the .SUBTITLE command after the .TITLE command.

The following example shows how to enter the .TITLE command and the .SUBTITLE command to create running heads with a title and a subtitle:

Input file (RUNNING.RNO)

```
.TITLE Country Gardens
.SUBTITLE Fragrant Garden Lilies
.PAGE SIZE 15
.BLANK 15
The first page does not have any running head information.
.BLANK 15
The title and subtitle appear at the top left of the second page.
The page number is at the top right.
.BLANK 15
Notice the position of the title, subtitle, and page
number on this page.
```

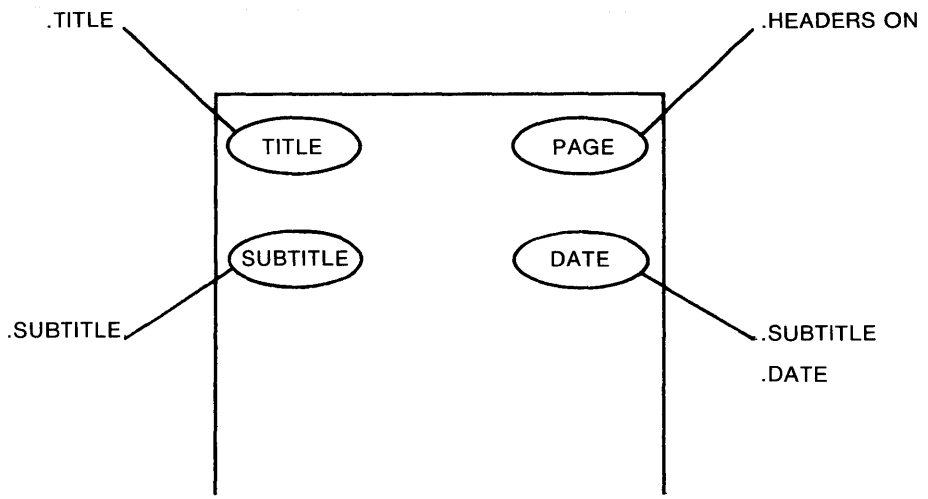
Output file (RUNNING.MEM)

```
The first page does not have any running head information.
Country Gardens                                          Page 2
Fragrant Garden Lilies

The title and subtitle appear at the top left of the  second
page.  The page number is at the top right.
Country Gardens                                          Page 3
Fragrant Garden Lilies

Notice the position of the title, subtitle, and page  number
on this page.
```

3.10.4 Organizing Running Head Information

Figure 3–10 shows the different parts of running head information and the commands you use to generate each part.

Figure 3–10 Running Head Information



You can use any combination of running head commands to alter the organization of information at the top of your documents.

3.10.5 Reorganizing Running Head Information

You can change the position of running head information on a page by using the .LAYOUT command. The .LAYOUT command can be entered before or after the running head information (.TITLE, .SUBTITLE, .DATE, .PAGE commands). You can center titles and subtitles at the tops of pages, center page numbers at the bottom, and surround page numbers with hyphens. All the available options are shown in the following syntax description:

```
.LAYOUT n1,n2
```

The parameter *n1* is a number from 0 to 3 that specifies an alternative arrangement of running head information as follows:

.LAYOUT 0	Restores the standard arrangement of a title and subtitle in the upper left of a page, and a page number and date in the upper right.
.LAYOUT 1	Titles and subtitles are centered at the tops of pages. Page numbers are centered at the bottom. Date does not appear.

.LAYOUT 2	Titles and subtitles appear at the top right of right-hand (odd-numbered) pages and the top left of left-hand (even-numbered) pages. Page numbers are centered at the bottom. Date does not appear.
.LAYOUT 3	Gives the standard page arrangement for title and subtitle (as in .LAYOUT 0) but with the addition of running page numbers centered at the bottom of pages between two hyphens (for example, - 13 -). Running page numbers are consecutive throughout the entire document rather than within chapters.

The parameter *n2* specifies how many lines below the last line of text on a page the number will appear.

Note: When you use .LAYOUT 1, .LAYOUT 2, or .LAYOUT 3, you *must* use the *n2* parameter. When you use .LAYOUT 0, you cannot use the *n2* parameter.

The following example uses the .LAYOUT 1 command, which centers the title and the subtitle at the top of the page and centers the page number at the bottom:

Input file (LAYOUT.RNO)

```
.TITLE Country Gardens
.SUBTITLE Fragrant Garden Lilies
.LAYOUT 1,2
.PAGE SIZE 15
.BLANK 15
This line of text is on page 1.
.BLANK 15
This line of text is on page 2.
.BLANK 12
This line of text is on page 3.
.BLANK 10
This line of text is on page 4.
```

Output file (LAYOUT.MEM)

```
This line of text is on page 1.



                        1
                    Country Gardens
                    Fragrant Garden Lilies

This line of text is on page 2.



                        2
                    Country Gardens
                    Fragrant Garden Lilies
```

```
This line of text is on page 3.


                                3
                          Country Gardens
                          Fragrant Garden Lilies

This line of text is on page 4.


                                4
```

For more information about the .LAYOUT command, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.10.6 Specifying the Title on the First Page

When you want running head information to appear on the first page of a document, enter the .FIRST TITLE command. You must insert the .FIRST TITLE command before any text on the first page.

By default, DSR uses .HEADER LEVEL titles for running head subtitles. So, if you want your own subtitles to override the header level titles, you must use the .NO AUTOSUBTITLE command. For more information about the .AUTOSUBTITLE and .NO AUTOSUBTITLE commands, see the *VAX DIGITAL Standard Runoff Reference Manual.*

The following example shows how to create running head information on the first page by using the .FIRST TITLE command. The .NO AUTOSUBTITLE command causes the specified subtitle (Fragrant Garden Lilies) to override the header level title ('Troubadour').

Input file (RUNNING.RNO)

```
.FIRST TITLE
.TITLE Country Gardens
.NO AUTOSUBTITLE
.SUBTITLE Fragrant Garden Lilies
.DATE
.HEADER LEVEL1'Troubadour'
"A magnificent trumpet Lily of most unusual coloring.  The inside is purest
white, the outside is a rich maroon with a white margin.  Very hardy.  It
produces 6-foot stems that bear up to a dozen or more huge 6-inch flowers.
July blooming."
.HEADER LEVEL2Planting
"Lilies are sent in late October and November.  In areas where the ground
freezes early, we recommend that you dig the holes for them early, covering
with mulch, so that your Lilies can easily be planted when they arrive."
.HEADER LEVEL3Price
90887-1   Each $3.75, Three $9.50, Six $18.00, Doz. $34.00
```

Output file (RUNNING.MEM)

```
Country Gardens                                              Page 1
Fragrant Garden Lilies                                 16 October 1988

1  'TROUBADOUR'

"A magnificent trumpet Lily of most unusual coloring.  The  inside  is
purest  white, the outside is a rich maroon with a white margin.  Very
hardy.  It produces 6-foot stems that bear up to a dozen or more  huge
6-inch flowers.  July blooming."

1.1  Planting

"Lilies are sent in late October and November.  In  areas  where  the
ground  freezes  early,  we  recommend that you dig the holes for them
early, covering with mulch, so that your Lilies can  easily be planted
when they arrive."

1.1.1  Price - 90887-1 Each $3.75,  Three  $9.50,  Six  $18.00,  Doz.
$34.00
```

3.11 Creating Notes and Footnotes

You can use DSR to format notes and footnotes. This section discusses the DSR commands you need to create notes and footnotes.

3.11.1 Using the .NOTE Command

The .NOTE command causes DSR to make the margins narrower, center a title over the text, and leave two blank lines before and one blank line after the title. The .END NOTE command causes DSR to leave a blank line after the note and restore the margin settings that were in effect before you entered .NOTE.

Use the following syntax for this command:

```
.NOTE note_title
text of note
.ENDNOTE
```

Specify a title for the note as a parameter to the .NOTE command. If you do not specify a title for the note, DSR provides the word NOTE.

The following example demonstrates how to create a note using the .NOTE command:

Input file (NOTE.RNO)

```
When you are entering text and you want to set off some
information, you can use the .NOTE and .END NOTE commands.
Notice the title and change in margins that DSR provides.
.NOTE Note Fun
This text is part of the note.  The margins are much narrower.
There are two blank lines before and one after the note.  Also,
the title is centered over the note.
.END NOTE
When you enter the .END NOTE command, your original margins
return.
```

Output file (NOTE.MEM)

```
When you are entering text and you want  to  set  off  some
information,  you  can use the .NOTE and .END NOTE commands.
Notice the title and change in margins that DSR provides.

                              Note Fun

                    This text is part of the note.
                    The margins are much narrower.
                    There  are  two  blank   lines
                    before and one after the note.
                    Also, the  title  is  centered
                    over the note.

When you enter the .END NOTE command, your original  margins
return.
```

3.11.2 Using the .FOOTNOTE Command

Use the .FOOTNOTE and .END FOOTNOTE commands to create footnotes. The .FOOTNOTE command places the text following it at the bottom of the current page if there is room. If there is not enough room for the entire footnote, DSR places it at the bottom of the next page.

The .END FOOTNOTE command ends the footnote and restores any case, fill, justify, spacing, or margin settings that you might have changed in the footnote text.

The following example demonstrates how to create footnotes:

Input file (FOOT.RNO)

```
.PAGE SIZE 35
.RIGHT MARGIN 55
When you want to add a footnote to your text, use the .FOOTNOTE
and .END FOOTNOTE commands.  DSR puts the footnote at the bottom
of the page.  If there is not enough room on the current page,
DSR puts the footnote on the next page.
.BLANK
For this example, the page size is set to 35 lines long and the
page width is set to 55 characters.
.FOOTNOTE
This text is part of the footnote.  Notice where it appears on
the page.
.END FOOTNOTE
```

Output file (FOOT.MEM)

```
When you want to add a footnote to your text,  use  the
.FOOTNOTE  and  .END  FOOTNOTE  commands.  DSR puts the
footnote at the bottom of the page.  If  there  is  not
enough  room on the current page, DSR puts the footnote
on the next page.

For this example, the page size is set to 35 lines long
and the page width is set to 55 characters.




This text is part of the  footnote.   Notice  where  it
appears on the page.
```

The .FOOTNOTE command does not provide a footnote symbol such as an asterisk (*) or (1). You can add one of these symbols before the text of the footnote by adding the .LEFT MARGIN command and the SPACE flag (#) as the following syntax shows:

Using an asterisk

```
.FOOTNOTE
.LEFT MARGIN -2;*#text
.END FOOTNOTE
```

Using a number

```
.FOOTNOTE
.LEFT MARGIN -2;(1)#text
.END FOOTNOTE
```

Text is the text of the footnote.

The following example shows how DSR moves the footnote to the next page and how you can precede the footnote by an asterisk:

Input file (FOOT.RNO)

```
.PAGE SIZE 15
.RIGHT MARGIN 50
This is the first line of text on this page.
Each page will have 15 lines.  If the footnote does not fit
on the page on which it occurs, DSR places it on the bottom
of the next page. But, DSR will not split a single footnote
over two pages.
.BLANK 2
The .FOOTNOTE command does not provide a footnote symbol such
as an asterisk (*) or (1).  But, if you want to put an asterisk
before the text of the footnote, add the .LEFT MARGIN command
and the SPACE flag.
.FOOTNOTE
.LEFT MARGIN 2
.INDENT -2;*#
This is the first line of the footnote.  Because there is not
enough room on the first page, DSR will put this footnote on
the second page.
.END FOOTNOTE
.BLANK 3
Here is more text following the footnote insert.
```

Output file (FOOT.MEM)

```
This is the first line of text on this page.  Each
page will have 15 lines.  If the footnote does not
fit on the page on which it occurs, DSR places  it
on the bottom of the next page.  But, DSR will not
split a single footnote over two pages.

The .FOOTNOTE command does not provide a  footnote
symbol  such  as  an asterisk (*) or (1).  But, if
you want to put an asterisk before the text of the
footnote,  add  the  .LEFT  MARGIN command and the
SPACE flag.
```

```
                                                              Page 2

Here is more text following the footnote insert.



*   This  is  the  first  line  of  the  footnote.
  Because  there  is  not enough room on the first
  page, DSR will put this footnote on  the  second
  page.
```

For more information about the .NOTE and .FOOTNOTE commands, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.12 Emphasizing Text

DSR allows you to emphasize text by either boldfacing or underlining. To boldface or underline, you insert a special character called a flag into your text where you want the boldfacing or underlining to occur. (To boldface, you need to enter the .FLAGS BOLD command before the flag, which causes DSR to recognize the bold flag.) The Bold and Underline flags are as follows:

```
Bold Flag           (*)

Underline Flag      (&)
```

When you precede a character by the Bold flag (*), the character is boldfaced, that is, overstruck once. You can cause the characters to be overstruck more than once by using the /BOLD qualifier in the DSR command line. (See the *VAX DIGITAL Standard Runoff Reference Manual* for information about the /BOLD qualifier.)

The following example shows how to boldface individual characters:

Input file (BOLD.RNO)

```
.FLAGS BOLD
Follow route *3 to route *7.
```

Output file (BOLD.MEM)

```
Follow route 3 to route 7.
```

You can pair the Bold flag with the Uppercase flag (^*) to turn boldfacing on and pair it with the Lowercase flag (*) to turn boldfacing off. The following example demonstrates how to boldface an entire line of text:

Input file (BOLD.RNO)

```
.FLAGS BOLD
^*This entire line of text is in boldface.\*
```

Output file (BOLD.MEM)

```
This entire line of text is in boldface.
```

When you precede a character by the Underline flag (&), DSR underlines the character. The following example shows how to underline individual characters:

Input file (UNDERLINE.RNO)

```
&A is for Amy and &B is for Basil...
```

Output file (UNDERLINE.MEM)

```
A is for Amy and B is for Basil...
```

You can pair the Underline flag with the Uppercase flag (^&) to turn underlining on and pair it with the Lowercase flag (\&) to turn underlining off. The following example demonstrates how to underline an entire line of text:

Input file (UNDERLINE.RNO)

```
^&KEEP OFF THE GRASS, PLEASE\&
```

Output file (UNDERLINE.MEM)

```
KEEP OFF THE GRASS, PLEASE
```

3.13 Creating a Table of Contents and an Index

You can use DSR to create a table of contents or an index. The table of contents you produce can display chapter titles and numbers, header levels, and appendix titles and letters. The index you produce can display a two-column index with alphabetized entries at the left of each column. Each entry is separated from its page numbers by a comma. Entries with different first letters are separated by a blank line.

Figure 3–11 displays the three steps to follow to create either a table of contents or an index. Step 1 is the same whether you are creating a table of contents or an index. It produces a single intermediate (binary) file with a file type of BRN. This BRN file contains both table of contents and indexing information.

3.13.1 Creating a Table of Contents

To create a table of contents, follow three steps.

1. Enter the following command line to generate an intermediate (binary) file:

   ```
   $ RUNOFF/INTERMEDIATE file.RNO
   ```

   Be sure to specify an RNO file. DSR produces a BRN file.

2. Enter the following command line to run the Table of Contents Utility:

   ```
   $ RUNOFF/CONTENTS file.BRN
   ```

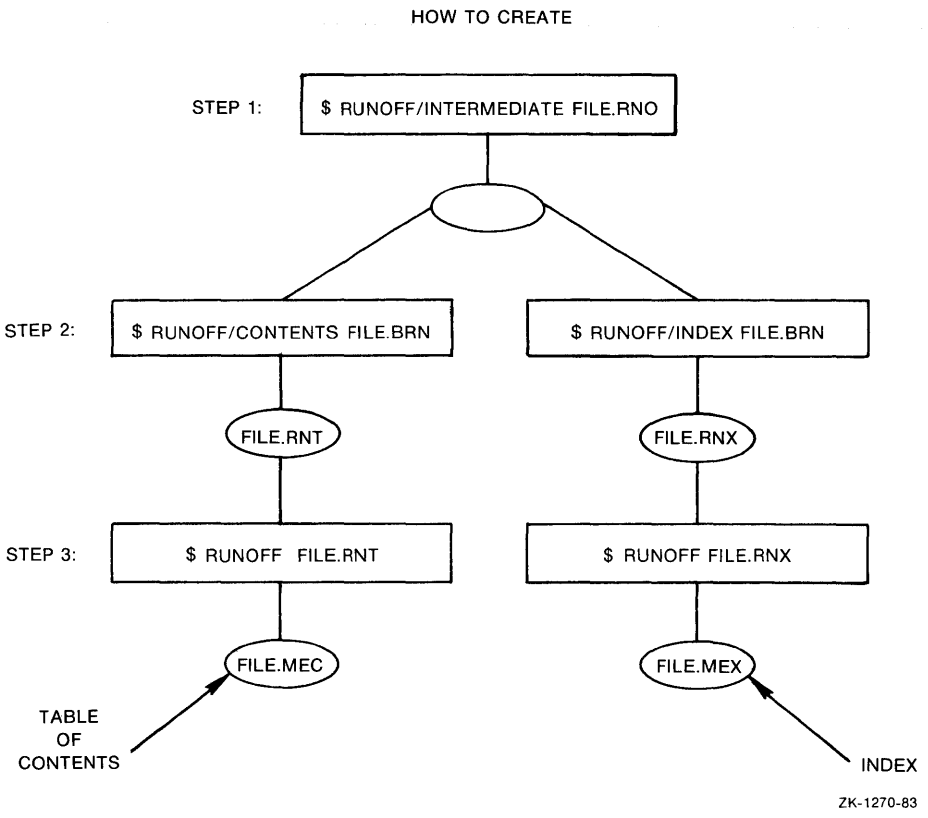
   Be sure to specify a BRN file. You can add qualifiers to this command line to customize the Table of Contents Utility. DSR produces an RNT file.

3. Enter the following command line to process the RNT file:

   ```
   $ RUNOFF FILE.RNT
   ```

   Be sure to specify an RNT file. DSR produces a MEC file. This MEC file contains the table of contents.

Figure 3–11 Creating a Table of Contents or an Index



The following table shows the command lines to enter for each step and the resulting file types:

Command Line	Resulting File Types
RUNOFF/INTERMEDIATE FILE.RNO	.BRN
RUNOFF/CONTENTS FILE.BRN	.RNT
RUNOFF FILE.RNT	.MEC

When you enter the RUNOFF/CONTENTS command without qualifiers, you get the following defaults:

- Chapter titles and numbers (generated by the .CHAPTER command).
- Section titles and numbers (generated by the .HEADER LEVEL command). By default, DSR allows up to six levels of headers to appear in the table of contents.
- Appendix titles and letters (generated by the .APPENDIX command).
- Chapter-oriented page numbers (1–1, 1–2, 1–3, . . .) for all table of contents entries.
- An output file with the same name as the input file.

3.13.1.1 Tailoring the Table of Contents Utility

To tailor the DSR Table of Contents Utility to your own needs, use the qualifiers listed in the following table:

Qualifier	Results
/BOLD	Any bolding specified in chapter and header titles appears in the table of contents.
/DEEPEST_HEADER=n	The number you specify for *n* determines the deepest header level displayed in the table of contents.
/OUTPUT=newfile	DSR produces an output file with the name specified by =newfile. This output file, like the default output file, has a file type of RNT.
/LOG	Processing information is displayed.
/IDENTIFICATION	The version of DSR used to process your file is displayed.
/NOOUTPUT	An output file is not produced.
/PAGE_NUMBERS=RUNNING	Running page numbers appear instead of chapter-oriented page numbers for all table of contents entries, whether or not you specified running page numbers in the document.
/NOSECTION_NUMBERS	Header level numbers do not appear in the table of contents.
/UNDERLINE	Any underlining specified in chapter and header titles appears in the table of contents.

3.13.1.2 Looking at Tables of Contents

The following examples demonstrate how two of the qualifiers already described can alter the appearance of a table of contents. For more information about these and other qualifiers, see the *VAX DIGITAL Standard Runoff Reference Manual.*

The first example shows a table of contents produced by DSR defaults. No qualifiers are added to the command line.

```
$ RUNOFF/CONTENTS

                                CONTENTS


CHAPTER 1       HOW TO TILE A FLOOR

        1.1     READING ABOUT TILING . . . . . . . . . . . 1-1
        1.1.1     Tiling For Fun . . . . . . . . . . . . . 1-2
        1.1.2     Your Home In Tile . . . . . . . . . . . 1-3
        1.1.3     Changing A Room With Tile . . . . . . . 1-3
        1.1.4     How To Tile Floors And Walls . . . . . 1-4
```

The second example changes the display of page numbers from chapter-oriented numbers (1–1, 1–2, 1–3, . . .) to running numbers (1,2,3, . . .) by using the following command line:

```
$ RUNOFF/CONTENTS/PAGE_NUMBERS=RUNNING
```

CONTENTS

```
CHAPTER 2       HOW TO CEDAR A CEILING

        2.1     GETTING STARTED . . . . . 12
        2.1.1     Various Surfaces . . . . . 12
```

The third example displays a table of contents without section numbers by using the following command line:

```
$ RUNOFF/CONTENTS/NOSECTION_NUMBERS

                                CONTENTS


CHAPTER 1       HOW TO TILE A FLOOR

                READING ABOUT TILING . . . . . 1-1
                  Tiling For Fun . . . . . 1-2
                  Your Home In Tile . . . . . 1-3
                  Changing A Room With Tile . . . . . 1-3
                  How To Tile Floors And Walls . . . . . 1-4
                BUYING THE TILE . . . . . 1-5
                  Researching Tiles Produced Abroad . . 1-5
                  Coordinating Colors . . . . . 1-6
                  Colors That Fade . . . . . 1-6
                  Tile Textures . . . . . 1-6
                  Types Of Tiles . . . . . 1-7
                  Ceramic . . . . . 1-7
                  Clay . . . . . 1-7
                  Cement . . . . . 1-7
                TOOLS FOR TILES . . . . . 1-8
                  Renting A Cutter . . . . . 1-8
                  Buying Or Renting Crimpers . . . . . 1-9
                ACCOMPANYING MATERIALS . . . . . 1-9
                  How To Adhere The Tiles . . . . . 1-9
                  Mastic . . . . . 1-9
                  Types Of Mastic . . . . . 1-10
                  Grout . . . . . 1-10
                  Coordinating Colors . . . . . 1-10
                  How To Mix . . . . . 1-11
                  How To Apply . . . . . 1-11

CHAPTER 2       HOW TO CEDAR A CEILING

                GETTING STARTED . . . . . 2-1
                  Various Surfaces . . . . . 2-1
```

3.13.1.3 Comparing New DSR with Previous Versions of DSR

Previous versions of DSR also required three steps to create a table of contents. These three steps and the corresponding new steps are shown in the following list:

1	OLD	RUNOFF/CONTENTS FILE.RNO
	NEW	RUNOFF/INTERMEDIATE FILE.RNO
2	OLD	RUN SYS$SYSTEM:TOC (.BTC)
	NEW	RUNOFF/CONTENTS FILE.BRN
3	OLD	RUNOFF FILE.RNT
	NEW	RUNOFF FILE.RNT

As shown in the list, previous versions of DSR generated BTC files instead of BRN files. Although these BTC files are no longer produced, the new DSR contents utility processes any BTC files you still have and want to process.

3.13.2 Creating an Index

To create an index, you enter .INDEX and .ENTRY commands throughout your file. The syntax for an index entry is as follows:

```
.INDEX topic> subtopic> subtopic
```

or

```
.ENTRY topic
```

For example, if you want the words "clown fish" to appear in your index, enter the .INDEX command followed by the words "clown fish":

```
The tank was teaming with tetras, but Marvin was interested in
.INDEX clown fish
the clown fish at the front of the store.
```

For more information about the index commands, see the *VAX DIGITAL Standard Runoff Reference Manual.*

After you enter the index commands in your file, you are ready to run the Indexing Utility. To run the Indexing Utility, follow three steps.

1. Enter the following command line to generate an intermediate (binary) file:

   ```
   $ RUNOFF/INTERMEDIATE FILE.RNO
   ```

   Be sure to specify a RNO file. DSR produces a BRN file.

2. Enter the following command line to run the Indexing Utility:

   ```
   $ RUNOFF/INDEX FILE.BRN
   ```

   Be sure to specify a BRN file. You can add qualifiers to this command line to customize the Indexing Utility. DSR produces a .RNX file.

3. Enter the following command line to process the .RNX file:

   ```
   $ RUNOFF FILE.RNX
   ```

   Be sure to specify an RNX file. DSR produces a MEX file. This MEX file contains the index.

The following table shows the command lines you enter for each step and the resulting file types:

Command Line	Resulting File Types
RUNOFF/INTERMEDIATE FILE.RNO	.BRN
RUNOFF/INDEX FILE.BRN	.RNX
RUNOFF FILE.RNX	.MEX

3.13.2.1 Tailoring the Index Utility

When you use the RUNOFF/INDEX command without qualifiers, you get the following defaults:

- 55 lines per page, including the top and bottom header areas
- Chapter-oriented page numbers for index entries

- Consecutive page numbers merged into ranges
- An output file with the same name as the input file

You can use the qualifiers listed in the following table to tailor the indexing utility:

Qualifier	Results
/IDENTIFICATION	The version of DSR used to process your file is displayed.
/LINES_PER_PAGE=n	The number you specify for *n* determines the size of the page text, including the top and bottom header areas.
/LOG	Processing information is displayed.
/OUTPUT=newfile	DSR produces an output file with the name specified by =newfile. This output file, like the default output file, has a file type of .RNX.
/NOOUTPUT	An output file is not produced.
/NOPAGE_NUMBERS	Index entries will not have page numbers.
/PAGE_NUMBERS=RUNNING	Running page numbers (1,2,3, . . .) appear instead of chapter-oriented page numbers (1–1, 1–2, 1–3, . . .) for all index entries, whether or not you specified running page numbers in the document.
/REQUIRE=filename	Allows you to change the heading on the first page of an index.
/RESERVE=n	DSR reserves *n* number of lines on the top of the index page.

For more information about all the qualifiers available for creating an index, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.13.2.2 Looking at Indexes

The following examples demonstrate how two of the qualifiers already described can alter the appearance of an index. For more information about these and other qualifiers, see the *VAX DIGITAL Standard Runoff Reference Manual.*

The first example shows an index produced by DSR defaults. No qualifiers are added to the command line.

```
$ RUNOFF/INDEX

                                                        Page Index-1

                                INDEX


Amadeus                                 Liszt, franz, 3-2, 4-11
  see mozart
                                        Mozart, wolfgang amadeus, 3-5,
Bach, carl phillip emanuel, 1-2             4-14
    to 1-3, 4-9
Bach, johann sebastian, 1-1, 3-2,       Prokofiev, sergei, 4-5, 4-15
    4-9, 4-12
Baroque composer                        Rachmaninoff, sergei, 3-3 to 3-4,
  see bach                                  4-13
Bartok, bela, 2-1, 3-4, 4-10,           Rite of spring
    4-13                                  see stravinsky
Britten, benjamin, 4-3, 4-14
                                        Satie, erik, 2-2, 4-10
Ceremony of carols                      Stravinsky, igor, 4-7, 4-15
  see britten                           Syrinx
Chopin, frederic, 4-3 to 4-4,             debussy, 4-8, 4-17
    4-14
Debussy, claude, 3-3, 4-13              Velvet gentleman
                                          see satie
French composer
  see debussy

                                        Waltz
Hindemith, paul, 4-6 to 4-7, 4-15         see chopin
```

In the second example, DSR displays index pages 15 lines long instead of 55 lines long (the default) by using the following command line:

```
$ RUNOFF/INDEX/LINES_PER_PAGE=15

                                                        Page Index-1

                                INDEX


Amadeus                                 Britten, benjamin, 4-3, 4-14
  see mozart
                                        Ceremony of carols
Bach, carl phillip emanuel, 1-2           see britten
    to 1-3, 4-9                         Chopin, frederic, 4-3 to 4-4,
Bach, johann sebastian, 1-1, 3-2,           4-14
    4-9, 4-12
Baroque composer                        Debussy, claude, 3-3, 4-13
  see bach
Bartok, bela, 2-1, 3-4, 4-10,           French composer
    4-13                                  see debussy
```

```
                                                          Page Index-2

Hindemith, paul, 4-6 to 4-7, 4-15
Liszt, franz, 3-2, 4-11

Mozart, wolfgang amadeus, 3-5,
    4-14
Prokofiev, sergei, 4-5, 4-15

Rachmaninoff, sergei, 3-3 to 3-4,
    4-13
Rite of spring
  see stravinsky
Satie, erik, 2-2, 4-10
Stravinsky, igor, 4-7, 4-15
Syrinx
  debussy, 4-8, 4-17
Velvet gentleman
  see satie

Waltz
  see chopin
```

The third example shows how to create an index with running page numbers (1,2,3, . . .), instead of chapter-oriented page numbers (1–1, 1–2, 1–3, . . .), by using the following command line:

```
$ RUNOFF/INDEX/PAGE_NUMBERS=RUNNING
```

```
                                                          Page Index-1

                                INDEX


Amadeus
  see mozart

Bach, carl phillip emanuel, 2 to
    3, 20
Bach, johann sebastian, 1, 8, 20,
    23
Baroque composer
  see bach
Bartok, bela, 4, 10, 21, 24
Britten, benjamin, 14, 25

Ceremony of carols
  see britten
Chopin, frederic, 14 to 15, 25
Debussy, claude, 9, 24

French composer
  see debussy

Hindemith, paul, 17 to 18, 26
Liszt, franz, 8, 22

Mozart, wolfgang amadeus, 11, 25

Prokofiev, sergei, 16, 26

Rachmaninoff, sergei, 9 to 10, 24
Rite of spring
  see stravinsky

Satie, erik, 5, 21
Stravinsky, igor, 18, 26
Syrinx
  debussy, 19, 28

Velvet gentleman
  see satie


Waltz
  see chopin
```

3.13.2.3 Comparing New DSR with Previous Versions of DSR

Previous versions of DSR also required three steps to create an index. These three steps are shown in the following list, along with the corresponding new steps:

1	OLD	RUNOFF/INDEX FILE.RNO
	NEW	RUNOFF/INTERMEDIATE FILE.RNO
2	OLD	RUN SYS$SYSTEM:TCX (.BIX)
	NEW	RUNOFF/INDEX FILE.BRN
3	OLD	RUNOFF FILE.RNX
	NEW	RUNOFF FILE.RNX

As shown in the list, previous versions of DSR generated BIX files instead of BRN files. Although these BIX files are no longer produced, the new indexing program processes any BIX files that you still have and want to process.

A Customizing EVE

This appendix describes how to use startup files to modify the standard EVE editor. Startup files hold key definitions and editing commands that set the characteristics of the editing environment. Startup files can also hold VAXTPU procedures, which augment the editing capability of the standard EVE editor.

By placing your definitions and procedures in a startup file, you can invoke the editor and automatically establish the editing environment your task requires. The following sections describe how to create and use different types of startup files.

A.1 Creating a Customized Editor

EVE provides the following ways of tailoring the standard editor to meet your editing requirements:

- Defining editing keys. You can assign an editing command to a key so that commands are faster to enter. For example, you can bind the EVE command ERASE WORD to the key sequence CTRL/D.
- Creating *learn sequences*. You can assign a series of commands or keystrokes to one key. For example, you can create a learn sequence where you can press one key to insert a new phone number into a standard memo heading.
- Writing editing procedures using the VAX Text Processing Utility (VAXTPU). Because EVE is built on a powerful, programmable utility called the VAX Text Processing Utility, you can expand the editor beyond its standard set of commands by using VAXTPU procedures. With VAXTPU language statements, you can write procedures that create features that are not available in standard EVE. For example, you can write a procedure to transpose two characters and assign this procedure to a key.

Section 1.9 and Section 1.12 in the EVE chapter describe how to define keys for command and learn sequences and how to write VAXTPU procedures.

A.2 Using Startup Files

You can save key definitions, learn sequences, or VAXTPU procedures in a *startup file*. A startup file allows you to save all the modifications you have made of standard EVE so you do not have to recreate your modifications at each editing session.

EVE has three types of startup files:

- Section files
- Initialization files
- Command files

A section file contains key definitions, learn sequences, and compiled VAXTPU statements and procedures in binary form. Because section files are in binary form, they set up the editing environment very quickly, but you cannot display or edit a binary file. Use a section file to implement editing features that are not likely to change from one editing session to another. For example, define keys that you use regularly in a section file.

An initialization file is an ASCII file containing standard EVE commands. You can easily display an initialization file and edit it. You can execute an initialization file when invoking EVE or during an editing session, with the @ command. Initialization files set up the editing environment more slowly than section files or command files. They contain key definitions but not learn sequences.

A command file is a VAXTPU source file containing VAXTPU statements and procedures. You can use a command file in two different ways. A command file can be used to generate an EVE section file. Or, it can be used as a command file that EVE executes to create an editing environment. For example, you can use a command file to control the appearance of the buffer and the startup mode of the editor. Command files execute more slowly than section files, but since they are ASCII files, you can display and edit them.

Command files and initialization files can be used for many of the same tasks. However, command files execute more quickly and offer more sophisticated editing tools.

A.3 Creating Section Files

EVE requires a section file for startup. By default, EVE uses the section file EVE$SECTION.TPU$SECTION located in directory SYS$SHARE. This default section file defines the editing keys, shown in Figure 1–1 and Figure 1–2, as well as the standard EVE commands, discussed in Chapter 1.

Rather than use the default section file, you can create a modified section file that contains the standard EVE functions as well as your own key definitions, learn sequences, and editing functions. You can create a section file in two ways:

- If you have a small number of key definitions and learn sequences, define editing keys and assign learn sequences interactively. (See Section 1.12.2 for details.) To save the definitions in a section file, enter the SAVE EXTENDED EVE command.

  The default extension for section files is TPU$SECTION. Each time you enter the SAVE EXTENDED EVE command, you can specify the same file to ensure that all your customizations are saved cumulatively in the same file.

- If you have a large number of modifications, create a command file with key definitions and VAXTPU procedures. (Note, however, that a command file cannot have learn sequences.) End the file with the SAVE and QUIT statements, assigning a section file name with the file type TPU$SECTION, as shown in Example A–1. Then invoke EVE with the /COMMAND qualifier and the command file name. EVE executes statements in the command file, saves the compiled procedures and key definitions in the section file named in the SAVE statement, and then

executes the QUIT statement, returning control to the DCL. You can now use the new section file, as named in the SAVE statement.

To use a section file, specify the section file name with the /SECTION qualifier on the EVE command line. For example, the following command invokes EVE with a section file named MY_SECTION.TPU$SECTION, located in directory ALEXIS on a disk called WORK1:

```
$ EDIT/TPU/SECTION=WORK:[ALEXIS]MY_SECTION
```

By default, VAXTPU uses the section file whose logical name is TPU$SECTION. If you define this logical name in your LOGIN.COM file, VAXTPU automatically uses your section file when you invoke EVE. For example,

```
$ DEFINE TPU$SECTION WORK1:[ALEXIS]MY_SECTION.TPU$SECTION
```

Use the EVE command SHOW SUMMARY to display the name of the current section file.

EVE executes a section file before a command file or an initialization file. Therefore, definitions in the command file and initialization file override section file definitions. When you want to set the characteristics of the editing environment, use either a command file or an initialization file. EVE executes these commands upon startup, so the appearance of the buffer and the editing mode is adjusted according to your definition.

A.4 Creating Initialization Files

Rather than defining keys or setting the characteristics of an editing session interactively, you can put EVE commands and key definitions in an initialization file. You can execute an initialization file when invoking EVE or during an editing session, by using the @ command; for example,

```
Command: @SETUP_INIT
```

Each command in an initialization file begins on a separate line. You can add comments to the file to document it, as long as you precede the comments with an exclamation mark and place them on a line separate from a command. An initialization file has a file type of EVE.

The following is an example of an initialization file:

```
set tabs every 5
set left margin 15
set right margin 75
overstrike mode
define key=Ctrl/D erase word
define key=Gold/W start of line
define key=KP5 fill paragraph
!
!Binds the EDT forward function (KP4 on
!EDT keypad) to GOLD F
!
define key=Gold/F EDT KP4
```

A.4 Creating Initialization Files

An initialization file can be specified with the /INITIALIZATION qualifier, defined as EVE$INIT in your LOGIN.COM file, or named EVE$INIT.EVE in your SYS$LOGIN directory. The following command invokes EVE with the initialization file named MY_INIT:

```
$ EDIT/TPU/INIT=WORK1:[ALEXIS]MY_INIT
```

By default, VAXTPU uses the initialization file whose logical name is EVE$INIT. If you define this logical name in your LOGIN.COM file, VAXTPU automatically uses your initialization file when you invoke EVE. For example, you could insert the following command in your LOGIN.COM file:

```
$ DEFINE EVE$INIT WORK1:[ALEXIS]MY_INIT.EVE
```

When EVE starts up, it looks first for a section file, then for a command file, and finally for an initialization file. Since an initialization file is executed after a section file and a command file, the definitions in an initialization file override those in a section file or a command file. For this reason, place commands that define the editing environment in either your command file or your initialization file. Commands that define the environment include the following:

- SET CURSOR BOUND or FREE
- SET FIND WHITESPACE or NOWHITESPACE
- SET KEYPAD
- SET GOLD KEY
- SET LEFT MARGIN
- SET RIGHT MARGIN
- SET SCROLL MARGINS
- SET TABS AT or EVERY
- SET TABS SPACES, MOVEMENT or INSERT
- SET TABS VISIBLE or INVISIBLE
- SET WIDTH
- SET WILDCARD VMS or ULTRIX
- SET WRAP or NOWRAP
- The default mode of the buffer: CHANGE MODE, OVERSTRIKE MODE, or INSERT MODE
- The default direction of the buffer: CHANGE DIRECTION, FORWARD, or REVERSE

A.5 Creating Command Files

A command file is a source program that contains VAXTPU procedures and executable statements. A VAXTPU procedure is a set of related VAXTPU statements that are executed when the procedure name is invoked. The statements and procedures define what happens when you press a key or enter a command.

When you use an EVE command, you are actually invoking a compiled VAXTPU procedure. For example, the EVE command SET KEYPAD EDT invokes the EVE_SET_KEYPAD_EDT procedure in the standard EVE section file.

EVE executes a command file after a section file. For this reason, any key definitions or procedures defined in a command file override those in a section file.

There are two different ways to use a command file. A command file can create an editing environment that is independent of the section file, or a command file can be used to produce a new section file.

Whenever you want to set editing defaults, use a command file because EVE executes the statements in the command file (or initialization file) at startup and applies the new defaults. See Section A.4 for a list of commands that set the editing environment. For example, you can use one command file to set up margins and tabs and a header for a memo and a second command file that sets tabs suitable for writing a financial report.

To create a command file, invoke EVE and specify a filename with an extension of TPU, such as MY_COMMAND.TPU. Once in the editor, enter VAXTPU statements and procedures.

Example A–1 shows a command file that modifies EVE to be more like the EDT editor. This file is meant to create a personal section file named MY_SECTION.TPU$SECTION.

If you intend to use a command file to create a section file, you conclude the file with the SAVE and QUIT statements and include a file specification for the section file, as described in Example A–1. To convert the command file to a section file, invoke EVE with the /COMMAND qualifier. For example:

```
$ EDIT/TPU/COMMAND=MY_COMMANDS
```

EVE executes the commands in the file and saves the compiled procedures and key definitions in the TPU$SECTION file that you name in the SAVE statement. The QUIT statement terminates the editor, returning control to the DCL. At this point, you have a new section file. Therefore, on invoking EVE, the editor automatically reads the section file that you have defined in your LOGIN.COM or in your SYSLOGIN directory.

One advantage of creating a section file from a command file is that a command file can be easily edited. This is especially important when you want to add VAXTPU procedures or add a large number of key definitions.

To use a command file to set the characteristics of the editing environment and add functions to the standard EVE editor, again invoke EVE with the /COMMAND qualifier. For example:

```
$ EDIT/TPU/COMMAND=DATA_SETUP
```

EVE again executes the statements in the command file, but since there is no SAVE statement, the compiled procedures and key definitions are not saved.

If you do not include a command qualifier, EVE searches for the file specified by the logical name TPU$COMMAND. You can define this logical name in your LOGIN.COM file. If this file is not found, VAXTPU then searches for a file named TPU$COMMAND.TPU in the current directory.

Example A–1 Sample EVE Command File

```
!****************************************************************************
!Command file making EVE more like EDT and implementing personal customizations
!****************************************************************************

!Procedure to delete a line and close the gap left by the deletion  (1)

procedure eve_zapline  (2)

eve_end_of_line;                                                    (3)

eve_erase_start_of_line;                                  (4)

eve_delete;

endprocedure  (5)

!Procedure to move the cursor to the beginning of the next paragraph:

procedure eve_next_paragraph  (6)

local   pat1,
        the_range;

pat1 := LINE_BEGIN + LINE_BEGIN + arb (1);
the_range := search_quietly (pat1, FORWARD, EXACT);

if the_range <> 0
then
    position (end_of (the_range));
    return (TRUE);  (7)
else
    return (FALSE);
endif;

endprocedure

!Procedure to make EVE behave more like EDT

procedure eve_mimic_edt

eve_set_keypad_edt;
eve_set_cursor_bound;
eve_set_left_margin(10);  (8)

endprocedure

!Procedure to transpose two characters

procedure eve_transpose

local   whack;

whack := erase_character (1);
move_horizontal (1);
copy_text (whack);

return (TRUE);
endprocedure
```

Example A–1 Cont'd. on next page

Example A–1 (Cont.) Sample EVE Command File

```
!Procedure to make both the screen width and the right margin narrow

procedure eve_narrow_screen

eve_set_width (80);
eve_set_right_margin (79);

endprocedure;

!Procedure to make both the screen width and the right margin wide

procedure eve_wide_screen

eve_set_width (132);
eve_set_right_margin (131);

endprocedure;

!Procedure to toggle screen width and right margin from the current setting
!to the other setting, i.e. change to wide if narrow, change to narrow if wide

procedure eve_change_width     ❾

if get_info (SCREEN, "width") <> 80
then
    eve_narrow_screen;
else
    eve_wide_screen;
endif;

endprocedure;

procedure tpu$local_init    ❿

eve_mimic_edt;              ⓫

eve$define_key ("eve_next_paragraph", CTRL_P_KEY, "Next Para",
                eve$x_user_keys);    ⓬

eve$define_key("eve_zapline", key_name ("o", shift_key), "Zap Line",
               eve$x_user_keys);     ⓭

eve$define_key ("eve_two_windows", F17, "Two Windows", eve$x_user_keys);    ⓮

eve$define_key ("eve_other_window", CTRL_G_KEY, "Other Window",
                eve$x_user_keys);          ⓯

eve$define_key ("eve_get_file('')", key_name (KP6, SHIFT_KEY), "Get File",
                eve$x_user_keys);       ⓰

eve$define_key ("eve_transpose", key_name (F20, SHIFT_KEY), "Transpose",
                eve$x_user_keys);       ⓱

endprocedure

tpu$local_init; ⓲

save ("WORK:[LINCOLN]MY_SECTION.TPU$SECTION");    ⓳

quit;      ⓴
```

Example A–1 illustrates the following points about writing command files:

❶ Document a command file using comments. An exclamation mark starts a comment. When VAXTPU encounters the mark, it ignores everything else on the line.

❷ Start each procedure with the word *procedure*.

❸ End each VAXTPU statement in the procedure with a semicolon.

❹ Each EVE command corresponds to the name of a VAXTPU procedure in the section file of standard EVE (SYS$SHARE:EVE$SECTION.TPU$SECTION).

The names of these VAXTPU procedures can be used in procedures you write, or they can be executed independently as VAXTPU statements. If your command file contains both user-written procedures and executable VAXTPU statements, put all the procedures before any of the executable statements.

❺ End each procedure with the word *endprocedure*.

❻ When you write a procedure implementing a command in EVE, use *eve_* as the first four characters of the procedure name. By following this convention, the procedure name, without the characters eve_, becomes an EVE command. For example, if you write a procedure called EVE_NEXT_PARAGRAPH in a command file, once the section file is compiled, you can use the new EVE command NEXT PARAGRAPH.

Since a command file executes after a section file, a procedure overwrites a command by the same name in the section file. For example, if you name a procedure EVE_ERASE_CHARACTER, the ERASE CHARACTER command executes your procedure, not the standard EVE command.

❼ In order for an EVE command to be usable with a repeat count (either by itself or as part of a learn sequence), it must return TRUE when it succeeds.

❽ If you use a VAXTPU procedure name that requires a parameter, put the parameter in parentheses. For example, the EVE command SET LEFT MARGINS requires a parameter specifying where to set the left margin. The EVE syntax is

```
SET LEFT MARGIN 10
```

But the VAXTPU syntax is

```
eve_set_left_margin(10);
```

Notice that, if a parameter is a string rather than an integer, you must enclose the parameter in quotes. For example, with EVE_SET_SCROLL_MARGINS, the parameter "10%" is a string and must be enclosed in quotes. To pass a null parameter, use an empty pair of quotes, such as eve_spawn("").

❾ A procedure must be compiled before it can be invoked. Then the procedure can be invoked as either an EVE command or as a VAXTPU executable statement. In this example, the procedure named EVE_CHANGE_WIDTH can be invoked as the EVE command CHANGE WIDTH. The procedure name can also be invoked as the VAXTPU executable statement EVE_CHANGE_WIDTH.

❿ Add a procedure called TPU$LOCAL_INIT to a command file that you use as a section file.

This procedure should contain all executable statements (including those calling other procedures) that you want to be defined and executed when EVE is invoked. Because any executable statement in this procedure is executed when EVE is invoked, the statements in this procedure become default settings for the EVE editor.

⓫ This VAXTPU statement invokes the procedure EVE_MIMIC_EDT, which contains VAXTPU statements that change EVE's settings. If you compile the sample command file, save it in a personal section file, and then invoke EVE using that section file, the keypad setting, cursor style,

and left margin are automatically set by the procedure EVE_MIMIC_EDT. As a result, EVE behaves like EDT at startup.

⓬ This VAXTPU statement uses the predefined EVE routine EVE$DEFINE_KEY to define the user-written procedure EVE_NEXT_PARAGRAPH for the key sequence CTRL/P. The EVE$DEFINE_KEY routine ensures that the program bound to the key has an error handler. The VAXTPU built-in DEFINE_KEY does not perform this step. Notice that there are four parameters to EVE$DEFINE_KEY. (This routine uses the same parameters as the VAXTPU built-in DEFINE_KEY.) The first specifies the EVE procedure or command to be bound to a key. The second specifies the key to which the command should be bound. The third specifies the label that EVE should use for the key in the HELP keypad diagram. The fourth is an EVE variable specifying the keymap list in which the key definition should be saved. Use the variable name EVE$X_USER_KEYS for the fourth parameter unless you are an advanced user implementing a special application.

⓭ This VAXTPU statement defines the procedure EVE_ZAPLINE for the key sequence GOLD/O. This line demonstrates the VAXTPU syntax to use when defining a sequence consisting of the GOLD key plus a letter key. Note that using "shift_key" makes the definition case insensitive, so that both *o* and *O* are defined.

⓮ This statement defines the EVE command TWO WINDOWS for the F17 key.

⓯ This statement defines the EVE command OTHER WINDOW for the key sequence CTRL/G.

⓰ This statement defines the EVE command GET FILE for the key sequence GOLD/KP 6. Note that this keypad binding supersedes the previous definition of the GOLD/KP 6 key. The EVE_MIMIC_EDT procedure sets the keypad to EDT and the EDT keypad then binds the GOLD/KP 6 key sequence to the INSERT HERE command. However, this VAXTPU statement changes the key binding to the GET FILE command. The pair of single quotes passes a null argument to the procedure.

⓱ This statement defines the procedure EVE_TRANSPOSE for the key sequence GOLD/F20.

⓲ The statement TPU$LOCAL_INIT calls the procedure TPU$LOCAL_INIT, which is then executed, creating new default settings for EVE.

⓳ Enter the SAVE statement to create a new section file. In parentheses and quotation marks, specify the device, directory, and filename of the section file. To update a section file, specify the existing personal section file.

⓴ Add the statement QUIT as the last statement in a command file if the file is to be compiled as a section file. QUIT ends EVE processing and returns control to the DCL.

A.6 Modifying the $DEFAULTS$ Buffer

In EVE, each new buffer you create uses default settings for the editing mode, direction, margins, and tab stops. The system buffer $DEFAULTS$ lists the default buffer characteristics.

These characteristics are either the EVE defaults or they derive from the command settings established in your section file, command file, and initialization file for the initial editing buffer. The initial buffer is the main buffer or the buffer cited on the EVE command line.

For example, if your initialization file sets margins, tab stops, cursor direction, and editing mode, then those settings are also set for the buffer $DEFAULTS$. EVE applies the characteristics listed in the $DEFAULTS$ buffer to all subsequent buffers that you create.

To see the current characteristics of the buffer, enter the command SHOW DEFAULTS BUFFER. EVE displays the following type of information:

```
Information about buffer $DEFAULTS$

    Not modified                Left margin set to 1
    Mode: Insert                Right margin set to 79
    Direction: Forward
    Max lines: No limit

Tab stops set every 8 columns
```

You can also modify characteristics of the $DEFAULTS$ buffer interactively. Move to the $DEFAULTS$ buffer by entering the command BUFFER $DEFAULTS$. Then enter the appropriate SET command; for example, SET LEFT MARGIN 7. The $DEFAULTS$ buffer now has a left margin of 7. As a result, any new buffers created during the editing session have a left margin of 7.

B EDT Commands and Equivalent EVE Commands

This appendix lists basic EDT commands and the EVE equivalents.

The EVE command SET KEYPAD EDT provides most EDT keypad functions. However, EVE does not implement or support EDT line-mode commands and "nokeypad" commands. SET KEYPAD EDT does not implement GOLD-key equivalents for EDT control keys. For example, GOLD/U and GOLD/W are not defined, although CTRL/U and CTRL/W are defined.

If you are accustomed to using an EDT initialization file, you can create a similar startup file for EVE, by using the corresponding EVE commands or VAXTPU procedures. See Appendix A for a description of startup files. For example, the following initialization files set up equivalent editing environments:

```
! EDT initialization file
!
set notruncate
set wrap 70
set cursor 7:14
def key gold 10 as "chglw."
def key gold u as "chguw."
def key func 34 as "ext =main.."
```

```
! EVE initialization file

set wrap
set right margin 70
set scroll margins 7 7
def key=gold/pf2 lowercase word
def key=gold/u uppercase word
def key=f20 buffer main
```

Table B–1 Corresponding EDT and EVE Commands

EDT Command or Function	Equivalent EVE Command
line-number	LINE. EVE does not display line numbers. Use the WHAT LINE command to display the current line number. Line numbers are always integers.
Repeat count with GOLD key	Implemented by SET KEYPAD EDT. Alternatively, use the EVE command REPEAT and type the repeat count.
ADJUST TAB or CTRL/T	Not implemented. To set tab stops, use the EVE command SET TABS or define a key for the WPS Ruler key (GOLD/R).
ADVANCE or KP4	FORWARD.
APPEND or KP9	Implemented by SET KEYPAD EDT.
ASC	No equivalent EVE command, but you can usually get the same effect by using QUOTE.
BACKUP or KP5	REVERSE.
BOL or BACKSPACE CTRL/H	Implemented by SET KEYPAD EDT. You can also use the EVE command START OF LINE or press the GOLD/Left arrow key.
BOTTOM or GOLD/KP4	BOTTOM or the GOLD/Down arrow key.
CHARACTER or KP3	Implemented by SET KEYPAD EDT.
CLEAR	DELETE BUFFER.
COMMAND	DO. However, EVE does not accept EDT line-mode commands.

Table B–1 (Cont.) Corresponding EDT and EVE Commands

EDT Command or Function	Equivalent EVE Command
COMPUTE TAB LEVEL or CTRL/A	Not implemented. To set tab stops, use the EVE command SET TABS or define a key for the WPS Ruler key (GOLD/R). EVE defines CTRL/A as CHANGE MODE, so it switches between insert and overstrike modes.
COPY	No equivalent EVE command, but you can get the same effects by using STORE TEXT and INSERT HERE.
CUT or KP6	REMOVE.
CHNGCASE or KP1	Implemented by SET KEYPAD EDT. You can also use the EVE commands CAPITALIZE WORD, LOWERCASE WORD, and UPPERCASE WORD.
CHGL	LOWERCASE WORD. In EVE Version 2.0, this command works on the currently highlighted text or on the current word.
CHGU	UPPERCASE WORD. In EVE Version 2.0, this command works on the currently highlighted text or on the current word.
CTRL/C	In EVE, CTRL/C can cancel an operation, such as a repeat count or a global replacement. However, its use is not recommended because CTRL/C interferes with the journaling facility. If you use CTRL/C, immediately exit from EVE and then restart the editing session.
CTRL/L (insert form feed)	INSERT PAGE BREAK. EVE always puts a page break on a line by itself.
DATE	No equivalent EVE command, but you can define a key for WPS Insert Date Time (GOLD/\ or GOLD/|). For example, the command DEFINE KEY=ctrl/d wps gold/| defines CTRL/D to insert the current date and time.
DECREASE TAB LEVEL or CTRL/D	Not implemented. To set tab stops, use the EVE command SET TABS or define a key for the WPS Ruler key (GOLD/R).
DEFINE KEY or CTRL/K	DEFINE KEY. Note that the EVE syntax is different, and EVE generally uses key names like the DCL. Setting the EDT keypad in EVE defines CTRL/K as LEARN.
DEFINE MACRO	No equivalent EVE command. However, you can usually create a learn sequence or VAXTPU procedure to do the same thing. In some cases you can use an EVE initialization file.
DEL BOL or CTRL/U	Implemented by SET KEYPAD EDT. You can also use the EVE command ERASE START OF LINE, although it is slightly different.
DEL C or COMMA	Implemented by SET KEYPAD EDT. Also, the EVE command ERASE CHARACTER is functionally the same. However, in EVE, the operation is sensitive to the mode of the buffer (insert or overstrike).
DEL EOL or GOLD/KP2	Implemented by SET KEYPAD EDT.
DEL L or PF4	Implemented by SET KEYPAD EDT. You can also use the EVE command ERASE LINE.
DEL W or MINUS	Implemented by SET KEYPAD EDT. The EVE command ERASE WORD performs a similar function.

Table B–1 (Cont.) Corresponding EDT and EVE Commands

EDT Command or Function	Equivalent EVE Command
DELETE	DELETE. In EVE, the delete operation is sensitive to the mode of the buffer (insert or overstrike).
DOWN	MOVE DOWN.
ENTER	RETURN.
EOL or KP2	Implemented by SET KEYPAD EDT. You can also use the EVE command END OF LINE (or press the GOLD/RIGHT arrow key), although it is slightly different.
EXIT	EXIT or CTRL/Z or F10. Note that in EVE, exiting does not write out a file unless you have made changes to it.
EXT	DO. Keep in mind that EVE does not have a separate line-mode state.
FILL or GOLD/KP8	FILL, FILL PARAGRAPH, or FILL RANGE. With EVE Version 2.0, the FILL command does a FILL RANGE if you have selected some text; otherwise, it does a FILL PARAGRAPH.
FIND	FIND. However, the rules for case sensitivity are different in EVE.
FIND=	BUFFER. In EVE, buffer names are usually the same as the file in that buffer.
FNDNXT or PF3	Implemented by SET KEYPAD EDT. You can also press FIND twice.
HELP	HELP.
INCLUDE	INCLUDE FILE. In EVE, the included file is copied into the buffer before the start of the current line.
INCREASE TAB LEVEL or CTRL/E	Not implemented. To set tab stops, use the EVE command SET TABS or define a key for the WPS Ruler key (GOLD/R). Also, EVE defines CTRL/E as END OF LINE.
INSERT	No equivalent EVE command, because EVE does not have a separate line-mode state. In an initialization file, you can use the EVE command TPU and the VAXTPU procedure COPY_TEXT.
LEFT	MOVE LEFT.
LINE or KP0	MOVE BY LINE. The EVE command LINE takes a line number and optionally a procedure name.
LINEFEED or CTRL/J	Implemented by SET KEYPAD EDT. The EVE command ERASE PREVIOUS WORD is similar.
MOVE	No equivalent EVE command, but you can get the same effect by using REMOVE and INSERT HERE.
OPEN LINE or GOLD/KP0	Implemented by SET KEYPAD EDT.
PAGE or KP7	MOVE BY PAGE.
PASTE or GOLD/KP6	INSERT HERE.
PRINT	No equivalent EVE command. However, you can create a learn sequence or VAXTPU procedure to do the same thing; that is, copy the current buffer, add page breaks, and write out the file. EVE does not insert line numbers as text in the buffer.
QUIT	QUIT. If you have modified any buffers, EVE prompts you to confirm that you are quitting.

Table B–1 (Cont.) Corresponding EDT and EVE Commands

EDT Command or Function	Equivalent EVE Command
REF or CTRL/W	REFRESH. EVE defines CTRL/R as REMEMBER (to end a learn sequence).
REPLACE or GOLD/KP9	Implemented by SET KEYPAD EDT. The EVE command REPLACE is similar to the EDT command SUBSTITUTE.
RESEQUENCE	No equivalent EVE command. EVE line numbers are always integers (no fractions). To find out the current line number, use WHAT LINE.
RESET or GOLD/PERIOD	RESET or GOLD/SELECT.
RIGHT	MOVE RIGHT.
SECT or KP8	Implemented by SET KEYPAD EDT. You can also use NEXT SCREEN and PREVIOUS SCREEN.
SELECT or PERIOD	SELECT.
SET [NO]AUTOREPEAT	No equivalent EVE command.
SET CASE	No equivalent EVE command.
SET COMMAND	@ command. For example, the command @ MYEVE executes an initialization file called MYEVE.EVE.
SET CURSOR	SET SCROLL MARGINS. EVE scroll margins are set at the top and bottom of the window, whereas in EDT, they are both set from the top. For example, the command SET SCROLL MARGINS 2 3 sets scroll margins at two lines from the top of the window and three lines from the bottom of the window. Also, in EVE, you can specify the scroll margins as percentages of the window height.
SET ENTITY	No equivalent EVE command.
SET [NO]FNF	No equivalent EVE command.
SET MODE	No equivalent EVE command. EVE is always a full-screen editor and does not have a separate line-mode state.
SET HELP	No equivalent EVE command, but you can define the logical names EVE$HELP and EVE$KEYHELP at the DCL level to specify other HELP files.
SET [NO]KEYPAD	SET KEYPAD EDT and SET KEYPAD NOEDT.
SET LINES number	No equivalent EVE command. However, if you use multiple windows, you can shrink and enlarge them. The size of the EVE main window depends on the size of your terminal screen (established with the DCL command SET TERMINAL/PAGE).
SET [NO]NUMBERS	No equivalent EVE command.
SET PARAGRAPH	No equivalent EVE command. In EVE, a paragraph (for purposes of FILL commands) is bound by blank lines, the top or bottom of the buffer, page breaks, or RUNOFF commands.
SET PROMPT	No equivalent EVE command.
SET [NO]REPEAT	No equivalent EVE command.
SET [NO]QUIET	No equivalent EVE command.
SET SEARCH	No equivalent EVE command. However, you can get some of the same effects by using WILDCARD FIND.

Table B–1 (Cont.) Corresponding EDT and EVE Commands

EDT Command or Function	Equivalent EVE Command
SET SCREEN	SET WIDTH. By default, the width of the EVE main window is the width of your terminal screen (established with SET TERMINAL/WIDTH). Typically, this is 80 columns.
SET [NO]SUMMARY	No equivalent EVE command.
SET [NO]TAB	SET TABS. The EVE syntax is different. Also, EVE does not implemenent EDT-style keys for computing tabs.
SET TERMINAL	No equivalent EVE command.
SET TEXT	No equivalent EVE command.
SET [NO]TRUNCATE	SET NOWRAP and SET WRAP.
SET [NO]VERIFY	No equivalent EVE command.
SET WORD [NO]DELIMITER	No equivalent EVE command. In EVE, a word is bounded by spaces, tabs, or the start or end of a line. A space before a word is considered part of that word (for the commands ERASE WORD, CAPITALIZE WORD, LOWERCASE WORD, and UPPERCASE WORD).
SET [NO]WRAP	SET RIGHT MARGIN.
SHL	SHIFT LEFT. The EVE command requires a numeric parameter (an integer, specifying the number of columns you want to shift the window to the left).
SHOW BUFFER	The following EVE commands: SHOW—To display information about the current buffer. SHOW BUFFERS—To get a list of the buffers you have created. SHOW DEFAULTS BUFFER—To show information about the $DEFAULTS$ buffer, which lists the settings used when you create buffers. SHOW SYSTEM BUFFERS—To get a list of the buffers created by EVE, such as the message buffer, commands buffer, and Insert Here buffer.
SHOW KEY	SHOW KEY. Either type the key name on the command line or press the key you want to know about, in response to an EVE prompt.
SHOW VERSION	SHOW SUMMARY.
SHR	SHIFT RIGHT. The EVE command requires a numeric parameter (an integer, specifying the number of columns you want to shift the window to the right).
SPECINS or GOLD/KP3	Implemented by SET KEYPAD EDT. You can also enter control codes by using the EVE command QUOTE, and you can enter DEC Multinational Characters by using the COMPOSE CHARACTER key on VT200-series and VT300-series terminals.
SUBS or GOLD/ENTER	Implemented by SET KEYPAD EDT. You can also use the EVE command REPLACE, which corresponds to the EDT line-mode command SUBSTITUTE.
SUBSTITUTE	REPLACE. The EVE command does not use qualifiers. The search follows the same rules as FIND for case sensitivity.

Table B–1 (Cont.) Corresponding EDT and EVE Commands

EDT Command or Function	Equivalent EVE Command
TAB ADJUST	No equivalent EVE command. To set tab stops, use the EVE command SET TABS or define a key for the WPS Ruler key (GOLD/R).
TOP or GOLDF/KP5	TOP or the GOLD/Up arrow key.

TYPE	No equivalent EVE command, but you can use NEXT SCREEN and PREVIOUS SCREEN to scroll through a file. Also, you can define keys for WPS Scroll Backup (GOLD/KP1) and WPS Scroll Advance (GOLD/KP0). For example:
	`Command: DEFINE KEY=gold/e5 wps gold/kp1`
	`Command: DEFINE KEY=gold/e6 wps gold/kp0`
	Defines GOLD/PREV SCREEN as Scroll Backup and GOLD/NEXT SCREEN as Scroll Advance.

UND C or GOLD/COMMA	RESTORE CHARACTER. In EVE, the function is sensitive to the mode of the buffer (insert or overstrike).
UND L or GOLD/PF4	RESTORE LINE.
UND W or GOLD/MINUS	RESTORE WORD.
UP	MOVE UP.
WORD or KP1	MOVE BY WORD.
WRITE	WRITE FILE. The EVE command always writes out the entire current buffer. You can create a VAXTPU procedure to write out a range.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Reader's Comments

Guide to VMS Text Processing
AA–LA13A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page Description

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------

Reader's Comments

Guide to VMS Text Processing
AA–LA13A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------