PDP–11
Keypad Editor User's Guide

AA–H853A–TC

February 1981

This document describes how to use the PDP–11 Keypad Editor (KED and K52) while working with the RT–11, RSX–11M, RSX–11M–PLUS, and IAS operating systems. It provides the information required to use the VT100 and VT52 video terminals for inspecting, creating, and editing ASCII text files.

This manual supersedes the *RT–11 Keypad Editor User's Guide*, AA–H366A–TC. This manual contains Update Notice 1, AD–H853A–T1 and Update Notice 2, AD–H853A–T2.

Operating System: RT–11 Version 5
RSX–11M Version 3.2
RSX–11M–PLUS Version 1
IAS Version 3.1

To order additional documents from within DIGITAL, contact the Software Distribution Center, Northboro, Massachusetts 01532.

To order additional documents from outside DIGITAL, refer to the instructions at the back of this document.

digital equipment corporation · maynard, massachusetts

First Printing, March 1980
Updated, February 1981
Updated, July 1984

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Printed in U.S.A.

A postage-paid READER'S COMMENTS form is included on the last page of this document. Your comments will assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

	digital™	
DEC	EduSystem	RSTS
DECmate	IAS	RSX
DECnet	MASSBUS	RT–11
DECsystem–10	PDP	UNIBUS
DECSYSTEM–20	PDT	VAX
DECUS	P/OS	VMS
DECwriter	Professional	VT
DIBOL	Rainbow	Work Processor

M32900

Contents

Chapter 3 Inspecting a File

Chapter 4 Editing or Creating a File

Chapter 5 Special Keypad Editor Features for Writers and Programmers

Appendix PDP–11 Keypad Editor Messages

Index

Figures

Tables

TO THE READER

The PDP–11 Keypad Editor is a text file editor that DIGITAL has designed for use with VT100 terminals that have special function keypads. A version of the keypad editor also can be used with VT52 and VT55 terminals. The keypad editor is suitable as the first text editor for a new user of the RT–11 V5, RSX–11M V3.2, RSX–11M–PLUS, or IAS V3.1 operating systems.

This preface describes:

- The relationship between this manual and the other keypad editor documentation.
- The skills that you should have before beginning to study this manual.
- The contents of this manual and the other documentation for the keypad editor.
- The symbols and conventions used in this manual and in the other documents for the keypad editor.

Keypad Editor Documentation

The keypad editor documentation is composed of three items:

- This manual, the *PDP–11 Keypad Editor User's Guide.*
- The *PDP–11 Keypad Editor Reference Card.*
- Special summaries called *HELP forms* that you can display on your screen while you are using the keypad editor.

This manual is the principal document for the keypad editor. It presents the keypad editor features in logically related groups, such as features for searching, for inserting, and for erasing.

The descriptions of the individual features are organized so that approximately the same technical level of detail appears at the same point in each description. For example, the description of each keypad function has the following five parts:

- Heading — The heading shows the function name and lists the VT100 and VT52 keys that control the function.
- Purpose and effect of the function.
- Examples of what the function does.
- Special details.
- Summary of situations when the function is invalid.

The *PDP-11 Keypad Editor Reference Card* contains a brief description of all keypad editor features.

The HELP forms include diagrams of the VT100 and VT52 keypads and summaries of the keypad editor commands and functions. The diagrams show the keypad editor features controlled by each keypad key. The summaries show the forms of the commands and the features controlled by each keyboard key.

Prerequisites

For RT–11 V5 Users —

Have the keypad editor program copied to your system volume or to another volume that you want to use. Complete Chapters 1 - 4 in the *Introduction to RT-11*. Study Chapter 4 in the *RT-11 User's Guide*, and learn to use the interactive commands DIR, EDIT, R, and RUN.

For RSX-11M V3.2 Users —

Check that the keypad editor programs have been installed. Complete the *RSX-11M Beginner's Guide* and the sections of the *Introduction to RSX-11M* that cover file specifications and default storage volumes.

For RSX-11M-PLUS Users —

Check that the keypad editor programs have been installed. Complete the *Introduction to RSX-11M-PLUS*. Study the *RSX-11M/M-PLUS MCR Operations Manual*, and learn to start and stop the Monitor Console Routine (MCR).

For IAS V3.1 Users —

Your IAS system already contains keypad editor programs for use with the DCL or MCR command language interfaces. If your installation uses DCL, read Chapters 1-6 of the *IAS PDS User's Guide*. Read Chapter 14 for details of PDS commands. If your installation uses MCR, read Chapters 1, 2, 3, and 6 of the *IAS MCR User's Guide*.

Chapter Summary

Chapter 1 discusses the general concepts of text file editing and summarizes how the keypad editor processes input files and stores output files. Chapter 1 also offers suggestions about how you can learn to use the keypad editor and includes an example that demonstrates several of the keypad editor features.

Chapter 2 describes how to start and stop the editor on each of the supported operating systems. This chapter tells how to use the keypad editor to inspect, create, and edit text files. The common sequences of operating system commands and keypad editor command lines are illustrated.

Chapter 3 describes the functions that are active when you are only inspecting (not creating or changing) a text file. This chapter covers functions that display different parts of a file and search for strings of characters. The chapter also covers other functions that are essential when you are making changes to a text file.

Chapter 5 describes several supplementary features. This chapter covers the uses of auxiliary files, sequences of keypad editor functions (called "macros"), and features for formatting text files, indenting high-level language source programs, and reordering MACRO–11 local symbols.

The Appendix lists the keypad editor messages in alphabetical order. An explanation of the causes and a description of the remedies follow each message.

Symbols and Conventions

This manual and the *PDP–11 Keypad Editor Reference Card* use the following symbols and conventions. Although most of them are the same as the symbols and conventions used in other documents for PDP–11 software, the VT100 terminal and the video orientation of the keypad editor require a few changes.

System prompts	Indicate that the system is ready for you to enter a command. The system prompts are: For RT–11: a period (.) For RSX–11M: the characters `MCR>` or a right-angle bracket (>) by itself. For RSX–11M–PLUS: the dollar sign ($) or the RSX–11M prompts (when the MCR is running).
Red print	Indicates the characters typed on the keyboard in examples of commands. Black print in these examples indicates the characters displayed by the system or by the keypad editor.
Uppercase letters (In commands)	Indicate the characters that you must type on the keyboard (see "Lowercase letters" also).
Lowercase letters (In commands)	Indicate the parts of commands or command strings that you must supply (see "Uppercase letters" also).
Underlined letters (In general forms of commands)	Indicate the shortest valid abbreviation of a keypad editor command.
CTRL/	Indicates a combination of the control key and another keyboard key. For example, for CTRL/U hold down the CTRL key and press the U key.
Square brackets (In general forms of commands)	Enclose an optional term or optional characters (do not type square brackets as part of a command unless the instructions explicitly require them).

Braces	Enclose a list of two or more terms from which you must choose and type one (do not type braces as part of a command).
	Enclose a keypad editor function name or name of a labelled key.
Dot matrix letters	Indicate prompts, short status messages, and keypad editor commands.

Chapter 1
Introduction

The PDP-11 Keypad Editor is a program that you can use to create and change computer files that DIGITAL calls *text files.* Text files contain only the untranslated letters, digits, and symbols that you can type on your keyboard. You cannot use the keypad editor to change other types of computer files, such as binary files, relative files, indexed files, or object files, because they contain special translated characters that the keypad editor program cannot read.

1.1 What Are Text Files For?

A text file can be used for various purposes, such as:

- For a memo or a collection of memos.
- For a computer program or the data a computer program requires.
- For a table of values, arranged in columns and rows, to attach to a report.
- For a list of names and addresses.
- For a chapter of a book or an entire book.

Regardless of the purpose of a text file, the keypad editor has features for you to use when you want to create or edit a file. The general procedure for editing a text file is as follows:

1. Start the keypad editor program. Chapter 2 describes the commands to use for each operating system that supports the keypad editor.
2. Specify the file that you want to edit (the main input file) and the new version of the file that will include your changes (the main output file). Chapter 2 describes how to type these file specifications.
3. Use the keypad editor's functions and commands to display different parts of the file and to make changes. Chapters 3 and 4 describe the simplest and most frequently used features, and Chapter 5 describes special features that you may also want to use.

4. Use the keypad editor EXIT command to close the output file and have your system store it.

5. Continue editing by typing other file specifications as in Step 2. Or stop the keypad editor by using the appropriate procedure for your operating system. Chapter 2 describes the requirements.

1.2 What Are KED and K52?

DIGITAL distributes two versions of the PDP-11 Keypad Editor. KED is the version for use with the VT100 family of video terminals. K52 is the version for use with VT52 and VT55 video terminals.

KED and K52 provide the same basic features. Any editing that you can do with KED you can also do with K52. The keypad editor versions are different only because of the different hardware features that VT100s and VT52s have.

In this manual and in the *PDP-11 Keypad Editor Reference Card*, the term *VT100* refers to all of DIGITAL's video terminals that are compatible with the VT100 terminal, and the term *VT52* refers to VT52s and VT55s.

The following list summarizes the general differences between KED and K52.

1. Keyboards and keypads

   Because the VT100 and VT52 terminals have different keyboards and keypads, KED and K52 in a small number of cases use different keys to control the editing functions.

2. Special graphic symbols

   Because the terminals have different sets of special graphic symbols, KED and K52 display files in slightly different ways.

3. Video features

   Because the VT100 has several video features that the VT52 does not have, such as reverse video display and 132-column screen width, KED provides six more commands than K52. These commands allow you to adjust VT100 features while you are editing.

1.3 What Does the Keypad Editor Do?

The keypad editor allows you to create, inspect, and edit files. In doing so, it manipulates the files that you have specified, processes the characters that your terminal sends when you type the keyboard and keypad keys, and controls the screen display on your terminal. For example, the keypad editor:

1. Opens and closes the files that you specify.

2. Inserts the letters, numbers, and symbols that you type on the keyboard.

3. Executes the commands that you enter.

4. Processes the editing functions that you specify by typing the keypad keys.

5. Immediately displays the effect of each function or command as you use it.

1.3.1 The Difference Between Functions and Commands

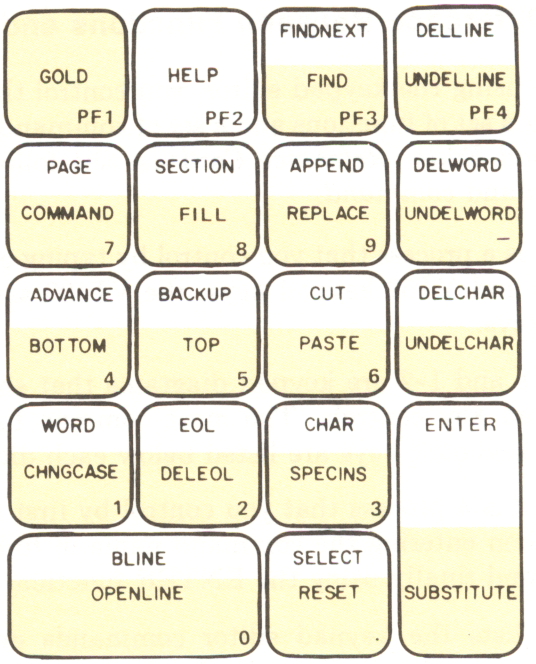
While you are using the keypad editor, you control the different editing processes by using a set of functions and a set of commands. This manual and the *PDP-11 Keypad Editor Reference Card* use the following definitions of the terms *function* and *command*:

1. A *function* is a process that you control by typing one or two keystrokes on the VT100 and VT52 keypad keys, the VT100 arrow keys, or ten of the keyboard keys.

   Figures 1–1 and 1–2 are keypad diagrams that show the functions that each keypad key controls. The most common functions that are controlled by keyboard keys are listed below each diagram.

2. A *command* is a process that you control by first using the COMMAND function, then entering the command name in full or abbreviated on the keyboard, and finally using the ENTER function.

   Table 1–1 lists the keypad editor commands and describes each one briefly.



Note: The letters, numbers and characters in the lower right corner of the keys are what actually appear on the keys.

Common Keyboard Functions

DELETE	Erases the character to the left of the cursor
LINEFEED	Erases the word to the left of the cursor
CTRL/U	Erases the line to the left of the cursor
CTRL/W	Repaints the screen
CTRL/C	Cancels
GOLD *integer*	Repeats any function except DELETE or SPECINS

Figure 1–1: Keypad Diagram for the KED Keypad Editor

Note: The letters, numbers and characters in the lower right corner of the keys are what actually appear on the keys.

Common Keyboard Functions

DELETE	Erases the character to the left of the cursor
LINEFEED	Erases the word to the left of the cursor
CTRL/U	Erases the line to the left of the cursor
CTRL/W	Repaints the screen
CTRL/C	Cancels
GOLD *integer*	Repeats any function except DELETE or SPECINS

Figure 1-2: Keypad Diagram for the K52 Keypad Editor

Table 1-1: Summary of Commands and Keyboard Functions

COMMAND or FUNCTION	PURPOSE
CLEAR PASTE	Clears the paste buffer.
CLOSE	Closes the auxiliary output file.
EXIT	Closes all open files.
FILL	Reformats the select range to fit within the current right margin (also available as a function on the VT100 keypad).
General editing functions	
CTRL/C	Cancels the command or function that is being processed.
CTRL/U	Erases one line to the left.
CTRL/W	Repaints the screen.
CTRL/Z	Cancels editor prompts.
DELETE	Erases one character to the left.
LINEFEED	Erases one word to the left.
GOLD nnn	Repeats the next function nnn times.
INCLUDE nnn PAGES	Copies pages (as currently defined) from the auxiliary input file.
nnn[LINES]	Copies text lines.
REST	Copies the remainder of the file.
LEARN	Begins recording a sequence of commands and functions as an editor macro.
Editor macro functions	
GOLD S	Stops recording the macro.
GOLD X	Executes the macro, as currently recorded.
LOCAL[starting-value[increment]]	Renumbers MACRO-11 local symbols.
[OPEN]INPUT file-spec	Opens an auxiliary input file.
[OPEN]OUTPUT file-spec	Opens an auxiliary output file.
PURGE	Purges the auxiliary output file.
QUIT	Purges all open output files.
SET [ENTITY]PAGE { nnn[LINES] / "string" / 'string' }	Defines a page by line count. Defines a page by a marker string of one or more characters.
SET [ENTITY]SECTION { nnn[LINES] / "string" / 'string' }	Defines a section by line count. Defines a section by a marker string of one or more characters.
SET QUIET	Sets the VT100 to signal with the reversed background.
SET NOQUIET	Sets the VT100 to signal with the terminal bell.
SET [SCREEN]80	Sets the VT100 to the 80-column width.
SET [SCREEN]132	Sets the VT100 to the 132-column width.
SET [SCREEN]LIGHT	Sets the VT100 to the light background.
SET [SCREEN]DARK	Sets the VT100 to the dark background.
SET [SEARCH]GENERAL	Matches alphabetically regardless of case.

(continued on next page)

Table 1–1 (Cont.): Summary of Commands and Keyboard Functions

COMMAND or FUNCTION	PURPOSE
SET [SEARCH]EXACT	Matches alphabetically and requires the same case.
SET [SEARCH]BEGIN	Leaves the cursor at the beginning of the target.
SET [SEARCH]END	Leaves the cursor at the end of the target.
SET [SEARCH]BOUNDED	Limits searching to a page as currently defined.
SET [SEARCH]UNBOUNDED	Allows searching beyond the current page.
SET TABS[indent]	Enables structured tabs.
SET NOTABS	Enables structured tabs.
Structured tab functions: GOLD E GOLD D GOLD A	 Increments the tab level-counter. Decrements the tab level-counter. Aligns the tab level-counter to the cursor.
SET WRAP[nn]	Sets the right margin and enables word wrap.
SET NOWRAP	Disables word wrap.
SKIP { nnn PAGES / nnn[LINES] / REST }	Skips pages (as currently defined) within the auxiliary input file. Skips text lines. Skips the remainder of the file.
TABS ADJUST[{ ± }]nnn	Change indentation.
WRITE { nnn PAGES / nnn[LINES] / REST / SELECT }	Writes pages (as currently defined) to the auxiliary output file. Writes text lines. Writes the remainder of the file. Writes the select range.

1.3.2 General Procedure for Using Functions

The most frequently used editing process is insertion of new material. Therefore, insertion is the keypad editor's default function. To type an insertion for a file that you are creating or editing, use the keyboard keys in the same way that you use a typewriter. The standard function of each keyboard key, including the digit keys in the top row, is to insert the character that is printed on the key.

All of the VT100 keypad keys and all but one of the VT52 keypad keys control two keypad editor functions.

In Figures 1–1 and 1–2, the function names that are printed in black are the standard functions of the keypad keys. To use a standard function, press the corresponding key by itself. For example, to use the DELWORD function, press the VT100 keypad key labeled with a minus sign (–) or the VT52 keypad key labeled 9.

The function names that are printed with a yellow background are the alternate functions of the keypad keys. To use an alternate function, use the GOLD function first as a prefix and then press the key for the function. For example, to use the TOP function, press the VT100 keypad key labeled PF1 and then the keypad key labeled 5. With a VT52, press the unlabeled blue key on the keypad and then the keypad key labeled 5.

Five standard and five alternate functions are controlled by keys on the keyboard. The functions are described in Chapters 3, 4, and 5. To use them follow the same procedure as for the keypad functions.

1.3.3 General Procedure for Using Commands

A command is a process that you control by completing three steps:

1. Use the COMMAND function to specify that you want to enter a command. The editor displays the following prompt for commands:

   ```
   COMMAND:
   ```

2. Enter the command name in full or abbreviated on the keyboard.

3. Use the ENTER function to specify that your command is complete.

For example, to stop editing a file and make the keypad editor store your new version, use the EXIT command as follows:

- With a VT100:
  (1) Press the PF1 key, then the 7 key on the keypad.
  (2) Enter the EXIT command by typing `EXIT` on the keyboard.
  (3) Press the Enter key on the keypad.
- With a VT52:
  (1) Press the blue key, then the 7 key on the keypad.
  (2) Enter the EXIT command.
  (3) Press the Enter key on the keypad.

This manual uses a concise method to describe sequences of commands and functions. Figure 1–3 illustrates how the manual describes the three-step sequence for the EXIT command.

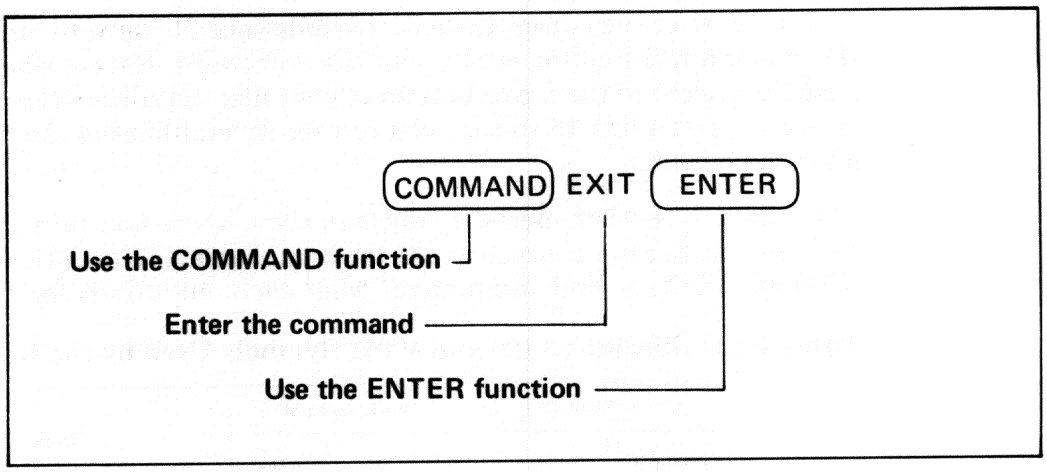


Figure 1–3: Documentation Method for a Sequence of Commands and Functions

As Figure 1–3 illustrates, each function name is enclosed in an oval, and each command is printed with dot matrix letters like the ones you see on your screen.

1.3.4 What You See on the Screen

The keypad editor uses the screen on your terminal like a window to show you 24 screen lines of your file*. When you start editing a file, you see the first 24 screen lines. By using the keypad editor functions that are described in Chapter 3, you can make the editor scroll the file upward and downward on your screen so that you can see different 24-line segments of the file. By using other functions and commands, you can change the file by inserting and erasing material. After each change, the editor immediately shows the current file contents or displays the message WORKING . . . on your screen until the process is complete.

The editor always displays a special blinking symbol on the screen called the cursor. This manual uses the symbol ▒ in examples to show where the cursor is.

The cursor marks the location in your file where the editor will insert a character if you type one. Therefore, to insert new material in your file, you need only to use the editor functions to advance or back up the cursor to the location where you want the insertion to appear and then type the insertion.

The cursor also shows where the editor will begin processing the other functions and commands. For example, to erase something in your file, move the cursor to the correct location and then use appropriate functions.

---

* Models of the VT100 terminal that do not include the advanced video option use 24 screen lines when the line length is set to 80 characters but only 14 screen lines when the line length is set to 132 characters.

As you advance the cursor, the editor scrolls your file upward, and as you back the cursor up, the editor scrolls your file downward. Except when the cursor's location is close to the top or bottom of your file, the editor generally keeps the cursor in screen line 16 so that you can see several lines of the file before and after the cursor.

The editor uses other special symbols to show where non-printing characters, such as the Escape character, are located. Table 1–2 lists the symbols that KED and K52 use and summarizes what each one stands for.*

Table 1–2: Special VT100 and VT52 Symbols Used by the Keypad Editor

VT100 Symbol		VT52 Symbol		Usage
In prompts	In text	In prompts	In text	
▮ or _	▮ or _	_	_	The cursor
None	▒	None	▮	The end-of-file symbol
None	REVERSE or _	None	None	VT100 marking for characters in a select range+
↹	None	^I	None	The Horizontal Tab character
↓	↓	^K	÷	The Vertical Tab character
←	←	^M	None	The Carriage Return character
↓	None	^J	None	The Linefeed character
FF	FF	^L		The Formfeed character
None	ESC	None	±	The Escape character
← ↓	None	^M ^J	None	The New Line Terminator
None	◆	None	→	Each wrapped line

+For models of the VT100 terminal without the advanced video option, the keypad editor marks select ranges as follows:

- With the block cursor ▒, KED uses reverse video.
- With the underscore cursor (_), KED uses underscoring.

For short prompts, the editor temporarily erases the top two screen lines and displays the prompt in the top screen line. After you respond to the prompt, the editor restores the two screen lines. For messages, the editor uses the same process, except that it temporarily erases the bottom three screen lines.

With VT100 terminals, KED also uses the VT100's built-in video and graphic features to clarify some editing processes. For example, KED displays messages and your responses to prompts with the reverse video feature.

---

* Table 1–2 shows the symbols that DIGITAL distributed most recently with VT52 terminals. Older VT52s may display different symbols. Check the table of alternate characters in the *DECscope User's Manual* for your unit if K52 displays different symbols from those shown here.

1.4 What You Can Do to Your Files

You can use the keypad editor to inspect a file, create a new file, or edit a file. This section tells you how to use the editor to manipulate the file in each case. Chapter 2 describes how to enter file specifications.

1.4.1 Inspecting Files

When you inspect a file, you are able to use only those functions and commands that display different parts of the file. Therefore, you cannot make any changes to the file. Inspecting a file can be especially useful when you want to review a file or teach someone how to use the keypad editor.

To inspect a file, specify an input file but no output file. The editor opens the input file and displays the top 24 screen lines of the file, with the cursor on the first character that is in the file. To look at other parts of the file, you use the editor's commands and functions to advance and back up the cursor.

Figure 1–4 shows the relationship between the editor and the file you are inspecting.

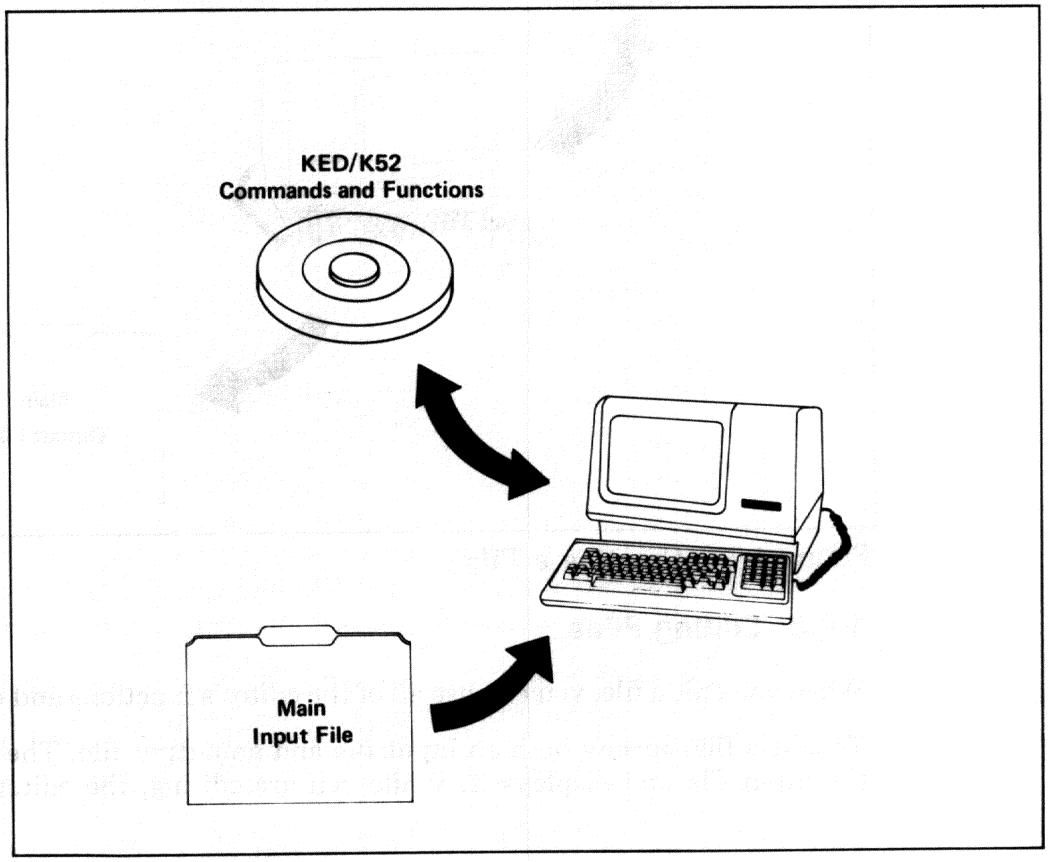


Figure 1–4: Inspecting a File

1.4.2 Creating Files

When you create a new file, you can use all of the editor's functions and commands.

To create a file, specify an output file but no input file. The editor initially displays a blank screen except for one special symbol, called the end-of-file symbol, and the cursor. With VT100 terminals, the editor uses the symbol ▒ as the end-of-file symbol, and on VT52s the editor uses the symbol ■. As you type the new material on the keyboard, the editor inserts each character at the cursor's position and shifts the end-of-file symbol to the right or to the next line. When you stop creating the new file with the EXIT command, the editor closes the output file.

Figure 1–5 shows the relationship between the editor and the file that you create.

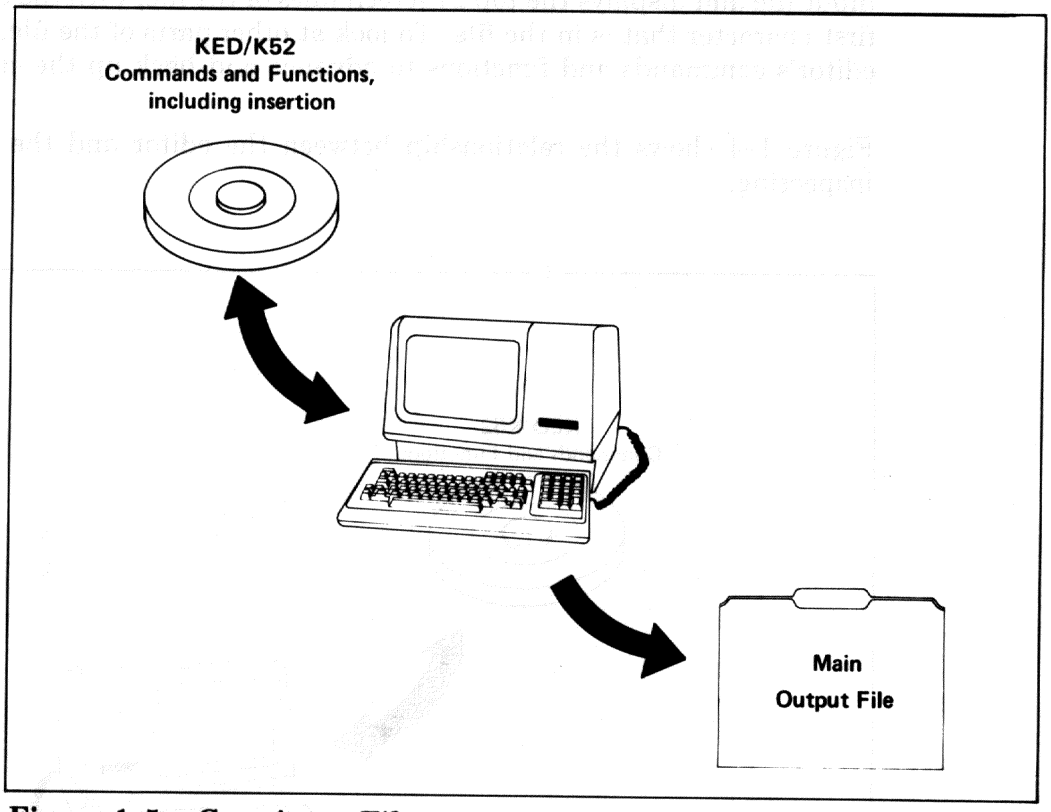


Figure 1–5: Creating a File

1.4.3 Editing Files

When you edit a file, you can use all of the editor's functions and commands.

To edit a file, specify both an input file and an output file. The editor opens the input file and displays it. While you are editing, the editor displays 24

screen lines of the input file, and, as you use the editor's commands and functions, the editor immediately shows the changes that you make. When you stop editing by issuing the EXIT command, the editor closes the output file. After the session, the output file contains your changes. Your original input file remains unchanged, in case you need to refer to it again.

Figure 1–6 shows the relationship between the editor and the main output and input files.

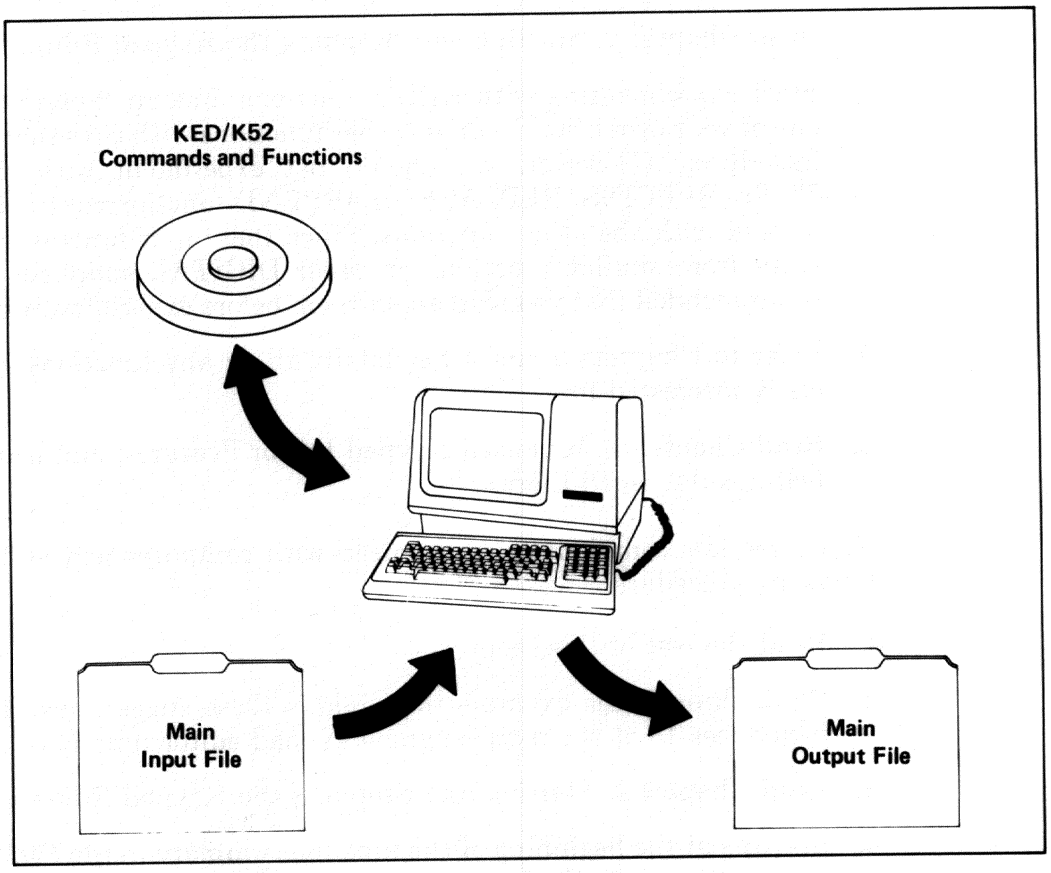


Figure 1–6: Editing a File

1.4.4 Using Auxiliary Files

The most common uses for the keypad editor involve only two files, the main input file and the main output file. In addition to the main files, you can also use an auxiliary input file and an auxiliary output file. Chapter 5 introduces auxiliary files, describes the commands that open and close them, and provides examples of when they are useful.

1.5 How to Learn to Use the Keypad Editor

The following procedures suggest two ways to learn to use KED. Because everyone seems to learn how computer text editors work in his or her own way, feel free to start experimenting in your own way whenever you like.

If you are skilled with any other text file editor that DIGITAL distributes, use the following procedure.

1. Read the rest of this section.
2. Work through the example that follows these suggestions.
3. Read Chapter 2, Starting and Stopping the Keypad Editor.
4. Start experimenting with KED's functions. Use an experimental copy of one of your own files. Compare other functions with ones that the example introduces. (However, you should not experiment with the SUBSTITUTE, SPECINS, REPLACE, or APPEND functions until after you have worked with the other functions. Since these four functions work differently from similar functions on other DIGITAL–supplied editors, it is recommended that you read about them before experimenting with them.)
5. Refer to Chapters 3 and 4 for details about any functions you are especially interested in.
6. Read Chapter 5, Advanced Keypad Editor Features, and learn to use the features described there.

If you are less confident or experienced with computer text editing, use the following procedure.

1. Read the rest of this section.
2. Work through the example that follows these suggestions. The example introduces 18 of the most common keypad editor functions.
3. Read Chapter 2, Starting and Stopping the Keypad Editor.
4. Starting at the beginning of the function summary in the *PDP–11 Keypad Editor Reference Card,* use a small ASCII file to experiment with each function. Although the descriptions on the *Reference Card* are short, they cover every common keypad editor feature.
5. Refer to full descriptions of functions in Chapters 3 and 4 whenever you think you need them.
6. After you have used the more common features of the keypad editor for a few days, read Chapter 5, Advanced Keypad Editor Features, and experiment with the features described there.

1.6 Special Terms and Concepts in this Manual

This short section presents the full definition of a keypad editor *word* and seven other special terms that appear frequently in this chapter.

You do not need to memorize these definitions, but you should read them and remember that the terms have special meanings to you as a keypad editor user.

CURSOR'S CHARACTER

The cursor's character is that character either underlined by the cursor, or on which the cursor is superimposed.

WORD

In this manual, *word* always means a keypad editor word, which is, in most cases, a string of printing characters and the spaces that follow the string.

The characters do not have to be letters of the alphabet; all digits and punctuation marks, such as periods (.), commas (,), and quotation marks (' and "), can be parts of words.

The Horizontal Tab character, a string of spaces that is flanked by tabs or line terminators, and each of the four line terminators that the keypad editor recognizes are words by themselves.

LINE TERMINATORS

A line terminator is a character or combination of characters that causes the keypad editor to start displaying subsequent characters on a new screen line. The keypad editor recognizes four line terminators:

1. Formfeed — When the keypad editor detects a Formfeed character it starts displaying characters at the left margin on the next screen line.
2. Vertical Tab — When the keypad editor detects a Vertical Tab character it starts displaying characters at the left margin on the next screen line.
3. New Line — This manual uses the term New Line for the most common line terminator in text files. The New Line terminator is generated by the Return key on the keyboard. In prompts, the keypad editor displays the terminator as the Carriage Return — Linefeed pair. The keypad editor functions that process individual characters always treat the New Line terminator as a single character. To clarify some examples in this manual, the symbol stands for the New Line terminator.

4. Linefeed by itself — When the keypad editor detects a Linefeed character that is not preceded by a Carriage Return, it also starts displaying characters on the next screen line. However, the screen line after a Linefeed by itself always starts directly below the Linefeed character, not at the left margin.

TEXT LINE

A keypad editor text line is composed of a line terminator and all of the characters between it and the preceding line terminator. Note that text lines and screen lines are not necessarily the same because a text line may be too long to fit on one screen line. However, even when a text line is too long for one screen line, the keypad editor still treats it as one text line.

WRAPPED LINE

When a text line is too long to fit on one screen line, the keypad editor uses as many screen lines as necessary to display the line. To call attention to the overlong line, the keypad editor precedes each extra screen line with a special symbol. This is called a *wrapped line* because the keypad editor automatically wraps the long text line into lower screen lines. The symbols ◆ and → are the KED and K52 wrap symbols, respectively.

TARGET

Most of the keypad editor's functions move the cursor forward or backward in your file. This manual frequently uses the term *target* for the location to which the keypad editor moves the cursor. Different functions move the cursor to different targets. For example, the target for the WORD function is the beginning of a word, and the target for the EOL function is a line terminator.

PAGE

A keypad editor *page* is a unit of text that you can define to suit your needs. Initially, a page ends with a Formfeed character and contains all of the characters back to the preceding Formfeed character. However, you can redefine a page by using the SET command to specify other targets for the PAGE function.

SECTION	A keypad editor *section* is another unit of text that you can define to suit your needs. Initially, a section is 16 consecutive text lines forward or backward from the cursor. However, you can also define a section by using the SET command to specify other targets for the SECTION function.

1.7 An Example of the Keypad Editor in Action

This example demonstrates some of the most common keypad editor functions. It has the following specific purposes.

- To show you what the screen looks like while you are editing.
- To show you some typical ways that the keypad editor changes your screen.
- To show you the keypad editor's cursor and end-of-file symbol.
- To introduce the special function keypad.
- To demonstrate that the names of the keypad editor functions explain most of what the functions do.

To use this example, you need a copy of the keypad diagram for the VT100 or the VT52 and the file SAMPLE.KED on the distribution volume. Figures 1–1 and 1–2 include both keypad diagrams as does the *PDP–11 Keypad Editor Reference Card*. You may find it useful to photocopy one of the diagrams and attach the copy to your terminal.

The first time you work with this example, follow each instruction carefully since each step depends on the preceding step. Therefore, do not experiment while working with the example for the first time.

Among other things, this example shows you how to change a word in the file to another word. The original file contains several occurrences of the word `//DATE//`. The example shows you how to erase each occurrence and insert the current date.

Each step in the example starts with an instruction. Read the instruction, then look at your screen while you type the keypad editor function the instruction asks for. Watch how the keypad editor executes the function. Finally, read the short description that follows the instruction in the example.

1. Install DIGITAL's keypad editor distribution volume in an available drive.
2. Copy the file SAMPLE.KED to one of your working volumes.
3. Invoke the keypad editor and specify SAMPLE.KED as the file you want to edit. Use one of the following procedures, or have an experienced colleague start the keypad editor for you.

For RT-11:

- Read Section 2.1.1.1 and use the appropriate procedure for your system.
- Type one of the following commands:

  for a VT52 terminal `EDIT/K52 SAMPLE.KED`RET
  for a VT100 terminal `EDIT/KED SAMPLE.KED`RET

For RSX-11M or RSX-11M-PLUS:

- Start the MCR (the prompt `MCR>` will appear on your screen).
- Type one of the following commands:

  for a VT52 terminal `K52 SAMPLE.KED`RET
  for a VT100 terminal `KED SAMPLE.KED`RET

4. Type your initials or your name. Use the keyboard keys exactly as if you were using a standard typewriter. Use the Delete key on the keyboard to correct typing mistakes.

   The keypad editor inserts each letter as you type it.

5. Use the end-of-line (EOL) function. Press the 2 key on the keypad.

   The keypad editor moves the cursor to the end of the first line.

6. Use the erase-left-word (LINEFEED) function. Press the Linefeed key on the keyboard.

   The keypad editor erases the word at the cursor's left. (The keypad editor also stores the characters you have erased, in case you want to recover them. A later step shows how to recover an erasure.)

7. Type the current date. Use the letter and number keys on your keyboard. Use the Delete key to correct typing mistakes.

   Again, the keypad editor displays each character as you type it.

8. Use the SECTION function. On a VT100, press the 8 key on the keypad. On a VT52, first press the blue keypad key and then the downarrow key.

   The keypad editor moves the cursor 16 lines forward. The keypad editor's default definition of a *section* is 16 text lines.

9. Use the WORD function. Press the 1 key on the keypad three times.

   For each WORD function, the keypad editor moves the cursor to the beginning of a *word*. *Word* and other keypad editor terms are defined in the section before this example.

10. Use the erase-line-right (DELLINE) function. On a VT100, press the PF4 key on the keypad. On a VT52, press the unlabeled grey key on the keypad.

    The keypad editor does four things.

    - It erases the cursor's character and the characters to the right of the cursor.
    - It also erases the line terminator.
    - It stores the string of characters (and the terminator).
    - It repaints your screen to show the effect of the erasure.

11. The preceding step erased too much. Therefore, use the undelete-line (UNDELLINE) function to recover the entire erasure. On a VT100, press the PF1 key, then the PF4 key on the keypad. On a VT52, press the blue key on the keypad, then the unlabeled grey key on the keypad.

    Most of the keys on the keypad control two keypad editor functions, a standard function and an alternate function. The functions shown on the top of the keys in the keypad diagram are standard functions. The functions shown on the bottoms of the keys are alternate functions.

    For example, on the VT100 PF4 key, DELLINE is the standard function. Unless you specify otherwise, that is the function the keypad editor will execute when you press the PF4 key. The alternate function of the PF4 key is to restore the last string you erased with DELLINE or one of the keypad editor's other line-erasure functions. To use the alternate function, you have to use the GOLD function as a prefix, and therefore press the PF1 key first.

    In this step, the UNDELLINE function restores the characters you just erased, including the line terminator, and the keypad editor shows the effect immediately.

12. Use the WORD function again. Press the 1 key on the keypad two more times.

    Again, the keypad editor moves the cursor two *words* forward.

13. Use the delete-to-end-of-line (DELEOL) function. Press the GOLD key and then the DELEOL key — on a VT100, the PF1 key, then the 2 key on the keypad. On a VT52, the blue key, then the 2 key on the keypad.

    The keypad editor erases the right-hand part of the line and stores the erasure. However, DELEOL does not erase the line terminator.

14. Use the BOTTOM function. Press the GOLD key and then the BOTTOM key — on a VT100, the PF1 key, then the 4 key on the keypad. On a VT52, the blue key, then the 4 key on the keypad.

    The keypad editor moves the cursor past all of the characters in the file. The keypad editor's symbol for the bottom of a file is ※ on a VT100 and ∎ on a VT52.

15. The keypad editor's searching functions move the cursor directly to the letters and strings you specify in the FIND function. You complete the FIND function with a model, and specify the direction of the search. The keypad editor moves the cursor to the target strings in the file that match your model.

    Use the FIND function. On a VT100, press the GOLD key and then the FIND key — the PF1 key, then the PF3 key on the keypad. On a VT52, press the blue key and the 8 key on the keypad.

    The keypad editor temporarily erases the top two screen lines and displays the prompt `Model:`.

    Type the string `//DATE//` — two slashes, the word `DATE`, and two slashes.

    The keypad editor displays each character of the model as you type it.

    Use the BACKUP function to specify a backward search. Press the BACKUP key — the 5 key on your keypad.

    The keypad editor searches backward through the file for the nearest occurrence of a string that matches your model. When the keypad editor finds a matching string, it places the cursor on the first character.

16. Use the erase-word-right (DELWORD) function. Press the DELWORD key — on a VT100, the hyphen (-) key on the keypad —, and then type the current date again. On a VT52, press the 9 key and then type the current date.

    The keypad editor erases the next word, stores the characters you erased, and displays each character you type.

17. Use the FINDNEXT function. Press the FINDNEXT key — on a VT100, the PF3 key on the keypad; on a VT52, the 8 key on the keypad.

    The keypad editor uses the latest model you provided, `//DATE//`, and searches backward for another occurrence of a matching string. (The keypad editor searches backward because BACKUP is the direction you specified most recently.) The cursor is now on one of three remaining occurrences of the string `//DATE//` in the file.

18. This step illustrates an easier way to change the remaining occurrences of `//DATE//` to the current date. The SUBSTITUTE function uses the keypad editor's paste buffer to make substitutions, and the first thing you have to do is put your substitution string in the paste buffer.

    Use the begin-selection (SELECT) function. Press the SELECT key — the period (.) key on the keypad.

    Type in the current date.

Use the cut-out (CUT) function. Press the CUT key — on a VT100, the 6 key on the keypad; on a VT52, the blue key and the 3 key on the keypad.

The SELECT function causes the keypad editor to mark the cursor's position as one end of a selected string. The screen does not show any effects of selecting until you move the cursor. On a VT100, as you type in characters, the keypad editor displays them in reverse video form.

The CUT function erases all of the characters that are in the current selection, and the keypad editor stores them in its paste buffer. The cursor is still on the first character of `//DATE//`.

19. Use the substitution (SUBSTITUTE) function. Press the GOLD key and then the SUBSTITUTE key — on a VT100, the PF1 key, then the Enter key on the keypad; on a VT52, the blue key and the Enter key on the keypad.

    The keypad editor substitutes the current date from the paste buffer for the search target and then executes the FINDNEXT function.

20. Use the SUBSTITUTE function again as described in the step above.

    Each time you use the SUBSTITUTE function, the keypad editor substitutes and then searches. In this step, the keypad editor made the substitution but failed to find another string that matched the model. The keypad editor signals the failure by ringing the bell on your terminal.

21. Use the HELP function. Press the HELP key — on a VT100, the PF2 key on the keypad; on a VT52, the red key on the keypad.

    When the SUBSTITUTE function failed, the keypad editor loaded a short explanation of the failure into a special buffer. HELP causes the keypad editor to temporarily erase the bottom three lines on your screen and display the explanation.

22. Use the restore-screen (CTRL/W) function. Type CTRL/W — hold the CTRL key down while you type the letter W on the keyboard.

    The keypad editor removes the HELP message and restores the lines it temporarily erased.

23. Use the EXIT command to finish this example and to save the new version of the sample file that you have produced. Press the GOLD key and then the COMMAND key — on a VT100, the PF1 key, then the 7 key on the keypad. On a VT52, the blue key and then the 7 key on the keypad. Type the word `EXIT` on the keyboard and terminate the command with the Enter key on the keypad.

    The keypad editor temporarily erases the top two screen lines and displays the prompt `Command:`. The keypad editor displays each letter of the command as you type it. When you press the Enter key, the keypad editor moves the cursor to the lower left corner of the screen, pauses briefly, and then displays its identifying message again.

24. Type CTRL/C or CTRL/Z — hold the CTRL key down while you type the letter C or Z on the keyboard.

    When the keypad editor is displaying its prompt on RT-11 systems, CTRL/C returns you to operating system level. On RSX-11M systems, CTRL/Z returns you to operating system level and CTRL/C returns you to MCR level.

Chapter 2
Starting and Stopping the Keypad Editor

2.1 Starting a Session

2.1.1 Setting the Operating System for a Video Terminal

This section describes the settings that each operating system requires for the keypad editor. For example, you may need to specify that your terminal is a video terminal, not a printing terminal. You may need to specify the screen width that you want the operating system to assume when it sends characters to your terminal. And you may need to specify how you will be setting the transmit and receive rates on your terminal.

For some systems, the system manager may be responsible for all of the settings that are described in this section. If you are not sure about the settings for your system, check your local operating procedures or ask your system manager.

2.1.1.1 Procedure for RT-11 — The RT-11 operating system must be set for your video terminal. The system settings that the keypad editor requires depend on the specific RT-11 monitor that you are using. Use the following procedures:

1. For a multi-terminal monitor, type the following command:
   - SET TT CONSOL=n [where n is the logical unit number of your terminal]
2. For a single-terminal single job monitor, type only the following command:
   - SET TT SCOPE

2.1.1.2 Procedure for RSX-11M and RSX-11M-PLUS — With the RSX-11M and RSX-11M-PLUS operating systems, the keypad editor does not require a specific set of operating system settings except that the terminal must be identified properly to the system as a VT52 or a VT100. Use the SET /VT52=TI: command to run K52 and the SET /VT100=TI: command to run KED. The keypad editor transmits characters to the terminal and receives characters from the terminal when the system has any valid combination of settings for a VT100, VT52, or VT55 terminal.

2.1.1.3 Procedure for IAS — With IAS, the keypad editor does not require a specific set of operating system settings. However, the editor does require you to identify the terminal to the system as a VT52 or VT100. The keypad editor transmits characters to the terminal and receives characters from the terminal when the system has any valid combination of settings for a VT100 or VT52. Identify VT55 terminals as VT52's.

DCL users can use the PDS commands SET TERMINAL VT52 before running K52, and SET TERMINAL VT100 before running KED.

MCR users can use the MCR commands TER /VT52 before running K52 and TER /VT100 before running KED.

2.1.2 Setting VT100 Terminal Characteristics

To set your VT100 terminal characteristics, use the VT100 SETUP A and SETUP B modes. Refer to your *VT100 User Guide* for detailed instructions about the SETUP modes.

The keypad editor accepts any combination of settings that is valid for your operating system. With KED, you can also change your terminal screen width and the video background while you are editing. Section 3.3.2 describes the KED commands for changing the width and background.

You can choose either the block cursor ▒ or the underline cursor (—). The block cursor is easier to see, and the examples in this manual show the block cursor.

NOTE

With RSX–11M, RSX–11M–PLUS, and IAS systems, you can run KED with your terminal set for either the VT52 or the ANSI modes. However, when you exit from KED, your terminal will always be set to the ANSI mode.

2.2 Running the Keypad Editor on RT–11 Systems

You can run the keypad editor on RT–11 V5 systems in three ways:

1. By using the RT–11 interactive command EDIT with the command option /KED or /K52.
2. By using the RT–11 SET EDIT KED or SET EDIT K52 commands and then the EDIT command without the corresponding option.
3. By using the RT–11 interactive command R or RUN and then typing a Command String Interpreter (CSI) file specification line.

On earlier versions of RT–11 (V3 and V3B), you can use only the third method to run the keypad editor. The earlier versions do not support the /KED or /K52 options for the EDIT command or the KED or K52 arguments for the SET EDIT command.

When you use the RUN or R command, the keypad editor returns you to the CSI level and displays the CSI prompt (*). The special settings that you have made and the contents of the keypad editor's buffers are preserved for your next keypad editor session, and you can start another keypad editor session by completing another CSI file specification string. Or you can return to the RT-11 monitor level by typing CTRL/C. The keypad editor settings and buffer contents are lost only when you respond to the CSI prompt by typing CTRL/C.

The following sections describe how to use both the EDIT command and the RUN or R command. If you are using RT–11 V4 or V5, read both sections. However, if you are using only an earlier version of RT–11, skip to Section 2.2.5.

2.2.1 EDIT, R and RUN Command Default Values Under RT–11

Table 2–1 summarizes the command default values for the keypad editor running under RT–11.

Table 2–1: EDIT, R, and RUN Command Default Values under RT–11

Item	Default
Text editor	RT–11's default text editor is EDIT, the program that is described in Chapter 6 of the *RT–11 System User's Guide*. Section 2.2.3 describes how you can specify the keypad editor as the default text editor for your own versions of the RT–11 system.
KED & K52 volume	For the EDIT and R commands, the RT-11 system volume, SY:. For the RUN command, the default storage volume, DK:.
Editing process	Unless you specify a different option, KED and K52 automatically back up the file you specify on the same volume that contains the input file.
Input volume	The RT-11 default storage volume, DK:.
Output volume	The RT-11 default storage volume, DK:.
Input file name & type	No default; must be specified.
Output file name & type	For the /CREATE and /OUTPUT options, must be specified. For editing with automatic backup, KED and K52 use the original file name and type.
Maximum output file size	Depends on the number of contiguous free blocks on the output volume. KED and K52 allocate the larger of the following two sizes: • One-half of the largest number of contiguous free blocks. • All of the second-largest number of contiguous free blocks.

2.2.2 Using the RT–11 Interactive Command EDIT

The RT–11 interactive command EDIT invokes the text editors that DIGITAL distributes for use with RT–11. Chapter 4 in the *RT–11 System User's Guide* describes the EDIT command in detail and the following sections summarize how to use the command to start the keypad editor.

Version 5 of RT–11 provides two EDIT command options, /KED and /K52, for the two versions of the keypad editor. With the options, the full EDIT command syntax is as follows:

```
EDIT [{/KED} ] {[/CREATE] (SP) input-filespec[/ALLOCATE:size]                    }
     [{/K52} ] {/INSPECT (SP) input-filespec                                     }
               {/OUTPUT:output-filespec[/ALLOCATE:size] (SP) input-filespec       }
```

Use standard RT–11 file specifications as described in Chapters 3 and 4 of the *RT–11 System User's Guide*. The /CREATE, /INSPECT, /OUTPUT, and /ALLOCATE options are described in the following sections.

2.2.3 Specifying the Keypad Editor As Your Default Editor Under RT–11

If you want the keypad editor to be the default text editor that RT–11 starts when you use the EDIT command, use the SET EDIT command. The forms of the command for KED and K52 are as follows:

```
SET EDIT KED (RET)
SET EDIT K52 (RET)
```

You can also include the SET EDIT KED or SET EDIT K52 command in your startup indirect command file. If you do so, the indirect command file will specify KED or K52 as your default text editor each time you boot your RT–11 system volume.

When KED or K52 is the default text editor, you do not need to add the corresponding option (/KED or /K52) to the EDIT command when you want to start the keypad editor. However, you will need to use the /EDIT option if you want to use the standard RT–11 text editor EDIT.

2.2.4 Using Options with the RT–11 EDIT Command

This section describes how to use the options with the EDIT command to specify the following editing processes with KED and K52:

1. Editing files with automatic backup
2. Inspecting files
3. Creating new files
4. Editing files and using new output file names
5. Specifying the maximum output file size

2.2.4.1 Editing Files with Automatic Backup — The keypad editor's default editing process automatically creates a backup file. When you finish editing a file, the keypad editor renames your input file with the .BAK file type and stores the new file that you produce under the input file name and type. For example, if you begin an editing session with the following command, at the end of the session KED stores the original version of REPORT.TXT under the name REPORT.BAK and the new version of the file under the name REPORT.TXT.

```
EDIT/KED REPORT.TXT RET
```

2.2.4.2 Inspecting Files — When you add the /INSPECT option to the EDIT command, the keypad editor displays the original file that you specify, but the keypad editor's functions and commands that modify a file are disabled. You can use only the commands and functions that move the cursor and control auxiliary output files. Therefore, although you cannot make any changes to a file that you are inspecting, you can create auxiliary output files. Except for auxiliary output files, the keypad editor does not erase, create, or rename any files when you finish inspecting a file. For example, at the end of an editing session that you begin with the following command, the file INSPEX.PRG is unchanged.

```
EDIT/KED/INSPECT INSPEX.PRG RET
```

2.2.4.3 Creating New Files — When you add the /CREATE option to the EDIT command, the keypad editor creates a new file when you finish editing. For example, at the end of an editing session that you begin with the following command, KED stores a new file under the name PR26.DAT.

```
EDIT/KED/CREATE PR26.DAT RET
```

2.2.4.4 Editing Files and Using New Output File Names — When you add the /OUTPUT option to the EDIT command and specify an output file name, the keypad editor opens the output file and displays the original file that you specify. All keypad editor commands and functions are enabled. When you finish editing, the keypad editor closes the output file and leaves your original file unchanged. For example, at the end of an editing session that you begin with the following command, KED stores the changes you have made in a new file named SPCOUT.FOR and leaves the file GENIN.FOR unchanged.

```
EDIT/KED/OUTPUT:SPCOUT.FOR GENIN.FOR RET
```

2.2.4.5 Specifying the Maximum Output File Size — When you add the /ALLOCATE option to the EDIT command, the keypad editor uses the number you specify as the maximum size in blocks for the output file. For example, each of the following commands specifies 196 blocks for the output file.

```
EDIT/KED SPRINT.SPC/ALLOCATE:196 RET
EDIT/KED/CREATE WINTER.SPC/ALLOCATE:196 RET
EDIT/KED/OUTPUT:FALL.SPC/ALLOCATE:196 SUMMER.SPC RET
```

The maximum input file size for editing with the keypad editor is 16383 blocks. Therefore, 16383 blocks is also the largest output file size that you can allocate.

2.2.5 Using the RT–11 Interactive Commands R and RUN

Start any keypad editor session with the RT–11 RUN or R command.[1] If KED.SAV and K52.SAV are on your system volume, you do not need to specify a volume in the command. For example, with KED.SAV on volume SY: and with that volume installed in drive DX0:, the following commands are equivalent.

```
R KED RET                          [SY: is the default volume.]
R SY:KED RET
```

If you have stored KED.SAV and K52.SAV on another volume, you must type the device specification for the volume explicitly in the RUN or R command. The following examples show two explicit device specifications.

```
RUN RK0:KED RET
RUN DY1:KED RET
```

2.2.5.1 The RT–11 CSI File Specification String — When you have completed the RUN or R command, the keypad editor uses the RT–11 Command String Interpreter (CSI)[2] to prompt you for a file specification string. The CSI prompt is an asterisk (*). The form of the file specification string you type determines whether you are inspecting or editing a file.

2.2.5.2 Editing Files with Automatic Backup — The CSI strings for editing files with automatic backup can be in two forms:

1. *output-file*[/A:*size*]=*original-file* RET
2. *original-file*[/A:*size*] RET

You can add the /A:*size* option to any output file specification. Complete the option in the same way as for the EDIT command /ALLOCATE option, as described in Section 2.2.4.5.

In the first form, the specifications for the output file and original file must be the same if you want the keypad editor to use the automatic backup process.

For example, at the end of an editing session that you begin with the following commands, KED stores the original version of EXAMP.FOR under the name EXAMP.BAK and the new version of the file under the name EXAMP.FOR.

```
.R KED RET
*RK1:EXAMP.FOR=RK1:EXAMP.FOR RET
```

---

[1] Chapter 4 in the *RT–11 System User's Guide,* Keyboard Commands, documents the RUN command in detail.

[2] Chapter 1 in the *RT–11 System Utilities Manual* includes details on the Command String Interpreter.

The following example has the same effect as the preceding example.

Typing only the specification for the input file that you want to edit is the shortest way to specify the keypad editor's automatic backup process.

```
.R KED (RET)
*RK1:EXAMP.FOR (RET)
```

2.2.5.3 Creating New Files — The CSI string for creating a file is a single file specification followed by the /C option and terminated with the Return key.

If another file with the same specification exists, the keypad editor responds with a warning. You can cancel the session; or you can continue the session, erase the existing file, and create a new file under the old name.

Examples: The following example creates a new file on the default storage device DK:.

```
.R KED (RET)
*EXAMP.BAS/C (RET)
```

[DK: is the default CSI device. KED shows an empty screen with the cursor at the upper left corner.]

The following example creates a new file on DECpack drive RK1:. K52 warns that another file with the same name exists. It will be erased because the user responds Y (for *yes*) to the warning.

```
.R K52 (RET)
*RK1:CURRENT.DAT/C (RET)
?KED-W-Output file exists, Continue (Y,N) ?Y (RET)
```

[Y means that the existing file RK1:CURRNT.DAT will be erased. K52 shows an empty screen with the cursor at the upper left corner.]

The following example creates a new file on diskette drive DX0:. KED warns that another file with the same name exists. It is preserved because the user responds N (for *no*) to the warning.

```
.R KED (RET)
*DX0:ARGLO.FOR/C (RET)
?KED-W-Output file exists, Continue (Y,N) ?N (RET)
```

[N means that the existing file DX0:ARGLO.FOR will not be erased. KED identifies itself and displays the CSI prompt.]

2.2.5.4 Inspecting Files — The CSI string to inspect a file that exists is a single file name followed by the /I option terminated with the Return key. Note that the default storage device, DK:, is the default device for all CSI file specifications.[3]

---

[3] Chapter 3 in the *RT–11 System User's Guide*, System Conventions, includes details on permanent device names, and Chapter 4, Keyboard Commands, covers the ASSIGN and DEASSIGN commands and ways you can assign the logical name DK: to different physical devices.

Examples: The following example shows the complete procedure for starting KED to inspect a file on the default storage device DK:.

```
.                                    [RT-11 monitor prompt]
.R KED (RET)                         [SY: is the default volume.]
*RPT296.TXT/I (RET)                  [DK: is CSI's default device. KED
                                     displays the top 24 screen lines of
                                     DK:RPT296.TXT.]
```

The following example shows complete procedures to start K52 to inspect a file that is on DECpack drive RK1:.

```
.                                    [RT-11 monitor prompt]
.R K52 (RET)
*RK1:PROG1.BAS/I (RET)               [K52 displays the top 24 screen lines
                                     of RK1:PROG1.BAS.]
```

2.2.5.5 Editing Files and Using New Output File Names — The CSI string for editing files and using new output file names is in the following form:

output-file[/A:*size*]=*original-file* (RET)

The specifications for the output file and original file must be different if you want the keypad editor to store the edited version of the file under a new file name. The /A:*size* option is used in the same way as the EDIT command /ALLOCATE option described in Section 2.2.4.5.

For example, at the end of a session that you begin with the following commands, KED stores the edited version of TEMP.DAT under the name EXPT44.DAT.

```
.R KED (RET)
*EXPT44.DAT=TEMP.DAT (RET)
```

2.2.6 Restrictions for RT-11 Systems

This section describes the restrictions for running the keypad editor on RT-11 operating systems.

2.2.6.1 File Protection — Except for editing with automatic backup as described in Section 2.2.5, the keypad editor displays the following message when an output file specification is the same as for a file that already exists on the output volume:

```
?KED-W-Output file exists, Continue (Y,N)?
```

When the message appears, you can cancel the editing session by typing CTRL/C or N (for *no*). Or you can continue the session, erase the existing file, and create a new file under the old name by typing Y (for *yes*).

In the following example, KED warns that diskette DX1: already contains a file named MAYQUE.BAS. If the user stops the keypad editor with the EXIT command, the file will be erased because the user responds *yes* to the warning.

```
.EDIT/KED/OUTPUT:DX1:MAYQUE.BAS DX0:PG451.BAS RET
?KED-W-Output file exists. Continue (Y,N)? RET
```

However, if the user stops the keypad editor with the QUIT command, the old file will not be disturbed.

RT-11 does not support file version numbers as the RSX family of operating systems do.

2.2.6.2 Output File Size — Under RT-11, the keypad editor searches the output volume for contiguous free space in which to store a file that you are creating. Then the keypad editor allocates the free space for your editing session. In the following three cases, the keypad editor displays messages about the free space it has allocated. Appendix A provides guidelines for you to follow when the messages appear.

1. When the output volume does not contain any contiguous free space that is as large as your original file, KED and K52 both display the following message and return immediately to the RT-11 monitor or the CSI prompt:

   ```
   ?KED-F-Output file shorter than input file
   ```

2. When the largest amount of free space on the output volume is less than 10 blocks larger than your original input file, KED and K52 both display the following message:

   ```
   ?KED-W-Only MMMMMM blocks available for insertions
   - Continue (Y,N)?
   ```

   The keypad editor also displays the message when you want to create a new file and the output volume contains less than 10 blocks of free space.

3. When you add material to a file that you are creating or editing and the file becomes as large as the free space allocated for it, the keypad editor refuses to accept more material and signals you each time you try to insert more. You can use the HELP function to see an explanation of the signal. For example, if a file you are creating reaches its maximum size while you are typing on the keyboard, the keypad editor signals the error by ringing the bell on your terminal or reversing the video background of your terminal's screen. If you then use the HELP function, the keypad editor displays the following message:

   ```
   Insert finds file full
   ```

2.2.6.3 Accidentally Using the Wrong Kind of Terminal — The keypad editor is designed only for VT52s, VT55s, and the VT100 series of video terminals. If you accidentally try to run the keypad editor from a hard-copy terminal, such as the LA36 DECwriter, try the following procedures to recover control of your system:

- Type CTRL/C several times if the EDIT command prompt `FILE?` or CSI prompt (*) has been displayed. Continue working normally when the RT-11 monitor prompt (.) appears.
- Re-boot your system if you:
  1. Typed the EDIT/KED or EDIT/K52 command and included a complete file specification.
  2. Responded to the CSI prompt with a file specification.

If you accidentally run KED from a VT52 terminal, repeat the following procedure until the RT-11 monitor prompt (.) appears:

1. Type CTRL/C several times to return to the RT-11 monitor level. In most cases, the monitor prompt will not be displayed in this step, however.
2. Turn your VT52 off, then on.
3. Press the Return key. In most cases, the monitor prompt appears after this step.

If you have included the SET EDIT KED or SET EDIT K52 command in your start-up indirect command file, the EDIT command by itself will cause one of the problems described above if your system console is the wrong kind of terminal.

2.3 Running the Keypad Editor on RSX-11M Systems

You can run the keypad editor on RSX-11M V3.2 systems and RSX-11M-PLUS V1.0 systems. For RSX-11M systems, the system must be built with the full-duplex terminal driver during the system generation procedure.

If you are uncertain about how your RSX-11M system has been built, ask your system manager, and if you are responsible for building systems yourself, refer to the installation instructions in the *RSX-11M System Generation Manual.*

For RSX-11M-PLUS systems, the full-duplex terminal driver is the only terminal driver available. You can, therefore, use the keypad editor with any RSX-11M-PLUS system.

This section presents the details about running the keypad editor that apply to all RSX-11M systems as well as to RSX-11M-PLUS systems when you are using the MCR Command Line Interpreter (CLI). With RSX-11M-PLUS systems you can also use the DIGITAL Command Language Interpreter (DCL) EDIT command as described in Section 2.4.

You can run the keypad editor on RSX-11M V3.2 and RSX-11M-PLUS V1 systems in two ways:

1. By calling the keypad editor directly with the task names KED or K52 if the keypad editor has been installed as an RSX-11M system utility.

2. By using the RSX-11M Monitor Console Routine (MCR) command RUN, and specifying the keypad editor.

When you call the keypad editor directly with the task names KED or K52 and include a file specification string in the call, the keypad editor returns you to the MCR level when you finish an editing session. Any special settings that you made while using the keypad editor and anything that you have stored in the keypad editor's buffers, such as the paste buffer, are lost.

When you run the keypad editor with the RUN command, the keypad editor returns you to the keypad editor file specification string level and displays the prompt KED> before returning you to the MCR level. The special settings that you made and the contents of the keypad editor's buffers are preserved for your next keypad editor session. When the prompt KED> is displayed, you can start another keypad editor session by completing another keypad editor file specification string, or you can return to the MCR level by typing CTRL/Z. The settings and buffer contents are lost only when you respond to the prompt KED> by typing CTRL/Z.

With the keypad editor installed in your system, you can also call the keypad editor directly with the task names KED or K52 without including a file specification string. This case is the same as for the RUN command.

The following sections describe the two most common methods of running the keypad editor. The descriptions assume that the system library device LB: contains the keypad editor or that the keypad editor is one of the installed utilities in your RSX-11M system. Refer to the *RSX-11M/M-PLUS MCR Operations Manual* for full descriptions of the RUN command and MCR procedures. The *MCR Operations Manual* also describes how to use the MCR INS command to install utilities such as the keypad editor.

2.3.1 Start-up Default Values Under RSX–11M

Table 2–2 summarizes the command default values for the keypad editor running under RSX–11M.

Table 2–2: Start-up Default Values Under RSX–11M

Item	Default
Input & output UIC	The LOGON UIC. Use the SET /UIC command or the /UIC option for the RUN command to specify a different default.
Input & output volumes	The volume installed in the default device SY:. Use the ASN command to assign the logical device name SY: to different physical devices.
Input file name & type	No default; must be specified.
Output file name & type	When creating a new file from an existing file, must be specified. When creating a new version of a file, the keypad editor uses the input file name and type.
Input file version	The latest version of the input file that is on the input volume.
Output file version	When creating a new file, the keypad editor assigns version 1. Otherwise, the keypad editor assigns a version number that is one plus the version number of the latest version that is on the output volume.
Output file protection	When creating a new file, the default file protection. When editing a file, the keypad editor sets the output file protection to match the input file protection. (The *RSX–11 Utilities Manual* describes how to change the file protection.)
Output file carriage control attribute	When creating a new file, the keypad editor assigns the implied carriage control attribute. Otherwise, the keypad editor assigns the input file's attribute.
Size of the temporary file	The size of the temporary file limits the amount by which you can increase the size of the input file in one keypad editor session. When you create an entirely new file, the default size of the temporary file is 50 blocks. Otherwise, the default size is 50 blocks larger than the input file.

2.3.2 Calling the Keypad Editor with the Task Names KED and K52

With the keypad editor installed in your system, you can call the keypad editor directly by typing the task name. The two forms of the call are:

1. >KED *file-specification-string[/options]* RET
   >K52 *file-specification-string[/options]* RET

2. >KED RET
   >K52 RET

When you use the first form, the file specification string you type specifies one of the four following editing processes and the keypad editor starts immediately.

1. Storing a new version of a file and using the original file name
2. Inspecting a file
3. Creating a new file
4. Storing a new version of a file and using a new output file name

The different file specification strings and options are described in the following sections.

When you use the second form of the direct call with the task names KED and K52, the keypad editor displays the prompt KED> or K52>. Respond to the prompt by typing one of the file specification strings described in the following sections and add the option that you want the keypad editor to use.

NOTE

RSX-11M systems automatically use the input file with the highest version number and store the output file with the next higher version number.

Although the system and the keypad editor accept an explicit output file version number, by using one you may accidentally reduce the security of your files. For example, if you specify an output file version number that is lower than the version number of your input file, the highest version number no longer defines your latest file.

2.3.2.1 Storing a New Version of a File and Using the Original File Name — The general form of the file specification string to use when you want to store a new version of a file and use the original file name is as follows:

input-filespec [{ /CA / -CA / FO }] [/BL:*temporary-file-size*] RET

For the file specifications, use any of the forms that are valid for RSX-11M. The options are described in Section 2.3.2.5.

Examples: At the end of an editing session that you begin with the following command, KED stores a new version of the file named EXAMP.FTN. The new version includes all of the changes made to the latest version of EXAMP.FTN under your default UIC that KED finds on the default system device.

```
MCR>KED EXAMP.FTN RET
```

[KED displays the first 24 lines of the latest version of SY:EXAMP.FTN under your default UIC.]

2.3.2.2 Inspecting a File — The general form of the file specification string to use when you want to inspect a file is as follows:

input-filespec/IN

Use any valid RSX-11M file specification. The /IN option (for INSPECT) is required.

Examples: When you begin an editing session with the following command, KED immediately displays the first 24 lines of version 4 of the file SY:INSPEX.PRG. No changes can be made to the file because the /IN option disables all the keypad editor functions and commands that insert or erase material.

```
MCR>KED INSPEX.PRG;4/IN RET
```

2.3.2.3 Creating an Entirely New File — The general form of the file specification string to use when you want to create an entirely new file and the options that the keypad editor accepts are as follows:

output-filespec/CR [{/CA, /-CA, /FO}] [/BL:*temporary-file-size*] RET

Use any valid RSX-11M file specification. The /CR option (for CREATE) is required. The other options are described in Table 2-3.

Examples: At the end of an editing session that you begin with the following command, KED stores version 1 of a new file named EXAMP.BAS on the default device SY:.

```
MCR>KED EXAMP.BAS/CR RET
```

[SY: is the default output device. KED immediately displays the cursor in the upper-left corner of an empty screen.]

2.3.2.4 Storing a New Version of a File and Using a New Output File Name — The general form of the file specification string to use when you want to store a new version of a file and use a new output file name is as follows:

output-filespec [{/CA, /-CA, /FO}] [/BL:*temporary-file-size*]=*input-filespec* RET

Use any valid RSX-11M file specifications. The options are described in Table 2-3.

Examples: Assume that the default system device does not contain a file named SPCOUT.FTN. At the end of an editing session that you begin with the following commands, KED stores version 1 of a new file named SPCOUT.FTN on the default system device. The file SPCOUT.FTN;1 includes all of the changes made to the latest version of GENIN.FTN that KED finds on the default system device.

```
MCR>KED SPCOUT.FTN=GENIN.FTN RET
```

2.3.2.5 Keypad Editor Command Line Options — Table 2–3 lists the options and their meanings that you can use when you use the keypad editor.

Table 2–3: Options for Keypad Editor Command Lines

Option	Meaning
/BL:*temporary-file-size*	Use the /BL option (for BLOCKS) and specify the number of blocks that you want the keypad editor to use for the temporary file. To specify a decimal file size, add a decimal point to the number. To specify an octal file size, omit the decimal point. The keypad editor's use of the temporary file is explained in Section 2.3.3.2. Since the default temporary file size is usually adequate, you need to use the /BL option only in special cases, such as when you want to combine several files.
/CA	Create the output file with the implied carriage control attribute. This is the default option when you create an entirely new file.
/-CA	Create the output file with the embedded carriage control attribute.
/FO	Create the output file with the FORTRAN carriage control attribute. However, the keypad editor does not check that the first character in each record is a valid FORTRAN carriage control character.
/ID	Use the /ID option (for IDENTIFY) to have the keypad editor display its version number. Do not add the option to any file specification string. The option is valid only by itself in response to the KED> and K52> prompts and in the following forms of the direct call with the task name: >KED/ID RET >K52/ID RET

2.3.3 Restrictions for RSX–11M Systems

This section describes the restrictions for running the keypad editor on RSX–11M systems.

2.3.3.1 File, Account, and Volume Protection — On RSX–11M systems, the keypad editor supports the standard system conventions such as read, write, and update access to files, the characteristics of privileged and non-privileged accounts, and public and private devices. The *RSX–11M/M–PLUS MCR Operations Manual* describes these conventions in detail.

The keypad editor also warns you in the following cases:

1. When you specify an output file name and version number that already have been stored on the output volume, KED and K52 both display the following message:

```
?KED-W-Output file exists - Continue (Y,N)?
```

2. When you specify an output file that is open for another user or if the input and output file specifications are the same, KED and K52 both display the following message:

```
?KED-F-Unable to create output file
```

When the first message appears, you can cancel the editing session by typing N (for *no*). Or you can continue the session, erase the existing file, and store the new file that you create under the old name by typing Y (for *yes*).

After the keypad editor displays the second message, no files are created or changed. Following the message, the system displays the KED> prompt.

In the following example, the keypad editor warns that disk DM1: already contains the file MAYQUE.BAS;17. It will be erased because the user responds Y to the warning.

```
MCR>KED DM1:MAYQUE.BAS;17=MAYQUE.BAS <RET>
?KED-W-Output file exists - Continue (Y,N)? Y <RET>
```

2.3.3.2 Output File and Temporary File Size — On RSX-11M systems, the keypad editor allocates free space on the output volume for two purposes:

1. For a temporary file that the keypad editor uses until you finish the editing session.
2. For the final output file when you finish the editing session.

The following sections describe the restrictions and requirements that apply to the temporary file and final file. The most common problem occurs when the output volume contains too little free space for the file that the keypad editor creates. If you need more free space, you can:

- Specify an output volume that has more free space than SY:.
- Create more free space on the SY: volume by purging some of the files that are stored there.
- Specify a smaller temporary file size by using the /BL option.

The keypad editor allows you to move the cursor backward in the file that you are editing, inspecting, or creating. However, for this feature the keypad editor requires a temporary file, and when the keypad editor starts, it allocates

free space on the output volume for the temporary file. When you are creating an entirely new file, the temporary file initially is empty. When you are editing or inspecting a file, the keypad editor copies your RSX–11M input file into the temporary file as you move the cursor down through the file. As you back the cursor up through the file, the keypad editor uses the copy until you finish editing.

When you finish the editing session, the keypad editor copies the temporary file to the RSX–11M output file that you specified. If the keypad editor cannot complete the copying process, the system locks both the output file and the temporary file and displays the following message:

```
?KED-F-Unable to copy temporary file
```

When the keypad editor finishes copying, it truncates the output file after the last record and deletes the temporary file. If the keypad editor cannot close the output file properly, the system locks both the output file and the temporary file and displays the following message:

```
?KED-F-Unable to truncate or close output file
```

If either message appears, the locked files contain questionable data. If you must try to salvage part of either locked file, use the PIP UNLOCK command as described in the *RSX–11 Utilities Manual.*

The keypad editor allocates free space on the output volume for the temporary file and output file in one of the following ways:

1. When you are inspecting a file, the keypad editor allocates space for the temporary file in the default UIC and on the default system device SY: but does not allocate any space for an output file. The free space that the keypad editor requires is the same as the size of your input file. If the space available for the temporary file is too small for the input file, the keypad editor displays one of the following messages, and the system displays the KED> or K52> prompt:

   ```
   ?KED-F-Temporary file shorter than input file
   ?KED-F-Unable to create temporary file
   ```

   The messages are explained in Appendix A.

2. When you are creating a new file, KED normally allocates 50 blocks for the temporary file and 55 blocks for the output file. When you add the /BL option to your file specification string, the keypad editor allocates the amount of space you specify for the temporary file and 10% more for the output file. Both spaces are allocated on the output volume. When your output volume contains too little free space, the keypad editor displays the following message or one of the messages listed above, and the system displays the KED> or K52> prompt.

   ```
   ?KED-F-Unable to create output file
   ```

3. When you are editing a file, the keypad editor normally allocates for the temporary file 50 blocks more than the size of your input file. When you add the /BL option to your file specification string, the keypad editor allocates the amount of space you specify for the temporary file. With and without the /BL option, the keypad editor allocates for the output file 10% more than the size of the temporary file. Both spaces are allocated on the output volume. When your output volume contains too little free space, the keypad editor displays one of the messages described earlier in this section.

When you start the keypad editor, if the temporary file is not at least 10 blocks larger than the input file, KED and K52 both display the following message:

```
?KED-W-Only mmmmmm blocks available for insertions - Continue
(Y,N)?
```

If you will only be erasing material from your input file or adding a small amount of new material, you can continue the editing session by typing Y in response to the message. Otherwise, you can stop the session immediately by typing N.

When you add material to a file that you are creating or editing and the file becomes as large as the free space allocated for it, the keypad editor refuses to accept more material and signals you each time you try to insert more. For example, if a file you are creating reaches its maximum size while you are typing on the keyboard, the keypad editor signals the error by ringing the bell on your terminal or reversing the video background of your terminal's screen. If you then use the HELP function, the keypad editor displays the following message:

```
Insert finds file full
```

2.3.3.3 Output File Record Length — In the output file, the keypad editor creates a record for each text line in your file. You can create a text line of any length by typing or using other methods of inserting material. However, when the keypad editor copies your temporary file to your output file, the maximum output file record length the keypad editor creates is 256 bytes, not counting the record's line terminator. Therefore, if your temporary file includes a text line that is longer than 256 bytes, the keypad editor writes more than one record to the output file. In this case, the keypad editor writes 256-byte output records, adding the Carriage Return — Linefeed terminator, until 256 bytes or fewer remain in the temporary file record. The last output file record that the keypad editor writes has the terminator that the editor displays.

NOTE

The keypad editor does not warn you in this case.

2.3.3.4 Input File Record Length — When you are editing or inspecting a file, the size of the keypad editor's input buffer determines the maximum length of input file records that the keypad editor can read. The following section describes how you can increase the size of the internal input buffer when you start the keypad editor. Whenever an input file contains a record that is too long, KED and K52 both display the following message immediately after starting:

```
?KED-W-Input file contains records that will be truncated -
Continue (Y,N)?
```

When the message appears, you can cancel the editing session by typing N. Or you can continue the session by typing Y. If you continue, the keypad editor truncates each input file record that is too long but does not warn you in any way.

2.3.4 Running the Keypad Editor with Larger Paste and Input Buffers

On your RSX-11M system, the standard size of the keypad editor's internal input buffer and paste buffer is determined when KED and K52 are task built. The minimum size for each is 512 bytes, but you can install the keypad editor with a larger task increment.

You can increase the size of both the internal input buffer and the paste buffer for a single editing session by starting the keypad editor with the MCR command RUN, adding the /INC option (for INCREASE) to the command line, and specifying the increase in size that you need. The general forms of the RUN command in this case are as follows:

```
RUN $KED/INC=size (RET)
RUN $K52/INC=size (RET)
```

Table 2–4 shows the number of words you need to specify with the /INC option to increase both the internal input buffer and the paste buffer by increments of 512 bytes. Intermediate increases for the paste buffer are also possible. For example, when you specify 4400. words with the /INC option, the input buffer is increased by 512. bytes as shown in the table, and the paste buffer is increased by 924. bytes (4400.-3988.=412.+512.=924.).

Table 2–4: Values for the /INC Option

Desired increase in input and paste buffer size	Value to specify with the /INC option
512. bytes	3988. words
1024. bytes	4756. words
Each additional 512. bytes	Add 768. words

2.4 Running the Keypad Editor on RSX-11M-PLUS Systems

To run the keypad editor on RSX-11M-PLUS systems, use the same procedures as for RSX-11M. The keypad editor accepts all valid RSX-11M-PLUS file specifications.

In addition to the RUN command and the direct calls with the task names KED and K52, you can run KED and K52 with the RSX-11M-PLUS EDIT command. Complete the EDIT command with the /USING qualifier and add the task name KED or K52. For example, to use KED to inspect a file named STUDY.TST that is on the default system device, type the following EDIT command:

```
>EDIT/USING:KED STUDY.TST/IN RET
```

2.5 Stopping the Keypad Editor

To stop a keypad editor session, use the COMMAND function and the EXIT or QUIT command. You can stop a session while any part of a file is on your screen. You do not need to move the cursor to any special locations in a file to stop.

NOTE

Refer to a copy of the keypad diagram for the VT100 or VT52 when you are learning the following commands and functions. See Figure 1-1 and 1-2 for the keypad diagrams.

2.5.1 The COMMAND Function

VT100 Keys: The PF1 key, then the 7 key on the keypad.
VT52 Keys: The blue key, then the 7 key on the keypad.

To use any keypad editor command, press the GOLD key and then the COMMAND key. When you use the COMMAND function, the keypad editor temporarily erases the top two lines of your screen and displays the prompt `Command:`.

The keypad editor accepts several different commands. (Most are presented later in this section.) To complete a command:.

- Type the command name in full or use an abbreviation.
- Terminate the command with the ENTER function (press the Enter key on the keypad).

The following sections show complete procedures for using the EXIT and QUIT commands to stop the keypad editor.

2.5.2 The EXIT Command

The full form of the EXIT command is as follows. The keypad editor does not accept any abbreviation.

```
EXIT
```

When you use the EXIT command to stop an editing session, the keypad editor completes the following steps.

1. The keypad editor closes all files.
2. The keypad editor creates or renames files, if necessary.
3. If you have been editing or creating a file, the keypad editor saves the results of your work, including an open auxiliary output file.
4. The keypad editor then displays the prompt you see when you start any editing session. Therefore, you can immediately start another session by typing another file specification string.

2.5.3 The QUIT Command

The full form of the QUIT command is as follows. The keypad editor does not accept any abbreviation.

```
QUIT
```

When you use the QUIT command to stop an editing session, the keypad editor completes the following steps.

1. The keypad editor closes all input files without changing then in any way.
2. The keypad editor purges your main output file (and auxiliary output file, if one is open). "Purge" means that the temporary output file and the main output file are not preserved. The keypad editor displays the message

   ```
   ?KED-W-Output files purged.
   ```

3. The keypad editor then displays the same prompt that you see when you start an editing session. Therefore, you can immediately start another session by typing another file specification string.

Chapter 3
Inspecting a File

This chapter describes the basic functions that you may use while inspecting a file. While you are reading, refer to the keypad diagrams for the VT100 and VT52 in Figures 1–1 and 1–2. Auxiliary file functions, which may be used when you are inspecting a file, are described in Chapter 5.

3.1 Using the Standard and Alternate Functions

Each keypad key controls a standard function and an alternate function. To use an alternate function, use the GOLD function first and then press the keypad or keyboard key that controls the alternate function that you want to use. For example, the alternate function for the 4 key is BOTTOM. To use the BOTTOM function, press the GOLD and BOTTOM keys — on the VT100 keypad the PF1 key and then the 4 key, on the VT52 keypad the blue key and then the 4 key.

3.1.1 The GOLD Function

VT100 Key: The PF1 key on the keypad.
VT52 Key: The blue key on the keypad.

The GOLD function by itself has no effect on the file. When you press the GOLD key, the keypad editor processes the alternate function of the next keypad or keyboard key that you press.

The GOLD function applies only to the single function that follows. For example, to move the cursor first to the top of the file and then to the bottom, you must press the following sequence of keys (the TOP and BOTTOM functions are described later in this chapter):

GOLD TOP GOLD BOTTOM

A sequence of consecutive GOLD functions is equivalent to a single GOLD function. For example, the following sequences both execute the BOTTOM function once:

GOLD BOTTOM
GOLD GOLD GOLD BOTTOM

You can use the GOLD function at any time except when the `Command:` or `Model:` prompt is displayed. (The `Model:` prompt is for searching and is explained later in this chapter.)

3.1.2 The RESET Function

VT100 Keys: The PF1 key and the period (.) key on the keypad.
VT52 Keys: The blue key and the period (.) key on the keypad.

The RESET function has no effect on your file. However, the function is useful in two cases:

- When you have pressed the GOLD key and then decide to use a standard function instead of an alternate function.
- When you have built a select range and then decide to cancel it (Section 4.3 describes select ranges and ways to use them).

For example, the following sequences both execute the WORD function once (the ADVANCE, BACKUP, and WORD functions are described later in this chapter):

ADVANCE GOLD RESET WORD BACKUP
ADVANCE WORD BACKUP

The following sequence creates a select range with two text lines and then cancels the select range:

BLINE SELECT BLINE BLINE GOLD RESET

You can use the RESET function at any time except when a prompt is displayed.

3.2 Getting Help: The HELP Function

VT100 Key: The PF2 key on the keypad.
VT52 Key: The red key on the keypad.

When a keypad editor process fails, the keypad editor signals you in one of the following ways:

1. The keypad editor's normal signal is to ring the bell on your terminal. For K52, this is the only signal.

2. With a VT100, you can also set KED to signal quietly by using the SET QUIET command (the command is described later in this section).

In each case, when the keypad editor signals you, it loads a one-line explanation of the signal in an internal message buffer. When you use the HELP function, the editor temporarily erases the bottom three screen lines and displays the explanation.

For example, because STOP is not a keypad editor command, the following sequence causes the editor to signal that a message is available:

(COMMAND) STOP (ENTER)

When you use the HELP function after that sequence, the keypad editor displays the following message:

```
Illegal command
```

The HELP function can also display three special forms that summarize the keypad editor features.

The first form displayed is a screen version of the keypad diagram. KED displays the VT100 keypad diagram and K52 displays the VT52 keypad diagram.

If you use the HELP function while the keypad editor is displaying a keypad diagram, the keypad editor displays the second form which is a summary of all keypad editor commands. In this help form, KED shows the valid abbreviation for each command in reverse video or by underlining, depending on the form of the cursor that you are using. K52 shows only the full forms of the commands.

If you use the HELP function again without an intervening function or command, the keypad editor displays a summary of the GOLD keyboard commands.

While the keypad editor is displaying only a part of your file, use the HELP function as follows to display a keypad editor message or either of the help forms:

1. If there is a keypad editor message in the message buffer:
   - The first HELP function displays the message.
   - The second HELP function displays the keypad diagram.
   - The third HELP function displays the command summary.
   - The fourth HELP function displays the GOLD keyboard command summary.
   - Subsequent HELP functions alternately display the diagram and command summary.

2. If there is not a keypad editor message in the message buffer:

   - The first HELP function displays the keypad diagram.
   - The second HELP function displays the command summary.
   - The third HELP function displays the GOLD keyboard command summary.
   - Subsequent HELP functions alternately display the diagram and command summary.

When a message or help form is displayed, you can make the keypad editor restore all lines temporarily erased from your file by using the ENTER or CTRL/W function. You can also use any other valid keypad editor function. In this case, the keypad editor completes the function before repainting the screen.

You can use the HELP function at any time except when a keypad editor prompt is displayed.

3.3 Using the VT100 Commands

Because the VT100 has several video features that the VT52 does not have, such as reverse video display and 132-column screen width, KED provides six more commands than K52. The commands allow you to adjust your VT100 display while you are using KED.

The SET QUIET and SET NOQUIET commands control how KED signals you when a command or function fails.

The four SET SCREEN commands set the screen width to 80 characters or 132 characters and set the display background to dark or light.

3.3.1 Setting KED's Message Signal

If you work in a quiet area, you can use the SET QUIET command to set KED to signal you quietly rather than with the VT100 terminal bell. You can use the SET NOQUIET command to restore the bell signal whenever you wish.

3.3.1.1 The SET QUIET Command — The form of the SET QUIET command is:

```
SET QUIET
```

The SET QUIET command sets KED to signal you without sounding the terminal "bell" about commands and functions that fail.

When a command or function fails, KED temporarily reverses the background of your screen to indicate that a function failed. To see an explanation of the signal, use the HELP function. To continue editing, use ENTER or any other valid function. KED then restores the normal background and executes the command or function that you specified.

If you use VT100 SETUP commands to control the screen display background, the SET QUIET command causes the screen to be set to DARK when a function fails. However, if you use the keypad editor SET DARK or SET LIGHT commands, a function failure will reverse the background.

The SET QUIET command is illegal with K52.

3.3.1.2 The SET NOQUIET Command — The general form of the SET NOQUIET command is:

```
SET NOQUIET
```

The SET NOQUIET command sets KED to signal you about commands and functions that fail by ringing the VT100 terminal bell. This is KED's default signal. To see an explanation of the signal, use the HELP function. To continue editing, use the ENTER or CTRL/W functions, or use any other valid function.

The SET NOQUIET command is illegal with K52.

3.3.2 Setting the VT100 Screen Width and Background

As noted earlier in this manual, you can set your VT100 screen width and background before you start KED by using the VT100's SETUP A and SETUP B features. The SETUP features are described in your *VT100 User Guide*. For your convenience, KED also provides two sets of commands that you can use to adjust your VT100 screen while you are editing.

The SET SCREEN 80 and SET SCREEN 132 commands set the VT100 screen width.

The SET SCREEN LIGHT and SET SCREEN DARK commands set the VT100 screen background.

The default settings are the settings that are in effect when you start KED.

3.3.2.1 The SET SCREEN 80 Command — The general form of the SET SCREEN 80 command is:

```
SET[ SCREEN] 80
```

After the SET SCREEN 80 command, KED sets the VT100 terminal for 80 characters per screen line. In order to display all line terminators correctly, the maximum number of printing characters on a screen line in this case is 78.

The SET SCREEN 80 command is illegal with K52.

3.3.2.2 The SET SCREEN 132 Command — The general form of the SET SCREEN 132 command is:

```
SET[ SCREEN] 132
```

After the SET SCREEN 132 command, KED sets the VT100 terminal for 132 characters per screen line. In order to display all line terminators correctly, the maximum number of printing characters on a screen line in this case is 130.

The SET SCREEN 132 command is illegal with K52.

3.3.2.3 The SET SCREEN DARK Command — The general form of the SET SCREEN DARK command is:

```
SET[ SCREEN] DARK
```

After the SET SCREEN DARK command, KED sets the VT100 terminal to display all printing characters as light characters on a dark background. The non-printing characters, such as spaces and New Line terminators, are not displayed.

The SET SCREEN DARK command is illegal with K52.

3.3.2.4 The SET SCREEN LIGHT Command — The general form of the SET SCREEN LIGHT command is:

```
SET[ SCREEN] LIGHT
```

After the SET SCREEN LIGHT command, KED sets the VT100 terminal to display all printing characters as dark characters on a light background. The term *reverse video* refers to this display method. The non-printing characters such as spaces and New Line terminators are not displayed.

The SET SCREEN LIGHT command is illegal with K52.

3.4 Terminating Commands and Finishing with HELP: The ENTER Function

VT100 and VT52 Keys: The Enter key on the keypad.

The ENTER function has the following effects:

1. When you are entering a command, the ENTER function terminates the command string and causes the keypad editor to begin executing the command.
2. When you are displaying a keypad editor HELP message or any form of the HELP function, ENTER terminates the HELP sequence and re-displays your file on the screen.

The ENTER function is invalid only when the `Model:` prompt is displayed.

3.5 Verifying that the Display Is Up-To-Date: The CTRL/W Function

VT100 and VT52 Keys: Hold down the CTRL key while you type the letter W on the keyboard.

The CTRL/W function repaints each character of your file that should be in your display and thus verifies that your display illustrates exactly what is in your file. Four cases in which the CTRL/W function is especially useful are as follows:

1. When the keypad editor is displaying a message or the keypad diagram after you have used the HELP function.
2. When the operating system has forced a system message to your terminal while you are using the keypad editor. Most messages of this type are written over part of the text line that contains the keypad editor's cursor.
3. To reset the keypad editor to application mode, that is, to cause the keypad to generate the escape sequences used by the keypad editor instead of the numbers generated when not using the keypad editor.
4. If you accidently reset your terminal, you can restore your file by typing CTRL/W. You can use the CTRL/W function at any time except when the `Model:` or `Command:` prompt is displayed.

3.6 Cancelling the Keypad Editor's Process: The CTRL/C Function

VT100 and VT52 Keys: Hold down the CTRL key while you type the letter C on the keyboard.

The CTRL/C function cancels any process that the keypad editor is executing. After you use the CTRL/C function, the keypad editor immediately stops what it is executing and displays the part of your file where the cursor is located. The keypad editor then signals you that a process has been cancelled. If you then use the HELP function, the keypad editor displays the following message:

```
CTRL/C entered to stop operation
```

The CTRL/C function is especially useful when:

1. The keypad editor is executing a long process that you want to cancel. For example, in a large file, the keypad editor may require several seconds to complete the BOTTOM function. While the editor is executing the function, you can cancel it by using the CTRL/C function.
2. You accidentally use the wrong function. For example, if you accidentally search backward instead of forward, you can cancel the search by using the CTRL/C function.

3. You change your mind about creating or copying an auxiliary file. For example, if you enter the command to copy 50 pages from a long auxiliary input file, you can cancel the process by using the CTRL/C function.

4. You enter a misspelled or incorrectly typed character string as a model for a FIND function. You can cancel the FIND operation by typing CTRL/C.

The CTRL/C function has no effect on the part of a keypad editor process that is complete. The process simply stops and the keypad editor displays the cursor wherever it has moved.

If you use the CTRL/C function while you are entering a command or a search model, the keypad editor cancels the COMMAND or FIND function immediately and signals that it has done so. If you then use the HELP function, the keypad editor displays one of the following messages:

```
Command canceled
Model prompt canceled
```

If the keypad editor is not executing any process when you use the CTRL/C function, the keypad editor signals that it is ignoring the function. If you then use the HELP function, the keypad editor displays the following message:

```
CTRL/C or CTRL/Z ignored - use QUIT
```

The message is worded in that way because CTRL/C and CTRL/Z are the most common ways to stop many of the other programs and utilities that your operating system provides. However, the keypad editor, protecting you from the bad effects of accidentally typing CTRL/C, requires that you use the QUIT command to stop editing.

You can use the CTRL/C function at any time, but it has no effect unless the keypad editor is executing a process.

3.7 The Arrow Functions

The arrow functions are controlled by the VT52's arrow keys on the keypad and the VT100's arrow keys at the upper right corner of the keyboard. The four arrow functions always move the cursor as directly as possible in the directions of the arrows on the keys.

The rightarrow function moves the cursor to the right and from a line terminator to the beginning of the next text line.

The downarrow function moves the cursor straight down.

The leftarrow function moves the cursor to the left and from the beginning of a line to the preceding line terminator.

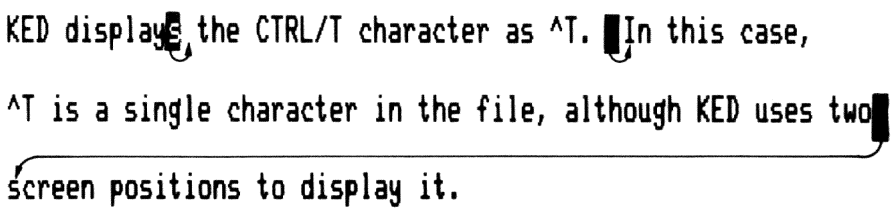
The uparrow function moves the cursor straight up.

3.7.1 The Rightarrow Function

VT100 Key: The rightarrow key on the keyboard.
VT52 Key: The rightarrow key on the keypad.

The rightarrow function moves the cursor to the character that follows the current character. When the cursor moves from a line terminator, the keypad editor generally scrolls the file upward.

Example: The following example shows how the rightarrow function moves the cursor in three cases.



The rightarrow function is invalid when:

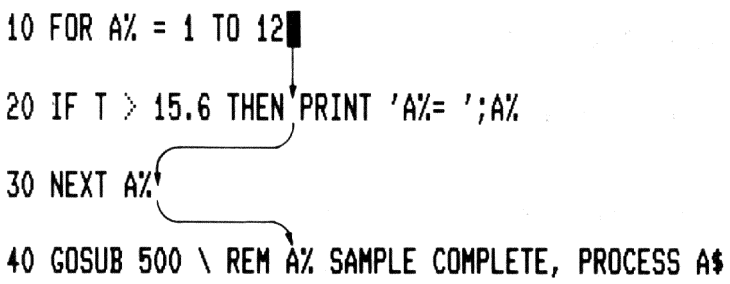
1. The cursor is on the end-of-file symbol.
2. A prompt is on the screen.

3.7.2 The Downarrow Function

VT100 Key: The downarrow key on the keyboard.
VT52 Key: The downarrow key on the keypad.

The downarrow function moves the cursor to the following text line. If a character is directly under the current character, the cursor moves to it. Otherwise, the cursor moves left within the target line to the nearest character.[1] The keypad editor generally scrolls the file upward.

Example: The following example shows how three consecutive downarrow functions move the cursor.



When you use consecutive downarrow functions, the cursor may stop to the left of its initial screen column in some text lines. The reason is as follows. The downarrow function always moves the cursor to a real character in your file. Each space is a real character, but the blank columns that follow line terminators and Horizontal Tab characters are not really spaces that are in your file. Therefore, whenever the downarrow function moves the cursor to the right of a line terminator, the keypad editor places the cursor on the terminator. In most cases, when the cursor moves into a blank that follows a Horizontal Tab character, the keypad editor places the cursor on the tab.

The downarrow function is invalid when:

1. The cursor is on the end-of-file symbol.
2. A prompt is on the screen.

[1] However, when the downarrow function moves the cursor into a text line that follows a single Linefeed character, the cursor moves to the right within the text line because the blank at the beginning of the text line does not represent a real character in your file.

3.7.3 The Leftarrow Function

VT100 Key: The leftarrow key on the keyboard.
VT52 Key: The leftarrow key on the keypad.

The leftarrow function moves the cursor to the character preceding the current character. When the cursor moves to a line terminator, the keypad editor generally scrolls the file downward.

Example: The following example shows how the leftarrow function moves the cursor in three cases.

```
KED moves the cursor with the arrow functions, searching functions
and functions that detect units of text, such as 'words'.  The
arrow functions are most useful for small cursor adjustments.
```

The leftarrow function is invalid when:

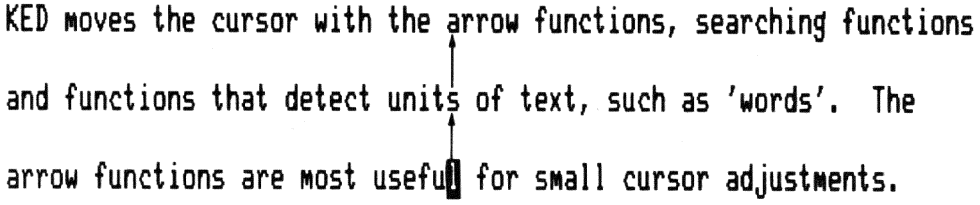
1. The cursor is at the top of the file.
2. A prompt is on the screen.

3.7.4 The Uparrow Function

VT100 Key: The uparrow key on the keyboard.
VT52 Key: The uparrow key on the keypad.

The uparrow function moves the cursor to the preceding text line. If a character is directly above the current character, the cursor moves to it. Otherwise, the cursor moves left to the nearest character in the target text line.[1] The keypad editor generally scrolls the file downward.

Example: The following example shows how two consecutive uparrow functions move the cursor.



When you use consecutive uparrow functions, the cursor may stop to the left of its initial screen column in some lines. The reason is as follows. The uparrow function always moves the cursor to a real character in your file. Each space (ASCII octal 040) is a real character, but the blank columns that follow line terminators and Horizontal Tab characters are not really spaces that are in your file. Whenever the uparrow function moves the cursor to the right of a line terminator, the keypad editor places the cursor on the terminator. In most cases, when the cursor moves into a blank that follows a Horizontal Tab character, the keypad editor places the cursor on the tab.

You cannot use the uparrow function when a prompt is on the screen.

---

[1] However, when the uparrow function moves the cursor into a text line that follows a single Linefeed character, the cursor moves to the right within the text line because the blank at the beginning of the text line does not represent any real characters in your file.

3.8 Controlling the Direction the Cursor Moves

The keypad editor has two directional modes: backup and advance. As a result, you can instruct the editor to perform many functions in either direction. In the backup mode, the keypad editor moves the cursor to the left and upward, that is, toward the beginning of the file. In the advance mode, the keypad editor moves the cursor to the right and downward, that is, towards the bottom of the file. The ADVANCE and BACKUP functions set the directional mode.

The arrow functions are not affected by the directional mode. They always move the cursor in the directions indicated by the arrows on the keys.

3.8.1 The ADVANCE Function

VT100 and VT52 Key: The 4 key on the keypad.

The ADVANCE function sets the keypad editor to the advance mode. After you press the ADVANCE key, the cursor movement functions advance the cursor toward targets that are to the right of the cursor and below it.

For an example of the ADVANCE function, see the examples for the WORD function and the BLINE function below.

You cannot use the ADVANCE function when the prompt `Command:` is displayed. When you are completing a model for the FIND function, the ADVANCE function terminates the model and directs the keypad editor to search toward the bottom of your file for a target that matches the model.

3.8.2 The BACKUP Function

VT100 and VT52 Key: The 5 key on the keypad.

The BACKUP function sets the keypad editor to the backup mode. After you press the BACKUP key, the cursor movement functions back the cursor up toward targets that are to the left of the cursor and above it.

For an example of the BACKUP function, see the examples for the WORD function and the BLINE function below.

You cannot use the BACKUP function when the prompt `Command:` is displayed. When you are completing a model for the FIND function, the BACKUP function terminates the model and directs the keypad editor to search toward the top of your file for a target that matches the model.

3.9 Moving the Cursor by Characters, Words, and Lines

The keypad editor processes files in terms of five units of text.

- Characters.
- Words.
- Text lines.
- Sections (discussed in Section 3.11).
- Pages (also discussed in Section 3.11).

Characters include the letters of the alphabet, digits, and all other printing and non-printing characters in the ASCII character set.

Words are strings of printing characters and the spaces that follow them, in most cases. However, the keypad editor treats the Horizontal Tab character and each of the four line terminators that the keypad editor recognizes as a word by itself.

Text lines are strings of printing and non-printing characters that end with one of the four line terminators the keypad editor recognizes:

1. The Carriage Return — Linefeed pair (the New Line terminator).
2. The Vertical Tab character.
3. The Formfeed character.
4. A Linefeed character (without a preceding Carriage Return character).

This chapter distinguishes text lines from screen lines because the keypad editor uses as many screen lines as necessary for displaying each text line that is in your file. However, the functions that process text lines, such as the BLINE function, generally have the same effects regardless of the number of screen lines that a text line requires.

Sections and pages can be defined to allow you to move the cursor by using groups of lines or by searching for certain characters.

3.9.1 The CHAR Function

VT100 Key: The 3 key on the keypad.

K52 does not include a CHAR function because the arrow functions are controlled by keypad keys on the VT52. Therefore, with K52 use the leftarrow and rightarrow functions.

Depending on directional mode, the CHAR function advances or backs up the cursor one character. When the cursor moves to a different line, the keypad editor generally scrolls the file in the opposite direction.

Example: The following example shows how the CHAR function advances and backs up the cursor in three cases.

```
Leftarrow and rightarrow are equivalent to the sequences
BACKUP CHAR and ADVANCE CHAR.
```

Note that the keypad editor displays the Horizontal Tab character (ASCII octal 011) as a blank that starts at the tab character's real position and ends at the next normal hardware tab stop (columns 1, 9, 17, etc.). Therefore, a blank that is caused by a tab character looks the same as a blank that is caused by an equivalent number of spaces (ASCII octal 040). Usually the file's context will distinguish between the two cases, but you can also use the CHAR function to distinguish them. In most cases when a tab character is present, the CHAR function makes the cursor skip more than one print position backward to a tab character and forward from a tab character.

The CHAR function is invalid:

1. In the backup mode when the cursor is at the top of the file.
2. In the advance mode when the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

3.9.2 The WORD Function

VT100 and VT52 Key: The 1 key on the keypad.

Depending on directional mode, the WORD function advances or backs up the cursor to the first character of a word. When the cursor's character is the first character of a word, the cursor advances to the first character of the next word or backs up to the first character of the preceding word. When the cursor moves to a different line, the keypad editor generally scrolls the file in the opposite direction.

Example: The following example shows how the WORD function advances and backs up the cursor in three cases. In each case, the cursor moves to a word boundary.

```
Most punctuation characters and spaces are parts of words. A
space is never the first character in a word except when
it is the first character:

    1. In a file.
    2. In a text line.
    3. Following a Tab character.

To KED, each of the following strings is a word:

    AB347T,      THAT,      DK3:REM.TMP
```

The WORD function is invalid:

1. In the backup mode when the cursor is at the top of the file.
2. In the advance mode when the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

3.9.3 The BLINE Function

VT100 and VT52 Key: The 0 key on the keypad.

Depending on directional mode, the BLINE function advances or backs up the cursor to the first character of a text line. When the cursor's character is the first character of a text line, the cursor advances to the first character of the next text line or backs up to the first character of the preceding text line. When the cursor moves to a different line, the keypad editor generally scrolls the file in the opposite direction.

Example: The following example shows how three consecutive BLINE functions advance and back up the cursor. In either direction, the cursor moves to the character that follows a line terminator.

```
100 REM Read and store up to 10 ASCII lines
110 OPEN "DK0:SAMPLE.TXT" FOR INPUT AS FILE #4
120 FOR C% = 1 TO 10
130 LINPUT A$(C%)
140 IF END #4 GO TO 500
150 NEXT C%
500 STOP
```

The BLINE function is invalid:

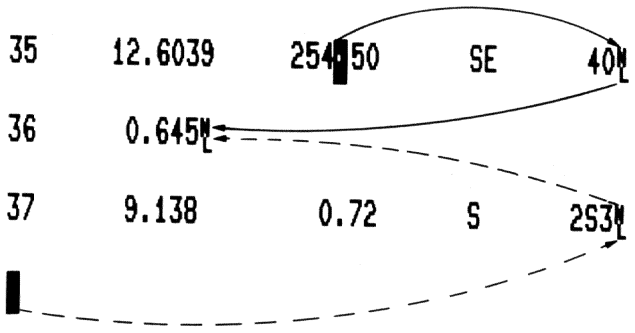
1. In the backup mode when the cursor is at the top of the file.
2. In the advance mode when the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

3.9.4 The EOL Function

VT100 and VT52 Key: The 2 key on the keypad.

Depending on the directional mode, the EOL function advances or backs up the cursor to the nearest line terminator. When the cursor starts on a line terminator, the EOL function advances the cursor to the next line terminator or backs it up to the preceding line terminator. When the cursor moves to a different text line, the keypad editor generally scrolls the file in the opposite direction.

Example: The following example shows how the EOL function advances and backs up the cursor. The ¶ symbol is shown only to clarify that the target for the EOL function is a line terminator (the keypad editor does not display the symbol).



The EOL function is invalid:

1. In the backup mode when the cursor is at the top of the file.
2. In the advance mode when the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

3.10 Moving the Cursor to the Top and Bottom of a File

From any location in the file that you are editing, you can move the cursor directly to the bottom of the file or to the top of the file.

Moving the cursor to the bottom is useful when you want to extend the file. For example, when you want to complete a memo that you started in an earlier keypad editor session, you can use the following procedure:

1. Start the keypad editor.
2. Specify the name of the file that contains the incomplete memo.
3. When the keypad editor displays the first lines of the file, use the BOTTOM function to move the cursor directly to the bottom.
4. Insert the material that you want to add.

Moving the cursor to the top is useful when you want to add material before the first character in the file and when you want to start other editing processes from the top of the file.

3.10.1 The BOTTOM Function

VT100 Keys: The PF1 key and the 4 key on the keypad.
VT52 Keys: The blue key and the 4 key on the keypad.

The BOTTOM function moves the cursor directly to the end-of-file symbol. When the cursor's original position is far above the bottom of the file, the keypad editor may temporarily erase the top two screen lines and display the message `WORKING...` . When the keypad editor completes the function, the keypad editor displays the cursor on the end-of-file symbol.

The BOTTOM function is invalid when:

1. The cursor is on the end-of-file symbol.
2. A prompt is displayed.

3.10.2 The TOP Function

VT100 Keys: The PF1 key and the 5 key on the keypad.
VT52 Keys: The blue key and the 5 key on the keypad.

The TOP function moves the cursor directly to the first character in the file. When the cursor's original position is far below the top of the file, the keypad editor may temporarily erase the top two screen lines and display the message `WORKING...` . When the keypad editor completes the function, the keypad editor displays the cursor in column 1 of the first screen line.

The TOP function is invalid when:

1. The cursor is at the top of the file.
2. A prompt is displayed.

3.11 Moving the Cursor by Pages and Sections

Defining pages and sections of text permits easy access to and handling of your file. You can define these units in one of the following ways:

- by line-count
- by marker-strings

The default definitions are a single Formfeed character for a page and 16 text lines for a section.

3.11.1 Using Line-Count Definitions

To define a page or section by line-count, specify the number of text lines that you want the keypad editor to jump over when you use the PAGE or SECTION function.

For example, if you define a section as 24 lines, the keypad editor advances or backs up the cursor by 24 lines each time you use the SECTION function. In this case, the SECTION function has the same effect as 24 consecutive BLINE functions.

3.11.2 Using Marker-String Definitions

To define a page or section by a marker-string, specify the string of characters that exist as text in your file which you wish to use to indicate the beginning of pages or sections of your file. Each time that you use the PAGE or SECTION function, the keypad editor searches for the marker-string that you specified.

For example, if you define a section with the string `SECT`, the keypad editor advances or backs up the cursor to the nearest occurrence of that string each time you use the SECTION function. Note that this will find occurrences of `PSECT` and `CSECT` in your file and will use them as SECTION marker-strings. In this case, the SECTION function has the same effect as the FIND function in searching for the marker-string.

3.11.3 Setting the Page Definition: The SET ENTITY PAGE Command

The general forms of the command to set the page definition are:

```
SET [ENTITY ]PAGE "marker-string"
SET [ENTITY ]PAGE 'marker-string'
SET [ENTITY ]PAGE integer[ LINES]
```

The first two forms, which are equivalent, set a marker-string definition. The marker-string must be enclosed within matching single quotes (') or double quotes ("). The marker-string can be up to 53 characters long and can contain any printing characters or line terminators.

The third form sets a line-count definition. The integer can be from 1 to 32767.

3.11.4 Setting the Section Definition: The SET ENTITY SECTION Command

The general forms of the command to set the section definition are:

```
SET [ENTITY ]SECTION "marker-string"
SET [ENTITY ]SECTION 'marker-string'
SET [ENTITY ]SECTION integer[ LINES]
```

The first two forms set a marker-string definition and are equivalent. The marker-string must be enclosed within matching single quotes (') or double quotes ("). The marker-string can be up to 51 characters long and can contain any printing characters or line terminators.

The third form sets a line-count definition. The integer can be from 1 to 32767.

3.11.5 Other Settings that Affect Page and Section Usage

When you define a page or section in terms of a marker-string, the keypad editor uses a searching procedure to locate it in your file (later sections describe the searching functions). The current setting for search matching (exact or general) determines the way the keypad editor processes pages or sections.

For example, if you define a page with the string `Page` and the keypad editor is set for general matches during searching, the keypad editor ignores all case distinctions in the marker-string and detects `Page`, `PAGE`, and `page` as the same marker-string.

If `Page` is your marker-string and the keypad editor is set to search for exact matches, the keypad editor detects `Page` as the only legitimate marker-string.

3.11.6 The PAGE Function

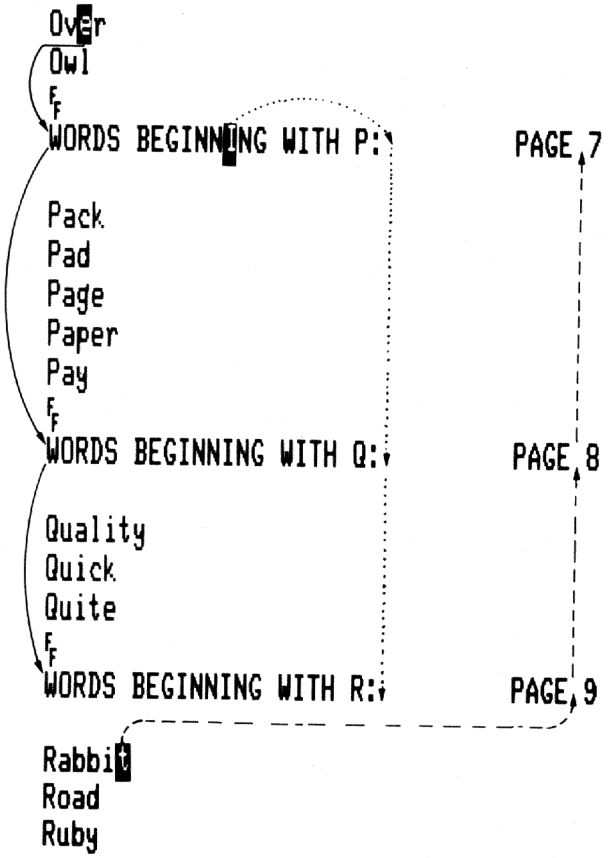
VT100 and VT52 Key: The 7 key on the keypad.

When a page is defined by a marker-string, the PAGE function advances or backs up the cursor to the first character that follows the marker-string. The default definition of a page is in terms of a single Formfeed character as the marker-string.

When a page is defined by line-count, the PAGE function advances or backs up the cursor by the equivalent number of BLINE functions.

Example: The following example shows how three consecutive PAGE functions advance and back up the cursor in three cases:

1. Advancing, with the default definition of a page (a single Formfeed character as shown by the symbol $^{F}_{F}$ on the VT100 and on the VT52).
2. Advancing, with a page defined as one colon (:).
3. Backing up, with a page defined as the string PAGE and with exact search matching. (In this case, the keypad editor skips the string Page near the middle of the example.)



You can use the SET ENTITY PAGE command to change the definition of a page at any time while you are using the keypad editor.

The PAGE function is invalid:

1. In the backup mode when the cursor is at the top of the file.
2. In the advance mode when the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

3.11.7 The SECTION Function

VT100 Key: The 8 key on the keypad.
VT52 Keys: The blue key and the downarrow key on the keypad.

The only significant differences between the PAGE and SECTION functions are that they use different keypad keys and that they have different default definitions. The default definition of a section is 16 consecutive text lines.

3.12 Searching for Strings in a File

The most flexible way to move the cursor through a file is by searching for strings of characters and moving the cursor to them. The PAGE and SECTION functions use searching to locate and display parts of a file in terms of the page and section marker-strings that you have specified. The FIND and FINDNEXT functions are more convenient for moving the cursor directly to strings whose locations you do not know.

The FIND function searches for a new string. The keypad editor asks for a model of the string you are looking for and advances or backs up the cursor until it finds a target string that matches the model that you have specified. Use the DELETE and CTRL/U functions to correct typing errors in the model.

The FINDNEXT function searches for the next occurrence of a string in the file that matches the current search model. You can change the keypad editor's directional mode between FINDNEXT functions to reverse the direction of search.

3.12.1 Commands to Change the Way the Keypad Editor Searches

You can use the SET SEARCH command to change three features of the keypad editor's searching functions. You can specify whether the keypad editor:

- Searches for targets that match the current model exactly or generally.
- Places the cursor on the first character of a target that matches the model or to the right of the last character.
- Limits a search to a page (as currently defined) or is allowed to extend the search to the top or bottom of the file.

3.12.1.1 The SET SEARCH GENERAL Command — The general form of the command to specify that the model and target match generally is:

```
SET [SEARCH ]GENERAL
```

After you enter the command, the keypad editor ignores the differences between uppercase letters and lowercase letters when you use the FIND or FINDNEXT function. General searching is the keypad editor's default setting. For a general search, an uppercase letter or a lowercase letter in the model requires only the same letter in the same position in the target string.

The command has no effect on the way the keypad editor processes nonalphabetic characters. For example, for the model `Ten`, several target strings match the model generally: `Ten`, `TEN`, `ten`, and `tEn` are four of the general matches.

3.12.1.2 The SET SEARCH EXACT Command — The general form of the command to specify that the model and target match exactly is:

```
SET [SEARCH ]EXACT
```

After you enter the command, the keypad editor depends on the differences between uppercase letters and lowercase letters when you use the FIND or FINDNEXT function. For an exact search, each uppercase letter (the set A — Z) in the model requires an uppercase letter in the same position in the target string. The lowercase letters must also match exactly.

The command has no effect on the way the keypad editor processes nonalphabetic characters. For example, for the model `Ten`, the only targets that match the model exactly begin with an uppercase `T` and lowercase `en`.

3.12.1.3 The SET SEARCH BEGIN Command — The general form of the command to move the cursor to the beginning of a target is:

```
SET [SEARCH ]BEGIN
```

After you enter the command, the keypad editor locates the cursor on the first character of a target string that matches the current search model when you use the FIND or FINDNEXT function. This is the keypad editor's default setting.

3.12.1.4 The SET SEARCH END Command — The general form of the command to move the cursor to the end of a target is:

```
SET [SEARCH ]END
```

After you enter the command, the keypad editor locates the cursor just to the right of the last character of the target string when you use the FIND or FINDNEXT function.

3.12.1.5 The SET SEARCH BOUNDED Command — The general form of the command to limit a search is:

```
SET [SEARCH ]BOUNDED
```

After you enter the command, the keypad editor limits each search to the page in which the cursor is located when you use the FIND or FINDNEXT function. The editor uses the current definition of a page, either the default definition (the part of a file between the cursor and a Formfeed character) or the latest definition that you specified with the SET ENTITY PAGE command. The FIND and FINDNEXT functions do not advance or back up the cursor across a page boundary unless the cursor starts exactly at a page boundary.

Normally, the keypad editor searches until it finds a target string that matches the current model or until the cursor reaches the top or bottom of the file. When you are editing files that are longer than 30 or 40 blocks, you may find it useful to limit the range of each search by using the SET SEARCH BOUNDED command. However, if a page is currently defined by a marker-string, the string must be present in the file for you to benefit from the limited search.

If your file is not divided into pages by a specific marker-string, defining a line-count for a page is the most convenient way to limit searches. When you define a page by line-count and use the SET SEARCH BOUNDED command, the keypad editor uses the line-count that you specified and searches only within that number of text lines before or after the text line that contains the cursor.

3.12.1.6 The SET SEARCH UNBOUNDED Command — The general form of the command to allow searching throughout the file is:

```
SET [SEARCH ]UNBOUNDED
```

After you enter the command, the keypad editor extends searches beyond the limits of the page that contains the cursor when you use the FIND or FINDNEXT command. This is the keypad editor's default setting.

3.12.2 The FIND Function

VT100 Keys: The PF1 key and the PF3 key on the keypad.
VT52 Keys: The blue key and the 8 key on the keypad.

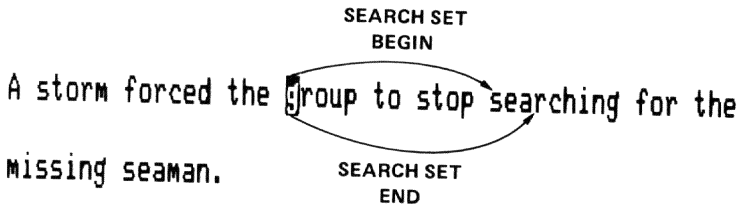
Depending on directional mode, the FIND function advances or backs up the cursor to the next occurrence of a target string that matches the model you type. Use the SET SEARCH command to specify the kind of match you require, the final position of the cursor when a search is successful, and how much of the file the keypad editor is to search.

When you press the FIND key, the keypad editor temporarily erases the top two screen lines and displays the prompt `Model:`. The keypad editor displays each character of the model string as you type it. After you terminate the model with the ADVANCE function or the BACKUP function, the keypad editor moves the cursor in the direction you specify. When the keypad editor searches far above or below the part of the file on the screen, the keypad editor may temporarily erase the top two screen lines and display the message `WORKING...` during a short pause before repainting the screen. When the keypad editor finds a target that matches the model, the keypad editor repaints the screen.

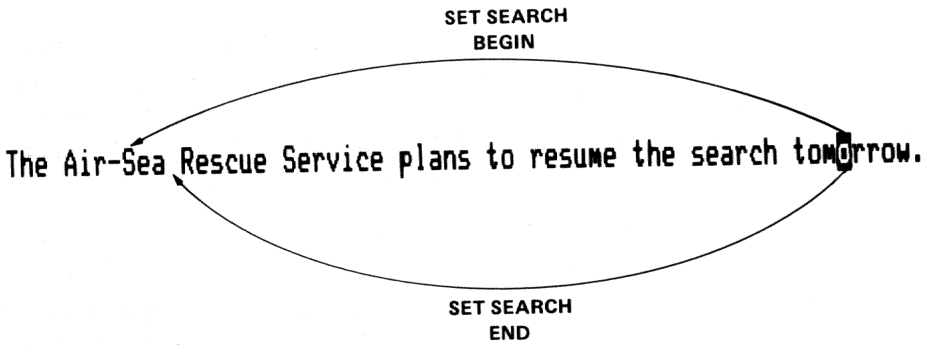
When the keypad editor finds no matching string in an extended search, the cursor moves to the top of the file or the keypad editor end-of-file symbol, and the keypad editor signals the failure.

When the keypad editor finds no matching string in a limited search, the cursor advances or backs up to the nearest page boundary (according to the current definition of page), and the keypad editor also signals the failure.

Examples: The first example illustrates how the keypad editor moves the cursor in a successful general search with the model `sea`. When the cursor's final position is set to the beginning of the target, the keypad editor places it on the `s` of `sea`. When the cursor's final position is set to the end of the target, the keypad editor places it just to the right of the `a` in `sea`.



The second example illustrates how the keypad editor moves the cursor in a successful exact search with the model `Sea`.



The model can be up to 60 characters long and contain any printing characters or line terminators.

To search directly for a more distant occurrence of a string than the nearest one, use the special keypad editor technique for repeating a function. Section 3.12 describes how to repeat any keypad editor function.

The FIND function is invalid:

1. In the backup mode when the cursor is at the top of the file.
2. In the advance mode when the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

3.12.3 The FINDNEXT Function

VT100 Key: The PF3 key on the keypad.
VT52 Key: The 8 key on the keypad.

Depending on the directional mode, the FINDNEXT function advances or backs up the cursor to the nearest occurrence of a target string that matches the latest search model you typed. When the keypad editor searches far above or below the part of the file on the screen, the keypad editor may temporarily erase the top two screen lines and display the message WORKING... during a short pause before repainting the screen. When the keypad editor finds a target that matches the model, the keypad editor repaints the screen.

The FINDNEXT function has the same effect as the FIND function. FINDNEXT also is affected by what you have specified for matches, final cursor location, and search limits. These details are the same as for the FIND function.

The FINDNEXT function is invalid:

1. When the keypad editor has no search model (because you have not completed a FIND function yet).
2. In the backup mode when the cursor is at the top of the file.
3. In the advance mode when the cursor is on the end-of-file symbol.
4. When a prompt is displayed.

3.13 Repeating Functions

The keypad editor provides four methods for repeating functions:

1. With either a VT100 or a VT52 terminal, you can repeat most standard and alternate functions up to 65535 times by preceding the function that you want to repeat with the following sequence:

   GOLD *integer*

   For example, the following sequence advances the cursor five characters to the right:

   ADVANCE GOLD 5 CHAR

   The following sequence makes the keypad editor advance the cursor directly to the fifth occurrence of the letter e:

   GOLD 5 GOLD FIND e ADVANCE

The functions that you cannot repeat with the special GOLD function sequence are:

- The DELETE function, which you can use to correct the integer that you type.
- The CTRL/U function, which you can use to correct the integer that you type.
- The SPECINS function, which must use the integer as the ASCII decimal code for a character.

You can cancel a repeat function entirely by typing CTRL/C or CTRL/Z while the `Repeat:` prompt is displayed.

2. With a VT100 terminal on which the autorepeat feature is enabled, you can repeat any standard function by holding down the key that controls the function until the keypad editor has executed the function as many times as you want. (The *VT100 User Guide* describes how to enable and disable the autorepeat function.)

3. With a VT52 terminal, you can repeat any standard function by simultaneously pressing the Repeat key and the key that controls the function you want to repeat. Hold down the keys until the keypad editor has executed the function as many times as you want.

4. With either a VT100 or a VT52 terminal, you can store and repeatedly execute sequences of functions and commands by using the keypad editor's macro feature. The macro feature is described in Chapter 5.

Chapter 4
Editing or Creating a File

This chapter describes the keypad editor functions that can modify a file. Before you study this chapter, you should understand the cursor movement functions described in Chapter 3 because the following explanations depend upon them.

4.1 Inserting Material into the File

The keypad editor provides several ways for you to insert material into the file that you are editing or creating.

To insert text, type the text on the keyboard exactly as you would on a standard typewriter. The only characters that you cannot insert directly in this way are special characters such as the Escape character or the Linefeed character. Text insertion is the keypad editor's default function.

To insert an Escape character or other special characters that you cannot type directly, such as the Linefeed character, use the SPECINS function. The SPECINS function inserts a character by its ASCII code.

To insert a blank line, press the Return key or use the OPENLINE function.

To insert the last character, word, or line that you erased, use the UNDELCHAR, UNDELWORD, and UNDELLINE functions. Section 4.2 describes these functions.

To insert material that you have stored in the keypad editor's paste buffer, use the PASTE, REPLACE, and SUBSTITUTE functions. Sections 4.3 and 4.4 describe these functions.

To insert material that is in a different file, use the keypad editor's auxiliary input file features. Chapter 5 describes how to use auxiliary files.

The following sections describe how to insert material by typing on the keyboard.

4.1.1 Typing on the Keyboard: The Default Function

The keypad editor's default function is insertion. The keypad editor processes the output from each key on the keypad and keyboard as if it is a function, such as PAGE or TOP, or part of a function, such as the integer in a repeated function or the letters of a search model you are typing. When the keypad editor cannot associate a key with a function, the keypad editor processes the key as if you mean to insert a character in the file.

Therefore, to insert any valid characters in the file, move the cursor to the location where you want to make an insertion and type the insertion on the keyboard. The keypad editor immediately shows each character you insert.

You cannot insert any characters by typing on the keypad or by using the VT100 arrow keys. Also, the set of control characters that you can directly insert depends on the operating system that you are using. However, the SPECINS function provides an indirect way to insert most of the control characters.

4.1.2 Inserting the Line Terminators

The most common line terminator in text files is the New Line terminator. To insert the New Line terminator, type the Return key on the keyboard.

The Linefeed, Vertical Tab, and Formfeed line terminators are less common. To insert them, you must type the CTRL/ key sequence or SPECINS function sequence that is listed in Table 4–1. Table 1–2 shows the symbols that the keypad editor uses to display the line terminators on VT52 and VT100 terminals.

Table 4–1: Inserting the Linefeed, Vertical Tab, and Formfeed Line Terminators

Line Terminator	To Insert, type:
Linefeed	GOLD 1 0 GOLD SPECINS
Vertical Tab	CTRL/K or CTRL/k
Formfeed	CTRL/L or CTRL/l

4.1.3 Inserting Blank Lines: The OPENLINE Function

The OPENLINE function has the same effect as typing the Return key on the keyboard followed by the leftarrow function. The keypad editor inserts the New Line terminator and places the cursor on it. Characters that were originally to the right of the cursor move down one screen line, and the keypad editor scrolls all lower lines downward.

Example: The following example shows a section of a file before and after two consecutive OPENLINE functions. The ↵ symbol is shown only to clarify that the OPENLINE function inserts a line terminator and that the keypad editor places the cursor on it (the keypad editor does not display the symbol).

```
BEFORE:  One way to do five consecutive OPENLINE functions is to↵
         type the following sequence:  GOLD 5 GOLD OPENLINE↵

AFTER:   One way to do five consecutive OPENLINE functions is to↵
         type the following sequence:  ↵
         ↵
         GOLD 5 GOLD OPENLINE↵
```

The OPENLINE function is invalid:

1. When you are inspecting a file.
2. When any prompt is displayed.
3. When earlier insertions have made the input file as large as it can be for the space that the keypad editor has allocated for the output file.

4.1.4 Inserting Characters by ASCII Code: The SPECINS Function

The SPECINS function inserts the character whose ASCII code (in decimal) you specify. The function requires a special GOLD function sequence, as follows:

GOLD *integer* GOLD SPECINS

When you use this sequence, the keypad editor interprets the integer you type as a decimal value, evaluates the seven low order bits, and inserts the corresponding ASCII character. Therefore, except for the Null character, you can use the SPECINS function to insert any ASCII character. For example, the following sequence inserts the Escape character (ASCII octal 033, decimal 27).

GOLD 27 GOLD SPECINS

4.1.5 Repeated Insertion of Single Characters and Strings

The methods for repeating a function that are described in Chapter 3 are also useful when you want to insert several consecutive single non-numeric characters. For example, if you want to insert a line of 50 asterisks as a border for a comment block in a program, the following sequence is the quickest method to produce the pattern shown:

GOLD 50 *
```
**************************************************
```

You can also insert a string several times by combining the CUT and PASTE functions with techniques for repeating. (Details about the CUT and the PASTE functions are presented later in this chapter.) The general procedure is as follows:

1. Type in the string.
2. Use the CUT function to store it in the paste buffer.
3. Repeat the PASTE function as many times as you want.

For example, with the string "[ ] " (the string is [SP][SP]) in the paste buffer, the following sequence creates the pattern shown.

GOLD 20 GOLD PASTE
```
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

The VT100 autorepeat feature and VT52 Repeat key offer another way to insert consecutive single characters. For example, when you hold down the *X* key on a VT100 for about one second, the VT100 begins to autorepeat the character and you see *X* inserted at about 15 characters per second. This feature is useful when you cannot count an exact number of insertions in advance, but it is slower than the method for multiple insertion described above.

4.2 Erasing and Restoring Text

The keypad editor has three sets of functions to erase and restore single characters, words, and text lines. Directional mode has no effect on any of the functions. For each erasure function, the keypad editor stores the erasure in the corresponding character, word, or line buffer and therefore you can restore the last erasure to the file. However, after a multiple erasure, you can restore only the last erasure you made.

Erasing and Restoring Single Characters

The DELCHAR function erases the cursor's current character.

The DELETE function erases the character just to the left of the cursor.

The UNDELCHAR function restores the character you erased with the last DELCHAR or DELETE function.

Erasing and Restoring Words

The DELWORD function erases the cursor's current character and all other characters between the cursor and the beginning of the next word.

The LINEFEED function erases all characters between the cursor and the end of the preceding word.

The UNDELWORD function restores the string you erased with the last DELWORD or LINEFEED function.

Erasing and Restoring Text Lines

The DELLINE function erases the cursor's current character and all other characters between the cursor and the beginning of the next text line (that is, through the line terminator for the current text line).

The DELEOL function is similar to DELLINE except that the DELEOL function erases a line terminator only if the cursor is originally on one.

The CTRL/U function erases all characters between the cursor and the end of the preceding text line. If the cursor is originally on a line terminator, the CTRL/U function erases the entire preceding text line.

The UNDELLINE function restores the string you erased with the last DELLINE, DELEOL, or CTRL/U function.

4.2.1 The Character, Word, and Line Buffers

The keypad editor uses a character buffer, a word buffer, and a text line buffer to store the last character, word, or line erasure that you make. The UNDELCHAR, UNDELWORD, and UNDELLINE functions copy the contents of the corresponding buffer back to your file at the cursor's location when you use the functions.

The definitions for character, word, and text line that the keypad editor uses are listed in Chapter 1. The maximum capacities of the buffers are as follows:

- Character buffer — One line terminator or ASCII character.
- Word buffer — 80 characters
- Text line buffer — 132 characters

When you start a keypad editor session, the buffers are empty. For each character, word, and text line erasure during the session, the keypad editor discards the current contents of the corresponding buffer and stores the erased material.

In certain cases that are described in Section 4.4 for the REPLACE and SUBSTITUTE functions, the keypad editor also stores the string that is removed from your file in the word buffer.

When you stop editing, the contents of the character, word, and text line buffers are preserved until you return to the RT-11 monitor level or the RSX-11M or RSX-11M-PLUS MCR level.

4.2.2 The DELCHAR Function

VT100 Key: The comma (,) key on the keypad.
VT52 Key: The 6 key on the keypad.

The DELCHAR function erases the cursor's character and stores it in the character buffer in case you want to restore it to the file. The keypad editor shifts to the left any characters that remain in the text line. The keypad editor also scrolls lower lines upward.

Example: The following example illustrates what happens when the DELCHAR function erases a single character within a text line.

```
BEFORE:  515360775 423 695-44T R
         800422961 X504 277-03T W
         209489962 799 068-01H H

AFTER:   515360775 423 695-44T R
         800422961 504 277-03T W
         209489962 799 068-01H H
```

The following example illustrates what happens when the DELCHAR function erases a line terminator.

```
BEFORE:  All the lights are frozen;↵
          The cursor blinks blandly.↵
         Soon, I shall see the dump.↵

AFTER:   All the lights are frozen;↵
          The cursor blinks blandly.Soon, I shall see the dump.↵
```

The DELCHAR function erases all line terminators as if they are single characters. Therefore, although the New Line character is represented in the file by the Carriage Return–Linefeed pair, the DELCHAR function removes both characters.

The DELCHAR function is invalid:

1. When you are inspecting a file.
2. When the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

4.2.3 The DELETE Function

VT100 and VT52 Key: The Delete key on the keyboard.

The DELETE function erases the character at the cursor's left and stores it in the character buffer in case you want to restore it to the file. The keypad editor shifts the cursor, the cursor's character, and any characters that remain in the text line, moving them to the left. The keypad editor also scrolls lower lines upward.

Example: The following example illustrates what happens when the DELETE function erases a single character within a text line.

```
BEFORE:  515360775 423 695-44T R
         800422961 X504 277-03T W
         209489962 799 068-01H H

AFTER:   515360775 423 695-44T R
         800422961 504 277-03T W
         209489962 799 068-01H H
```

The following example illustrates what happens when the DELETE function erases a line terminator.

```
BEFORE:  All the lights are frozen;↵
          The cursor blinks blandly.↵
         Soon, I shall see the dump.↵

AFTER:   All the lights are frozen;↵
          The cursor blinks blandly.Soon, I shall see the dump.↵
```

When you are responding to a prompt, the DELETE function also erases the character to the left of the prompt's cursor. Therefore, the DELETE function is one way to correct a typing error in a response. However, in this case, you cannot restore the characters that you erase.

The DELETE function is invalid:

1. When you are inspecting a file.
2. When the cursor is at the top of the file.

4.2.4 The UNDELCHAR Function

VT100 Keys: The PF1 key and the comma (,) key on the keypad.
VT52 Keys: The blue key and the 6 key on the keypad.

The UNDELCHAR function inserts the last character that you erased with either the DELCHAR function or the DELETE function. If you used the DELCHAR function to erase the last single character, the keypad editor places the cursor on the restored character. If you used the DELETE function to erase the last single character, the keypad editor places the cursor to the right of the restored character. In both cases, the keypad editor shifts characters in the original text line to the right to make room for the restored character.

Example: The following example assumes that the character buffer holds one period (.) as the result of a previous DELCHAR function. The example shows a segment of a file before and after using the UNDELCHAR function.

BEFORE:
```
DIGITAL distributes KED as the file KEDSAV on
distribution volume.
```

AFTER:
```
DIGITAL distributes KED as the file KED.SAV on
the distribution volume.
```

The keypad editor's functions that restore erasures to the file are not primarily intended for moving text to new locations in the file, but you can use them for that purpose. For example, one way to change the location of an Escape character that is in the file is to erase it with the DELCHAR function or the DELETE function, move the cursor to where the Escape character should appear, and then use the UNDELCHAR function to insert it.

The UNDELCHAR function is invalid:

1. When you are inspecting a file.
2. When the character buffer is empty (because you have not completed either a DELCHAR function or a DELETE function).
3. When a prompt is displayed.
4. When earlier insertions have made the input file as large as it can be for the space that the keypad editor has allocated for the output file.

4.2.5 The DELWORD Function

VT100 Key: The hyphen (-) key on the keypad.
VT52 Key: The 9 key on the keypad.

The DELWORD function erases the cursor's character and all characters to the right that are in the same word. The keypad editor stores the characters it erases in the word buffer in case you want to restore them to the file. The keypad editor shifts any characters that remain in the text line to the left. The keypad editor also scrolls lower lines upward.

Example: The following example shows a segment of a file before and after using the DELWORD function to erase the part of a word that is to the cursor's right.

BEFORE: `The sponsors gave some 26 or 27 prizes to adults.`

AFTER: `The sponsors gave 26 or 27 prizes to adults.`

The DELWORD function always erases at least one character. Erasing continues through the last character of the cursor's original word.

The keypad editor defines a word as follows:

- Any line terminator
- A Horizontal Tab character
- A string of spaces at the beginning of a text line
- A string of spaces flanked by Horizontal Tab characters or line terminators
- A string of other characters and the spaces that follow the string (you cannot easily distinguish these spaces from the blank the keypad editor creates for a Horizontal Tab character, but note that a Horizontal Tab character is always a word by itself.)

The DELWORD function is invalid:

1. When you are inspecting a file.
2. When the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

4.2.6 The LINEFEED Function

VT100 and VT52 Key: The Linefeed key on the keyboard.

The LINEFEED function erases characters from the cursor's left through the first character in a word. The keypad editor stores the characters it erases in the word buffer in case you want to restore them to the file. The keypad editor shifts any characters that remain in the text line to the left. The keypad editor also scrolls lower lines upward.

Example: The following example shows a segment of a file before and after using the LINEFEED function to erase the part of a word that is to the cursor's left.

BEFORE:
```
The RT-11 command to start KED is

        R SY:KED
```

AFTER:
```
The RT-11 command to start KED is

        R KED
```

The LINEFEED function always erases at least one character. Therefore, when the cursor is on the first character of a word, the LINEFEED function generally erases the preceding word.

You can insert a Linefeed character into the file with the SPECINS function.

The LINEFEED function is invalid:

1. When you are inspecting a file.
2. When the cursor is at the top of the file.
3. When a prompt is displayed.

4.2.7 The UNDELWORD Function

VT100 Keys: The PF1 key and the hyphen (-) key on the keypad.
The PF1 key and the LINEFEED key on the main keypad.
VT52 Keys: The blue key and the 9 key on the keypad.

Except as noted below, the UNDELWORD function inserts the last string you erased with either the DELWORD function or the LINEFEED function. If you used the DELWORD function to erase the last word, the cursor finishes on the first character restored. If you used the LINEFEED function, the cursor finishes to the right of the last character restored. In both cases, the keypad editor shifts characters in the original text line to the right to make room for the restored characters.

Example: The following example assumes that the word buffer holds the string ###-# as the result of a previous LINEFEED function. The example shows a segment of a file before and after using the UNDELWORD function.

BEFORE:

```
The general form of a Social Security number is

#-####
```

AFTER:

```
The general form of a Social Security number is

###-#-####
```

The functions that restore erasures to the file are not intended primarily for moving text to new locations in the file, but you can use them for that purpose. For example, one way to reverse the order of two words is to erase one with the DELWORD function or the LINEFEED function, move the cursor where you want to insert the erasure, and use the UNDELWORD function.

NOTE

The SUBSTITUTE function, and in some cases the REPLACE function, discards the current contents of the word buffer and stores the string it removes in the word buffer. The two functions are described in Section 4.4.

The UNDELWORD function is invalid:

1. When you are inspecting a file.
2. When the word buffer is empty (because you have not completed either a DELWORD function or a LINEFEED function).
3. When a prompt is displayed.
4. When earlier insertions have made the input file as large as it can be for the space that the keypad editor has allocated for the output file.

4.2.8 The DELLINE Function

VT100 Key: The PF4 key on the keypad.
VT52 Key: The unlabeled grey key on the keypad.

The DELLINE function erases the cursor's character and all other characters to its right through the next line terminator. The keypad editor stores all characters it erases in the line buffer in case you want to restore them to the file. If the cursor's original position was column 1, the keypad editor scrolls lines below the erasure upward. If the cursor's original position was to the right of column 1, the keypad editor concatenates the part of the text line that is to the left of the cursor with the next text line, and then scrolls upward (if possible).

Example: The following example shows a segment of a file before and after using the DELLINE function to erase an entire line.

```
BEFORE:  296   WENTWORTH    ARRIVED 1440  CALL AT 1700
         /CONFIRM NEXT RECORD//
         297   SOREL        CANCELED

AFTER:   296   WENTWORTH    ARRIVED 1440  CALL AT 1700
         297   SOREL        CANCELED
```

The DELLINE function always erases at least one character. Erasing continues through the line terminator of the cursor's original text line.

The DELLINE function is invalid:

1. When you are inspecting a file.
2. When the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

4.2.9 The CTRL/U Function

VT100 and VT52 Keys: Hold down the CTRL key while you type the letter U on the keyboard.

The CTRL/U function erases all characters to the left of the cursor's character and in the same text line. The keypad editor stores the characters it erases in the line buffer in case you want to restore them to the file. The function always erases at least one character. If the cursor starts in column 1, the keypad editor erases the preceding text line entirely and scrolls upper lines downward. If the cursor starts to the right of column 1, the keypad editor erases the characters between the preceding line terminator and the cursor.

Example: The following example shows a segment of a file before and after using the CTRL/U function to erase an entire line.

BEFORE:

```
With BASIC-11, one way to use the CHR$ function to send
an Escape character to a terminal is
  //JB: PLEASE CHECK THIS.//
    PRINT CHR$(27)
```

AFTER:

```
With BASIC-11, one way to use the CHR$ function to send
an Escape character to a terminal is
    PRINT CHR$(27)
```

When you are responding to a prompt, the CTRL/U function erases all of the characters you have typed. Therefore, the function is one way to correct a typing error in a response. However, in this case you cannot restore the characters that you erase.

The CTRL/U function is invalid:

1. When you are inspecting a file.
2. When the cursor is at the top of the file.

4.2.10 The DELEOL Function

VT100 Keys: The PF1 key and the 2 key on the keypad.
VT52 Keys: The blue key and the 2 key on the keypad.

The DELEOL function erases the cursor's character and all characters to the right up to (but not including) the next line terminator. The keypad editor stores the characters it erases in a line buffer in case you want to restore them to the file.

Example: The following example shows a segment of a file before and after using the DELEOL function to erase the characters that are between the cursor and the line terminator.

BEFORE:
```
200 IF T$ = 'NONE' THEN GOTO 1000\ REM ****1000 ENTRY
210 REM REQUEST BUYER INFORMATION
```

AFTER:
```
200 IF T$ = 'NONE' THEN GOTO 1000
210 REM REQUEST BUYER INFORMATION
```

The function always erases at least one character. If the cursor starts on a line terminator, the effect is the same as erasing an entire text line, and the keypad editor scrolls lower lines upward. If the cursor starts to the left of a line terminator, the keypad editor shifts the line terminator to the left and generally does not scroll.

The DELEOL function is invalid:

1. When you are inspecting a file.
2. When the cursor is on the end-of-file symbol.
3. When a prompt is displayed.

4.2.11 The UNDELLINE Function

VT100 Keys: The PF1 key and the PF4 key on the keypad.
VT52 Keys: The blue key and the unlabeled grey key on the keypad.

The UNDELLINE function inserts the last string that you erased with the DELLINE function, the DELEOL function, or the CTRL/U function. If you used the DELLINE function or the DELEOL function to erase the last line, the cursor finishes on the first character restored. If you used the CTRL/U function to erase the last line, the cursor finishes to the right of the last character restored. In all three cases, the keypad editor shifts characters in the original text line to the right to make room for the restored characters; and in most cases, the keypad editor also scrolls the file.

Example: The following example assumes that the line buffer holds the string **HIGH SCORE FOR THE SEPTEMBER CYCLE (beginning with two spaces) as a result of a previous DELEOL function. The example shows a segment of a file after using the UNDELLINE function. The example also simulates how the keypad editor wraps a line when a restored erasure is too long to fit on the cursor's original screen line.

```
BEFORE:  ART   4.0     22      19      .864N
         MATH  4.0     31      30      .968N
         CHEM  6.0     30      12      .4N

AFTER:   ART   4.0     22      19      .864N
         MATH  4.0     31      30      .968   **HIGH SCORE FOR THE S
         ♦ EPTEMBER CYCLE N
         CHEM  6.0     30      12      .4N
```

The functions that restore erasures to the file are not primarily intended for moving text to new locations in the file, but you can use them for that purpose. For example, to move a line of code out of a FORTRAN DO-loop, erase it with the DELLINE function, move the cursor to the beginning of a line outside the loop, and use the UNDELLINE function.

The UNDELLINE function is invalid:

1. When you are inspecting a file.
2. When the line buffer is empty (because you have not completed a DELLINE, DELEOL, or CTRL/U function).
3. When a prompt is displayed.
4. When earlier insertions have made the input file as large as it can be for the space that the keypad editor has allocated for the output file.

4.3 Moving, Holding, and Copying Segments of Text

The keypad editor provides functions for selecting any part of a file, removing the selection, moving or copying into another location, and combining selections from different locations. Directional mode has no direct effect on any of these functions, but it indirectly determines the location, size, and destination of the selection by either advancing or backing up the cursor.

Selecting a Part of a File

The first step in moving, holding, and copying a selection is to create a select range by using the SELECT function and then moving the cursor.

The SELECT function marks an absolute location in the file. Then, when you either advance or back up the cursor, the characters it passes enter the select range. Therefore, a selection generally is defined as the string of characters between the cursor's current location and the cursor's original location when you used the SELECT function.

You can cancel a select range by using the RESET function.

A right select range extends to the right of the cursor's original SELECT location. A left select range extends to the left of the cursor's original SELECT location. There are minor differences between the ways the keypad editor processes right and left select ranges, but a little experience will show that these differences are easy to recognize and anticipate.

Removing a Selection

When a select range includes all of the consecutive characters that you want to move or copy, you can use the CUT or APPEND function to erase the selection and store it in the paste buffer.

The CUT function discards the contents of the paste buffer and stores your current selection. The APPEND function preserves the contents of the paste buffer and adds the new selection at the end of the paste buffer. Therefore, use the APPEND function when you want to combine selections from different parts of a file in the paste buffer.

Inserting Selections

When the paste buffer contains the selection you want to move or copy, you can use the PASTE function to insert it wherever you like. You can move a selection by cutting it, moving the cursor elsewhere, and pasting the paste buffer contents there. You can copy a selection by pasting it back in place before moving away from its original location.

For example, one way to change the order of several lines in a file is to use the CUT function to put the line that should be first into the paste buffer, and then use the APPEND function to put the remaining lines in the paste buffer in order. Finally, use the PASTE function to re-insert the ordered list.

4.3.1 The Paste Buffer

The keypad editor functions for moving, holding, and copying segments of text depend on the special internal buffer that is called the paste buffer. When you start a keypad editor session, the paste buffer is empty. After you use the CUT or APPEND function, the paste buffer generally contains the material that you removed most recently with the CUT or APPEND function. However, you can empty the paste buffer by using the CLEAR PASTE command. The next section describes the command in detail.

When you stop editing, the contents of the paste buffer are preserved until you return to the RT-11 monitor level or the RSX-11M or RSX-11M-PLUS MCR level.

The size of the paste buffer depends on the operating system you are using. The minimum size is 512 characters. With the different RT-11 monitors, the actual size available is often 2000 characters or more, depending on the monitor you are using, the amount of memory available, and the size of the system that is resident. With the RSX-11M or RSX-11M-PLUS systems, the actual size depends on how the keypad editor was installed or run, and you can increase the paste buffer size for any editing session by following the procedure that is described in Section 2.3.4.

4.3.2 Clearing the Paste Buffer: The CLEAR PASTE Command

The general form of the command to clear the paste buffer is:

```
CLEAR PASTE
```

The CLEAR PASTE command discards the contents of the paste buffer. The command is especially useful when you want to use the REPLACE function to erase a multi-line segment of a file.

4.3.3 The SELECT Function

VT100 and VT52 Key: The period (.) key on the keypad.

The SELECT function by itself has no visible effect. Until you use a function that processes the characters that are in the select range, any function that moves the cursor adds or subtracts characters from the select range. All keypad editor functions, including inserting and erasing, are valid after you use the SELECT function.

The way that the keypad editor displays the characters that are in a select range depends on the terminal, the form of the cursor, and the version of the keypad editor that you are using. The following list summarizes the different display methods:

- With KED on a VT100 that has the advanced video options —

  For both the block cursor ■ and the underline cursor (_), KED displays each character in the select range in the reverse video form. For example, if the VT100 is normally displaying light characters on a dark background, the characters in a select range appear as dark characters on a light background. The description of the CUT function in the following section includes a simulated reverse video display.

- With KED on a VT100 that does not have the advanced video options —

  For the block cursor ■, KED displays each character in the select range in the reverse video form. For the underline cursor, KED underlines each character in the select range.

- With K52 on either a VT52 or a VT100 —

  K52 displays characters that are in the select range in exactly the same way as for characters that are outside of the select range. K52 cannot change the display attributes.

A left select range is to the left of the original SELECT location. It always includes the cursor's character. When you are building a left select range, almost anything you insert goes into the file but not into the select range.

A right select range is to the right of the original SELECT location. It never includes the cursor's character but always stops with the character at the cursor's left. When you are building a right select range, almost anything you insert goes into the file and into the select range as well.

You can change direction at any time when you are building a select range. You can also cancel a select range entirely with the RESET function and start a select range from a new location by using the SELECT function.

The SELECT function is invalid when a prompt is displayed.

4.3.4 The CUT Function

VT100 Key: The 6 key on the keypad.
VT52 Key: The 3 key on the keypad.

The CUT function erases all characters that are in the select range, discards the contents of the paste buffer, and stores the selection in the paste buffer. The keypad editor shifts those characters to the right of the cursor in the cursor's line to the left, and scrolls the file upward or downward.

If a select range is empty and if the cursor is at a search target, the CUT function erases the search target and stores the characters in the paste buffer.

Example: The following example shows a segment of a file before and after using the CUT function to remove a selection. The example also illustrates that the keypad editor does not display line terminators within a select range. Therefore, a blank line, which contains only a line terminator, seems to break the select range into two parts.

With the cursor at the top of the select range that is shown in the "before" part of the example, one sequence of functions that would create the select range is:

ADVANCE BLINE BLINE BLINE BLINE BLINE

BEFORE:

```
You can build a select range by moving the cursor
forward or backward. The CUT and PASTE
functions work in the same way for both cases.

XED places the cursor to the right of the last
character inserted with the PASTE function.
Building a select range backward is convenient
when you decide to move some text or program
statements that you have just typed.
```

AFTER:

```
You can build a select range by moving the cursor
forward or backward. Building a select range backward is convenient
when you decide to move some text or program
statements that you have just typed.
```

When a selection is too large for the capacity of the paste buffer, the keypad editor does not signal you until you use the CUT function or the APPEND function. The most convenient way to move or copy a selection that is too large for the paste buffer is to use the keypad editor's auxiliary input file and auxiliary output file features. (Details about auxiliary files are in Chapter 5.)

NOTE

You can use the CUT function when you are inspecting a file, as well as when you are editing or creating a fle. When you are inspecting a file, the function does not erase any material from your main input file, but does allow you to remove a copy of a block of material from a file and paste it into another file. All of the other features described here are active, however.

The CUT function is invalid:

1. When you are not building a select range (because you have not used the SELECT function) and the cursor is not at a search target.
2. When the select range is empty (because the cursor is at the SELECT point).
3. When a selection is too large for the paste buffer.
4. When a prompt is displayed.

4.3.5 The APPEND Function

VT100 Key: The 9 key on the keypad.
VT52 Key: The blue key and the leftarrow key on the keypad.

There is only one difference between the CUT and APPEND functions. The CUT function discards the selection in the paste buffer and replaces it with the current selection. The APPEND function preserves the selection in the paste buffer and adds the current selection at the end of it. Therefore, when the paste buffer is empty, the CUT and APPEND functions have the same effect. All other details about the CUT function apply also to the APPEND function.

NOTE

You can also use the APPEND function like the CUT function when you are inspecting a file.

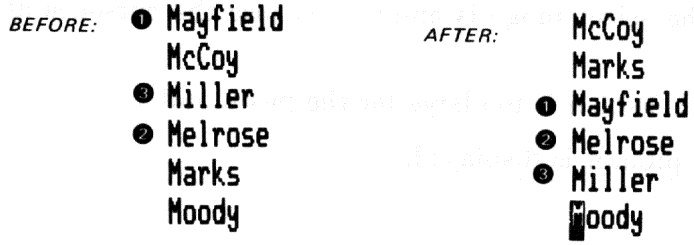
4.3.6 The PASTE Function

VT100 Keys: The PF1 key and the 6 key on the keypad.
VT52 Keys: The blue key and the 3 key on the keypad.

The PASTE function copies the selection that is in the paste buffer into the file. The keypad editor shifts the cursor's character and characters to its right (in the current text line) to the right and scrolls the file upward to make room for the insertion.

Example: The following example illustrates a convenient way to change the order of lines in a file by using the SELECT, CUT, APPEND, and PASTE functions.

The line with the symbol ❶ is stored in the paste buffer by using the CUT function. The line with the symbol ❷ is added to the paste buffer by using the APPEND function, and then the line with the symbol ❸ is also added with the APPEND function. The cursor is moved to the *M* in *Moody* and the three lines in the paste buffer are inserted by using the PASTE function.



The PASTE function is valid at any location in the file, and you can use it while you are building a select range. For example, one way to place a selection at the beginning of the paste buffer contents is as follows:

1. Move the cursor to the beginning of the selection that you want to add to the paste buffer.
2. Use the SELECT function.
3. Advance the cursor to the end of the selection.
4. Use the PASTE function to insert the paste buffer contents after the selection. The insertion is also in the new selection you are building.
5. Use the CUT function to store the new and larger selection in the paste buffer.

The PASTE function is invalid:

1. When you are inspecting a file.
2. When a prompt is displayed.
3. When earlier insertions have made the input file as large as it can be for the space the keypad editor has allocated for the output file.

4.4 Substituting and Converting Characters and Strings

Three functions perform direct substitutions in the file: REPLACE, SUBSTITUTE, and CHNGCASE.

The REPLACE function substitutes the contents of the paste buffer for a string in the file in two cases:

1. If you are building a select range and it is not empty, the REPLACE function erases and discards the characters in the select range and inserts the contents of the paste buffer.
2. If you are not building a select range but the cursor is at a search target, the REPLACE function erases and discards the target string and inserts the contents of the paste buffer. In this case, the keypad editor stores the erased material in the word buffer. If the cursor is at a valid search target and you are also building a select range, the first rule (above) takes precedence.

The SUBSTITUTE function is a convenient way to change several occurrences of the same string in a file to the string that is in the paste buffer. The keypad editor processes the SUBSTITUTE function by doing a REPLACE (according to the second rule listed above) and then a FINDNEXT in the direction you used last.

The CHNGCASE function changes lowercase letters to uppercase letters and uppercase letters to lowercase letters.

4.4.1 The REPLACE Function

VT100 Keys: The PF1 key and the 9 key on the keypad.
VT52 Keys: The blue key and the uparrow key on the keypad.

If a select range contains any characters or if the cursor is at a valid search target, the REPLACE function erases and discards the select range or search target, inserts the contents of the paste buffer, and places the cursor at the character that follows the insertion. When the paste buffer is empty, the keypad editor erases and discards the selection or search argument and inserts nothing. Therefore, the REPLACE function is a convenient way to erase a string that would require a complex combination of other erasure functions. (You can empty the paste buffer at any time with the CLEAR PASTE command.)

The keypad editor always replaces a select range when one exists. The keypad editor replaces a search target only if no select range exists and the cursor is at a valid search target.

When you use the REPLACE function to replace a search target, the keypad editor stores the search target in the word buffer.

NOTE

The REPLACE function erases a select range of any size, even when the select range exceeds the capacity of the paste buffer (when the keypad editor cannot complete a CUT or APPEND function, for example). There is no way to recover any characters you have typed in in this case. However, if you accidentally erase characters that were in the original input file you are editing, you can recover those characters by referring to a backup copy or the original input file.

The REPLACE function is invalid:

1. When you are inspecting a file.
2. When you are not building a select range (because you have not used the SELECT function) and the cursor is not at a search target.
3. When the select range is empty (because the cursor is at the SELECT point).
4. When a prompt is displayed.
5. When the paste buffer contains too much material to fit in the file.

4.4.2 The SUBSTITUTE Function

VT100 Keys: The PF1 key and the Enter key on the keypad.
VT52 Keys: The blue key and the Enter key on the keypad.

With the cursor at a valid search target, the keypad editor processes the SUBSTITUTE function by first completing a REPLACE and then completing a FINDNEXT in the direction you specified last. When you use the SUBSTITUTE function, the keypad editor repaints the screen after it completes the FINDNEXT function. The SUBSTITUTE function is affected by the current settings you have specified for exact and general matches, limited and extended searching, and the definition of a page.

You can change several occurrences of a string to whatever is in the paste buffer by repeating the SUBSTITUTE function. The sequence is GOLD *integer* GOLD SUBSTITUTE. The keypad editor accepts integers in the range from 1 to 65535.

The SUBSTITUTE function is invalid:

1. When you are inspecting a file.
2. When the cursor's original position (when you use the SUBSTITUTE function) is not on a valid search string.
3. When the keypad editor has no current search model (because you have not completed a FIND function yet).
4. In the backup mode when the cursor is at the top of the file.
5. In the advance mode when the cursor is on the end-of-file symbol.
6. When a prompt is displayed.
7. When the paste buffer contains too much material to fit in the file.

4.4.3 The CHNGCASE Function

VT100 Keys: The PF1 key and the 1 key on the keypad.
VT52 Keys: The blue key and the 1 key on the keypad.

The CHNGCASE function changes uppercase letters to lowercase letters and lowercase letters to uppercase letters in the following three cases:

1. With the cursor on a letter, the CHNGCASE function changes the case of only the cursor's character and then, depending on the directional mode, advances or backs up the cursor by one character as follows:

   - When you are not building a select range.
   - When the cursor is not at a valid search target.

2. With the cursor on a non-alphabetic character and depending on the directional mode, the CHNGCASE function only advances or backs up the cursor by one character.

3. With the cursor on any character and when you are building a select range that is not empty, the CHNGCASE function changes the case of each letter in the select range and then cancels the select range. The function does not move the cursor.

4. With the cursor on any character and when you are not building a select range but the cursor is at a valid search target, the CHNGCASE function changes the case of each letter in the search target. The function does not move the cursor.

The CHNGCASE function is invalid:

1. When you are inspecting a file.
2. In the advance mode when the cursor is on the end-of-file symbol.
3. In the backup mode when the cursor is at the top of the file.
4. When you are building a select range but the cursor is at the SELECT location (therefore, there are no characters in the select range).
5. When a prompt is displayed.

Chapter 5
Special Keypad Editor Features for Writers and Programmers

This chapter describes the keypad editor's commands and functions that provide special features for writers of memos, reports, and documentation as well as for programmers. The chapter covers:

1. Using auxiliary files

   With a small set of auxiliary file commands, you can combine files that you have stored with different file names. You can also split a single file into separate files and store them with different file names. These techniques are especially useful for moving a block of material within your main input file when the amount of material is too large for the paste buffer.

2. Using and changing right margin settings

   Although the VT100 and VT52 terminals do not allow you to set the right margin for a screen line, you can achieve that effect by setting the right margin for an editing session. You can reformat text lines that are already in your file so that they fit within the right margin, and you can have the keypad editor automatically start a new text line whenever a word that you type reaches past the right margin.

3. Using keypad editor macros

   When you want to repeat a multistep editing task several times, you can record the combination of keypad editor commands and functions as a keypad editor *macro* and execute the macro by using a single function.

4. Reordering MACRO-11 local symbols automatically

   When you use the keypad editor to create or edit source files for MACRO-11 programs, you can use a special keypad editor command to reorder local symbols so that your programs are more readable.

5. Using structured tabs

   Although the keypad editor and your operating system do not support variable tab stops, you can change the way the keypad editor processes the Tab key when you press the key at the beginning of a text line. The feature is especially useful for indenting source program statements to show the different nesting levels of procedures and groups of statements.

5.1 Using Auxiliary Files

The most common uses for the keypad editor involve only two files:

- The new file that you are creating, called the main output file.
- The existing file that you are editing, called the main input file.

You can also work with one auxiliary output file and one auxiliary input file, and you can change either auxiliary file at any time.

Figure 5–1 shows the relationship between the keypad editor and an auxiliary output file. You specify an auxiliary output file by using the OPEN OUTPUT command. After you enter the command, you can copy material from your main input file to the auxiliary output file. In this way, you can use an auxiliary output file to store a part of a larger file. If you erase the part of your main input file that you have copied and then store both output files, you will have split your main input file into two files.

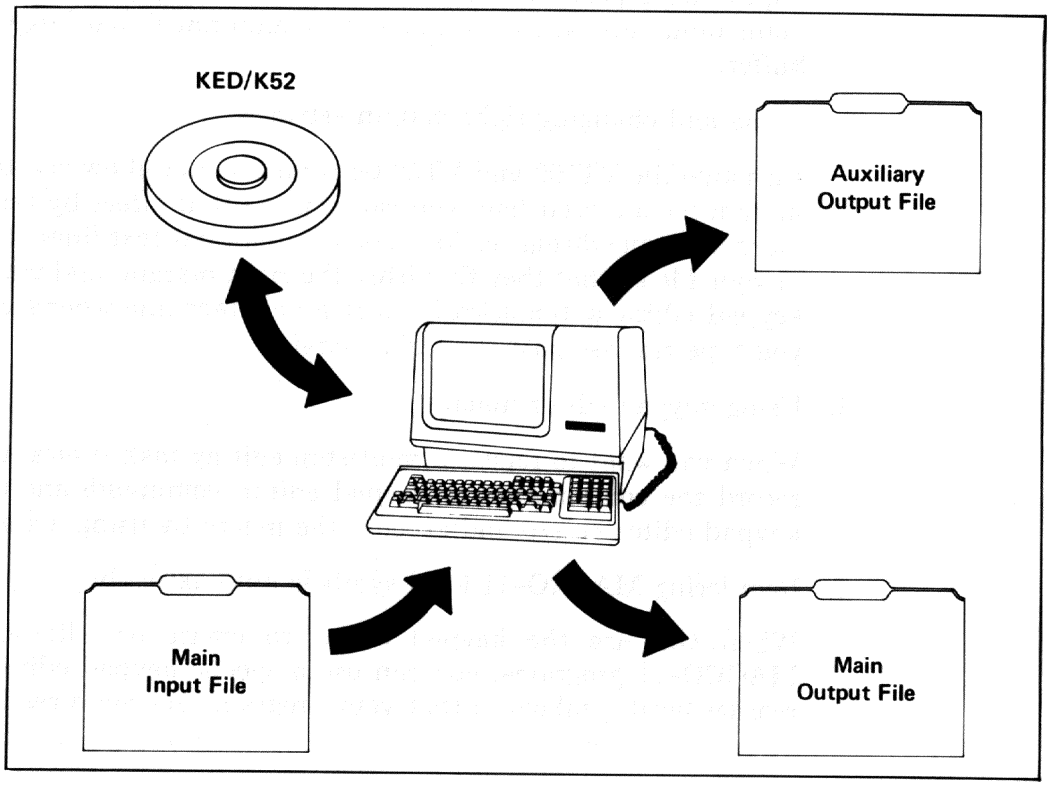


Figure 5–1: Relationship Between the Keypad Editor and an Auxiliary Output File

Figure 5–2 shows the relationship between the keypad editor and an auxiliary input file. You specify the auxiliary input file by using the OPEN INPUT command. After you enter the command, you can copy material from the auxiliary input file into your main input file at the cursor's location. When you copy an entire auxiliary input file, you will have combined the auxiliary input file with your main input file.

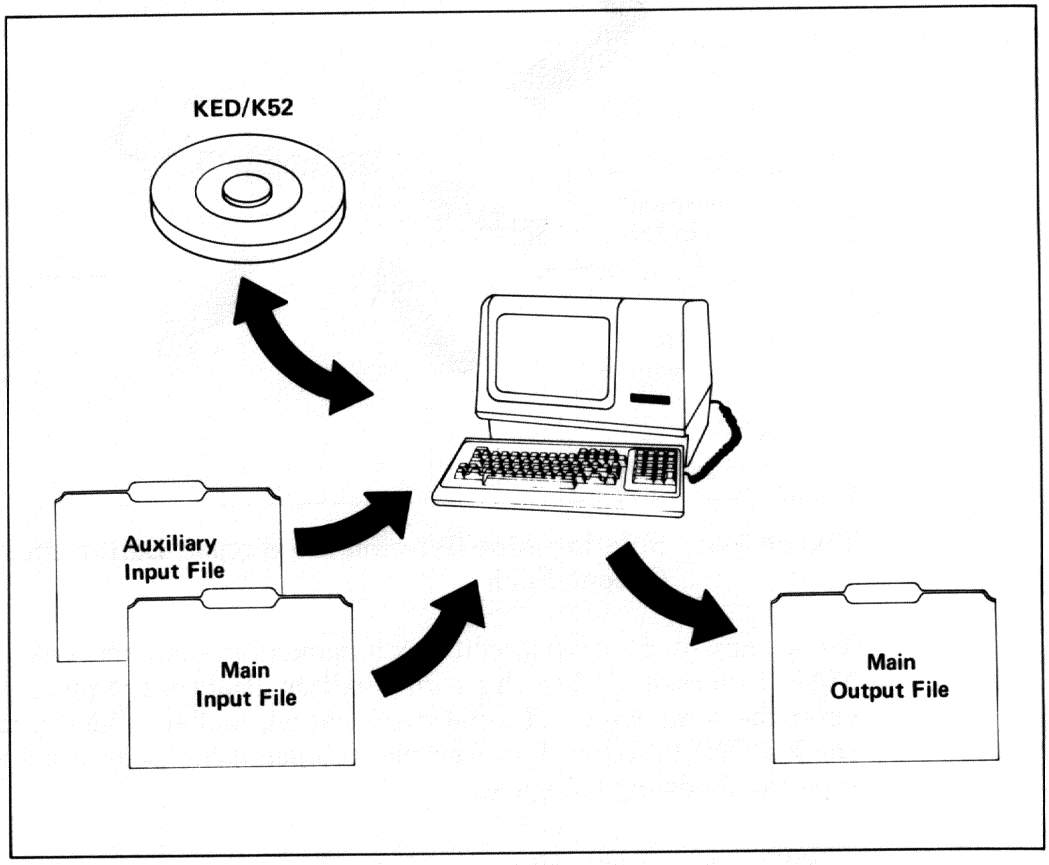


Figure 5–2: Relationship Between the Keypad Editor and an Auxiliary Input File

Figure 5–3 shows the full relationship between the keypad editor and the main and auxiliary files. You can use an auxiliary input file and an auxiliary output file at the same time, combining material from the auxiliary input file with your main input file, and copying parts of the file you are editing to an auxiliary output file. You can use only one auxiliary input file and one auxiliary output file at a time, but by changing them during an editing session, you can combine and split several files in many ways.

You can also create an auxiliary output file, store it, and then, during the same editing session, use the file as an auxiliary input file. This technique is especially useful for moving a block of material within your main input file when the amount of material is too large for the paste buffer.

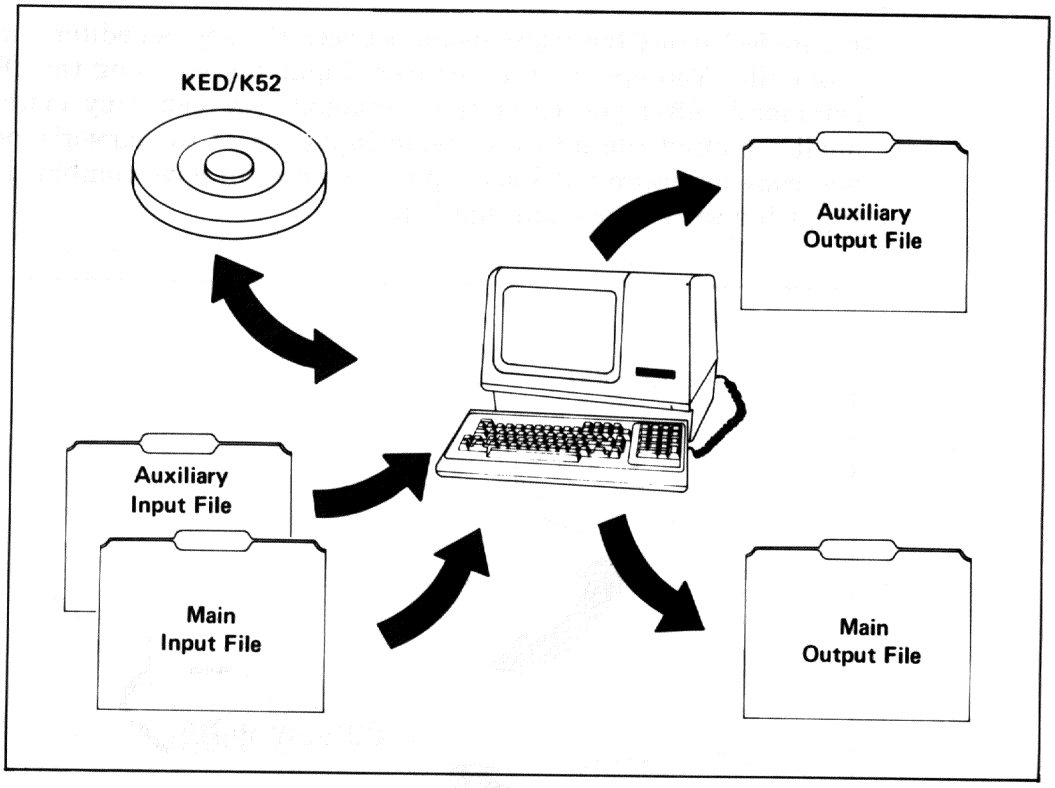


Figure 5–3: Relationships Between the Keypad Editor and All Input and Output Files

To use any of the keypad editor commands for auxiliary files, use the COMMAND function. When the keypad editor displays the prompt `Command:`, enter the command, add a file specification, and end the command by using the ENTER function. For example, to open a file as an auxiliary input file, type the following sequence:

(COMMAND) OPEN INPUT *file-spec* (ENTER)

5.1.1 Auxiliary File Specifications for Different Operating Systems

For an auxiliary file specification, you can use any of the forms that are valid for the main input and output file specifications. Chapter 2 summarizes the valid forms for the RT–11, RSX–11M, and RSX–11M–PLUS operating systems. The defaults described in that section also apply to auxiliary file specifications.

RSX–11M and RSX–11M–PLUS systems allow the use of the /CA, /–CA, and /FO command line options which are summarized in Table 2–3.

5.1.2 Using Auxiliary Files with VT52 Terminals

The keypad editor processes auxiliary files on VT100 terminals and VT52 terminals in the same way.

5.1.3 Auxiliary Output Files

Although the keypad editor protects you from changing a file while you are inspecting it, you can open an auxiliary output file, copy parts of the file you are inspecting to the auxiliary output file, and close it without any effect on the original file.

You can extract a selection from a file, modify the selection for another purpose, and print it.

During any keypad editor session, the general procedure for using auxiliary output files is as follows.

1. Start the keypad editor and complete a file specification string.
2. Use the OPEN OUTPUT command to open an auxiliary output file.
3. Use the WRITE command to copy material from your main input file to the auxiliary output file.
4. Use the CLOSE command to close and save the auxiliary output file.
5. Terminate the keypad editor session with the EXIT or QUIT command.

The four commands for auxiliary output files are OPEN OUTPUT, WRITE, CLOSE, and PURGE.

5.1.3.1 The OPEN OUTPUT Command — The general form of the command to open an auxiliary output file is:

```
[OPEN ]OUTPUT file-specification
```

The keypad editor accepts any valid file specification. The default output device is your system's default storage device. If another file with the same specification already exists, the keypad editor signals you and loads the following help message:

```
Auxiliary output file exists - Replace (Y,N)?
```

You can cancel the entire command; or you can have the keypad editor process the command, erase the existing file, and replace it with the new output you select.

Only one auxiliary output file can be open at a time. When you open a second auxiliary output file and one is already open, the keypad editor automatically closes the first one. (If you close an auxiliary output file that is empty, the file is saved but it has zero blocks of data.)

5.1.3.2 The WRITE Command — The general forms of the command to copy material from your main file to your auxiliary output file are:

```
WRITE SELECT
WRITE REST
WRITE integer PAGES
WRITE integer[ LINES]
```

In each sequence the keypad editor starts with the character the cursor is on and copies part of the file you are inspecting, editing, or creating to the auxiliary output file. The part that the keypad editor copies is always to the right of the cursor (and on lower lines). If you have already copied something with an earlier WRITE command, the keypad editor adds new material at the end of the auxiliary output file.

The form WRITE SELECT copies the current contents of the SELECT range to the auxiliary output file. When the copy operation is complete, the cursor remains at the same character at which it was positioned when the WRITE SELECT command was issued.

The form WRITE REST copies to the auxiliary output file the entire lower part of the file you are inspecting, editing or creating. When the keypad editor finishes, the cursor is on the end-of-file symbol. In effect, the keypad editor does a BOTTOM function and copies all characters from the cursor's original position through the last character in the file.

The form WRITE *integer* PAGES uses the current definition of a page and copies as many pages as you specify. When the keypad editor finishes, the cursor is on the first character that has not been copied to the auxiliary output file. The keypad editor accepts integers in the range from 1 to 65535. In effect, the keypad editor does the number of ADVANCE PAGE functions that you specify and copies all characters from the cursor's original position to the cursor's final target.

The form WRITE *integer* LINES uses the keypad editor's standard definition of text line and copies as many text lines as you specify. When the keypad editor finishes, the cursor is on the first character that has not been copied to the auxiliary output file. The keypad editor accepts integers in the range from 1 to 65535. In effect, the keypad editor does the number of ADVANCE and BLINE functions you specify and copies all characters from the cursor's original position to the cursor's final target.

5.1.3.3 The CLOSE Command — The form of the command to close and save an auxiliary output file is:

```
CLOSE
```

The CLOSE command closes the auxiliary output file, saving all information that you have copied to the auxiliary output file. There is no way to use the keypad editor's auxiliary output file feature to open a file that exists and add material to it. (See the section on auxiliary input files below for an example of how to do that.)

5.1.3.4 The PURGE Command — The form of the command to discard the contents of an auxiliary output file is:

```
PURGE
```

When an auxiliary output file is open, the PURGE command deletes the file and its contents and cancels the last OPEN OUTPUT command. Thus, all data copied to that file are lost. No files are erased or stored. If you want to use an auxiliary output file after the keypad editor completes the PURGE operation, start over with a new OPEN OUTPUT command.

5.1.4 Auxiliary Input Files

You can open an auxiliary input file at any time while you are editing or creating a different file. You can, for example, collect small files or parts of larger files and insert them in the main file you are editing. Since inserting material from an auxiliary input file means modifying your main file, this feature is not available while you are inspecting a file. You can only use it after you have started a keypad editor session to edit or create a file.

The general procedure for using auxiliary input files is as follows.

1. Start the keypad editor and complete the file specification string for editing or creating a file.
2. Use the OPEN INPUT command to open an auxiliary input file.
3. Use the SKIP command to skip past text lines or pages in the auxiliary input file that you do not want to insert in your main file.
4. Use the INCLUDE command to copy text lines or pages of the auxiliary input file to your main file.
5. Use the OPEN INPUT command to close an auxiliary input file and open another one.

5.1.4.1 The OPEN INPUT Command — The general form of the command to open an auxiliary input file is:

```
[OPEN ]INPUT file-specification
```

The keypad editor accepts any valid file specification. The default input device is your system's default storage device.

Only one auxiliary input file can be open at a time. Each time you open an auxiliary input file, the keypad editor closes the one that is open (if any). However, you can open the same auxiliary input file as many times as you like while you are using the keypad editor. You can also use any auxiliary output file as an auxiliary input file in the same session after you have closed it with the CLOSE command. (The CLOSE command has no effect on an open auxiliary input file.)

You do not need to explicitly close an auxiliary input file. Any method you use to terminate the keypad editor session automatically closes it.

When you have opened an auxiliary input file, you can either skip material in it or copy material from it to the main file you are editing.

5.1.4.2 The SKIP Command — The general forms of the command to skip material in an auxiliary input file are:

```
SKIP REST
SKIP integer PAGES
SKIP integer[ LINES]
```

The form SKIP REST skips directly to the end of the auxiliary input file, and the keypad editor then closes the auxiliary input file automatically.

The form SKIP *integer* PAGES uses the current definition of a page and skips past as many pages in the auxiliary input file as you specify. The keypad editor accepts integers in the range from 1 to 65535. If your current definition of a page does not correspond to the way you have marked pages in the auxiliary input file, this form is likely to skip to the end of the entire auxiliary input file. The keypad editor cannot detect this sort of mistake. For example, the keypad editor's default definition of a page is a string of any length with a Formfeed as the last character. If the auxiliary input file is divided into pages by the marker string -PAGE-, the form SKIP *integer* PAGES will skip past an unpredictable amount of auxiliary input file material.

The form SKIP *integer* LINES uses the keypad editor's standard definition of a text line and skips past as many text lines in the auxiliary input file as you specify. The keypad editor accepts integers in the range from 1 to 65535.

Note that there is no way to "rewind" an auxiliary input file. For example, after skipping a text line in an auxiliary input file, the text line that follows is the first one you have access to. When you have skipped or inserted the last text line of an auxiliary input file, the keypad editor automatically closes it. Any further attempt to skip or insert material will fail and the keypad editor signals that no auxiliary input file is open. When you want access to the same material in an auxiliary input file more than once, you must explicitly open the auxiliary input file each time.

5.1.4.3 The INCLUDE Command — The general forms of the command to copy material from an auxiliary input file are:

```
INCLUDE REST
INCLUDE integer PAGES
INCLUDE integer[ LINES]
```

In each form, the keypad editor copies the number of complete text lines that you specify from the auxiliary input file into the main file you are editing. The effect on your main file is generally the same as if you type the same text lines on your keyboard. The cursor, the cursor's character, and characters to the right all move to the right and downward to make room for the insertion. When the keypad editor completes the insertion, the cursor is to the right of the last character inserted from the auxiliary input file. The line terminators that were in the original material are in the same locations in the inserted copy. The keypad editor does not reformat the lines.

The form INCLUDE REST copies all of the text lines that remain in the auxiliary input file. After inserting the last text line, the keypad editor automatically closes the auxiliary input file.

The form INCLUDE *integer* PAGES uses the current definition of a page and copies as many pages from the auxiliary input file as you specify. The keypad editor accepts integers in the range from 1 to 65535.

The form INCLUDE *integer* LINES uses the keypad editor's standard definition of a text line and copies as many text lines from the auxiliary input file as you specify. The keypad editor accepts integers in the range from 1 to 65535.

5.1.5 Example: Using Auxiliary Files to Move a Large Amount of Material

The keypad editor paste buffer is usually large enough to allow you to move material around within your files. However, in some cases you are likely to want to move larger amounts of material than the paste buffer can hold. This section describes one method for using the keypad editor auxiliary file features to do this. The primary purpose of this example is to give you a complete model of a typical auxiliary file process that you can try immediately with any of your text files.

The general method is as follows:

1. Open an auxiliary output file.

   The example uses the auxiliary file name AUX.TXT, but you can use any file name that you like.

2. Copy material from your main input file to the auxiliary output file and close the auxiliary output file.

   The example copies three pages of material, but you can change that amount to whatever you like.

3. Erase the material from your main input file.

4. Move the cursor to where you want the material to appear.

   The example backs the cursor up by 150 lines, but you can use any cursor movement function that is appropriate for your file.

5. Copy the material from the auxiliary file you created to your main input file.

Opening the Auxiliary Output File

To open the auxiliary output file AUX.TXT, use the following sequence:

COMMAND OPEN OUTPUT AUX.TXT ENTER

Copying the Material That You Want to Move

Move the cursor to the beginning of the block of material that you want to move. This example assumes that the block begins at the beginning of page 8, just after the string -7- that is at the bottom of page 7. You should move the cursor with any functions that are appropriate for your file. Also, define a page in terms of the marker string that separates pages in your file.

To search forward for the string -7- use the following sequence:

(FIND) -7- (ADVANCE)

To move to the beginning of page 8 use the PAGE function:

(PAGE)

To copy pages 8, 9 and 10 to the auxiliary file use the following sequence (the cursor stops at the beginning of page 11):

(COMMAND) WRITE 3 PAGES (ENTER)

To close the auxiliary file use the CLOSE command:

(COMMAND) CLOSE (ENTER)

Erasing the Material That You Copied

An easy way to erase a large block of material is to clear the paste buffer and use the REPLACE function.

To clear the paste buffer use the CLEAR PASTE command:

(COMMAND) CLEAR PASTE (ENTER)

To define the 3-page block of material that is to be erased, put it in a select range with the following sequence (several other sequences work equally well):

(SELECT) (BACKUP) (PAGE) (PAGE) (PAGE)

To erase the material, use the REPLACE function to replace the select range with the contents of the paste buffer. Because the preceding step cleared the paste buffer, the buffer contains nothing. Therefore, the REPLACE function inserts nothing after taking out the 3-page select range:

(REPLACE)

Moving the Cursor to the Material's New Location

To back the cursor up by 150 lines use the special GOLD sequence for repeating a function:

BACKUP GOLD 150 BLINE

Copying in the Material at the New Location

To copy the auxiliary file material back into the main input file, first open the auxiliary file as an auxiliary input file. Use the following sequence:

COMMAND OPEN INPUT AUX.TXT ENTER

To copy all of the auxiliary file use the following command:

COMMAND INCLUDE REST ENTER

At this point, you have finished the process. You do not need to close the auxiliary file because the keypad editor does so automatically. However, you may want to delete the auxiliary file since you are probably finished using it. When you finish the editing session and return to the RT–11 monitor level or the MCR level in RSX systems, you can delete the file with the appropriate commands for your operating system.

5.2 Using and Changing Right Margin Settings

The keypad editor has a set of features for refining the overall format of paragraphs and lines of text. Memos, letters, and program documentation are examples of the kinds of files whose format you may want to change in order to present clear, attractive, and easy to read text. With the keypad editor's right margin features, you can:

- Use the SET WRAP command to adjust the right margin.
- Use the SET WRAP command to set the keypad editor to automatically start a new text line when you type a word that reaches past the right margin (the word wrap feature).
- Use the FILL command to reformat a selection within your file so that each text line contains only as many full words as can fit within the right margin.
- Use the SET NOWRAP command to stop the word wrap feature.

The following sections describe the right margin features, which work the same on all operating systems, in detail.

5.2.1 Right Margin Features with VT52 Terminals

With VT52 terminals, the FILL command provides the only way to reformat the text lines that are in your file. With VT100 terminals, you can use either the FILL command or the FILL function on the keypad. The other right margin features work the same way on both VT52 and VT100 terminals.

5.2.2 The SET WRAP Command

At any time during an editing session, you can use the SET WRAP command to specify or change the right margin you want the keypad editor to use. To set the right margin, specify the maximum number of characters you want the keypad editor to leave in a text line. The keypad editor also uses the right margin setting for processing the FILL command.

After you use the SET WRAP command and as you type material on the keyboard, the keypad editor automatically starts a new text line with each word that reaches past the right margin.

The three exceptions to the keypad editor's general rule for starting new text lines are as follows:

1. If you type a single word that is longer than the maximum number of characters that you have specified for the right margin, the keypad editor leaves the word in a text line by itself.
2. If you type the Tab key, the editor processes it normally, and usually the text line that contains the Horizontal Tab character reaches past the specified right margin.
3. If you insert a Carriage Return character by itself, the editor processes it normally by overwriting the beginning of the text line that contains the character.

This feature is called *word-wrap*. The keypad editor continues the word wrap process until you stop the process with the SET NOWRAP command.

The keypad editor automatically starts wrapping new text lines, but only with text that you type. The keypad editor does not automatically change any text lines that a file already contains, that you copy from an auxiliary input file, or that you insert with the PASTE, UNDELLINE, or other functions that restore erasures. Use the FILL command to reformat the existing text in a file so that words do not reach past your right margin.

The general form of the command to specify the right margin is:

SET WRAP[*line length*]

The line length must be an integer in the range 1–256. When you omit an explicit value, the keypad editor uses column 78 as the default right margin setting when the screen width is 80 and column 130 when the screen width is 132.

The following examples illustrate how the keypad editor displays the same text when you type the text in two cases:

1. Before entering a SET WRAP command — that is, before enabling the word-wrap feature.
2. After using the SET WRAP command to specify a right margin of 32.

The scale above each example shows the lengths of the lines that the keypad editor displays. In the second example, the ↓ symbol is shown only to clarify that the keypad editor automatically inserts the New Line line terminator when a word that is being entered reaches past the right margin.

Example 1: For this example, start the keypad editor in one of the following ways:

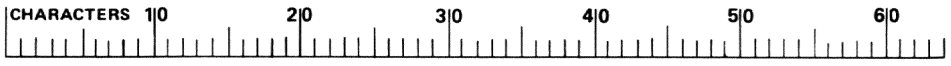
- For RT-11 systems

  Return to the monitor level. If you are using a VT100 terminal, set the screen width to 80 columns. Start the keypad editor with the RUN or R command.

- For RSX-11M and RSX-11M-PLUS systems

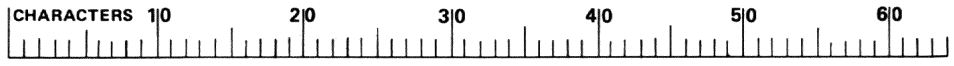
  Exit your current task with the CTRL/Z command. If you are using a VT100 terminal, set the screen width to 80 columns. Start the keypad editor with the RUN $KED or RUN $K52 command or the direct call with the task name KED or K52.

Now, type the text line that is below the scale. Do not press the Return key or enter any other line terminator while you are typing. (The following example simulates how your screen will look, but is not an exact image.)




```
The SET WRAP command sets KED to use the special word-wrap featu
↓ re. When you type a word that reaches past the right margin, K
↓ ED automatically starts a new text line.
```

For this example, enter the SET WRAP 32 command. Retype the text that you typed in the preceding examples. Do not press the Return key or enter any other line terminator.




```
The SET WRAP command sets KED to↓
use the special word-wrap↓
feature. When you type a word↓
that reaches past the right↓
margin, KED automatically starts↓
a new text line.
```

5.2.3 The SET NOWRAP Command

The SET NOWRAP command stops the word-wrap process. However, the command has no effect on the way the keypad editor processes the FILL command. For example, if you set the right margin with the SET WRAP 46 command and then enter the SET NO WRAP command, the FILL command continues to use the maximum line length of 46 characters when reformatting a select range.

The general form of the command to stop the automatic word-wrap process is:

```
SET NOWRAP
```

5.2.4 The FILL Command and VT100 FILL Function

VT100 Keys: Available as a command or as a function controlled by the PF1 key and the 8 key on the keypad.

VT52 Keys: Available only as a command.

The general form of the command to reformat the text in a select range is:

```
FILL
```

Use the FILL command (and also the FILL function on a VT100 terminal) to reformat text that you have already inserted in your file. A typical procedure for reformatting is as follows:

1. Move the cursor to the first character in the selection.
2. Use the SELECT function to begin building a select range.
3. Move the cursor just to the right of the last character in the selection.
4. Enter the FILL command.

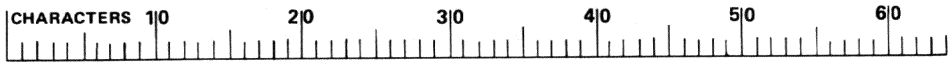
After reformatting, each text line that was in the select range contains only as many full words as possible without reaching past the right margin. Starting at the top of the select range, the keypad editor extends short text lines by moving words from lower text lines and breaks long text lines by inserting the New Line terminator after the last full word that fits within the right margin.

With VT100 terminals, you can use either the FILL command or the FILL function. They have the same effects. The default right margin depends on the screen width:

- The default is column 78 when the screen width is 80.
- The default is column 130 when the screen width is 132.

With VT52 terminals, only the FILL command is available. The default right margin is always column 78.

Examples: The following example shows the same text before and after using the FILL command. Assume that the right margin has been set to column 58. The scale above the example shows the lengths of lines that the keypad editor displays. The ↵ symbol is shown only to clarify the locations of line terminators before and after the keypad editor reformats the select range. (The "before" part of the example simulates your screen, but is not an exact image.)



BEFORE:

```
The FILL command reformats the text with in a select range so t
♦ hat no word reaches past the↵
right↵
margin.↵
The function is especially convenient when you need to use a ri
♦ ght↵
margin that is less that your screen width.↵
```

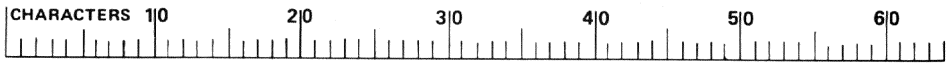
AFTER:

```
The FILL command reformats the text with in a select↵
range so that no word reaches past the right margin. The↵
function is especially convenient when you need to use a↵
right margin that is less that your screen width.
```

The keypad editor processes the FILL command in different ways when the last character within the select range is the New Line terminator and otherwise.

When the last character in the select range is the New Line terminator, the keypad editor reformats the select range and then attaches the text line that was below the select range to the last reformatted text line. However, the keypad editor does not reformat the attached line, and therefore in this case a wrapped line is often the result. The following example illustrates the effect. You can avoid creating the wrapped line by building the select range so that the bottom is at the end, not at the beginning, of a text line.

Before: (the ¶ symbol shows the locations of the original New Line terminators)



BEFORE:

```
The FILL command reformats the text with in a select range so t
♦ hat no word reaches past the¶
right¶
margin.¶
The function is especially convenient when you need to use a ri
♦ ght¶
margin that is less that your screen width.¶
¶
... ..... . .......... ..
```

After: (assume that the right margin has been set to column 58; the ¶ symbol shows the new locations of the Carriage Return–Linefeed terminators)

AFTER:

```
The FILL command reformats the text with in a select¶
range so that no word reaches past the right margin. The¶
function is especially convenient when you need to use a¶
right margin that is less that your screen width. ¶... ..
♦ ... . .......... ..
```

NOTE

DIGITAL's document formatting program RUNOFF and similar programs require files with special commands that must be located at the beginnings of text lines. When you use the FILL function to reformat lines in a file of this sort, be careful to build select ranges that do not include any of the special commands. For example, if you reformat a select range that includes a RUNOFF command, the keypad editor is likely to move the RUNOFF command so that the RUNOFF program cannot detect and process it.

The FILL command affects some special characters and line terminators in special ways, as noted in the following summary:

1. A New Line terminator is inserted immediately before each Vertical Tab character and Formfeed character.
2. A single space replaces any string of the following characters:
   - Spaces
   - Horizontal Tab characters
   - Carriage Return characters and Linefeed characters when they stand alone as well as when they are together. Therefore, if the paragraphs in your file are separated by blank lines, the FILL command will join the paragraphs. To avoid having the paragraphs joined, process each one separately or use the Vertical Tab or Formfeed characters to separate them.

5.3 Using Keypad Editor Macros

A keypad editor macro is a combination of keypad editor commands and functions that you have stored. The following examples describe three simple combinations of functions. Each combination would have exactly the same effect as one of the functions that are described in earlier chapters.

1. (RETURN) (Leftarrow) as the equivalent of the OPENLINE function

   When you press the Return key and then the leftarrow key, the effect is the same as when you use the OPENLINE function. The keypad editor inserts the New Line line terminator when you press the Return key and moves the cursor to the terminator when you press the leftarrow key.

   If you store the (RETURN) (Leftarrow) combination as a macro, the keypad editor will execute the combination each time you use the X function. In this case, your macro will do exactly what the OPENLINE function does.

2. (GOLD) 1 6 (BLINE) as the equivalent of the SECTION function

   When you use the combination, the keypad editor advances or backs up the cursor by 16 text lines. If the default definition of a section is your current definition, the effect of the combination is the same as when you use the SECTION function.

   If you store the combination as a keypad editor macro, the X function will do exactly what the SECTION function does, as long as a section is defined as 16 text lines.

3. (REPLACE) (FINDNEXT) as the equivalent of the SUBSTITUTE function

   When you use the combination, the keypad editor substitutes the contents of the paste buffer for a string in your file in two cases:

   a. When you have built a select range, the keypad editor replaces the select range.

   b. When the cursor is at a search target and you have not built a select range, the keypad editor replaces the search target.

   In the second case, the effect of the combination is the same as when you use the SUBSTITUTE function.

   If you store the combination as a keypad editor macro, the X function will do exactly what the SUBSTITUTE function does.

After you have stored a combination of commands and functions, you can execute the entire combination by using the X function. To use the X function, the alternate function of the X key on your keyboard, press the GOLD key on the keypad first, then the X key.

Keypad editor macros work the same with VT52 and VT100 terminals on the RT-11, RSX-11M, and RSX-11M-PLUS operating systems.

The LEARN command erases the last macro you specified. The keypad editor then stores a new macro by recording each keyboard and keypad key that you press until you use the X function or the S function to end the macro.

The S function ends the macro, but the keypad editor does not execute the macro until you use the X function.

The X function executes the macro. If you use the X function while you are recording a macro, the function ends the macro and executes it immediately.

The following sections describe in detail the macro command, the related functions, and several full examples of macros that you may find useful.

5.3.1 Recording a Macro: The LEARN Command

You can use the LEARN command to record a combination of functions and commands as a keypad editor macro at any time during a keypad editor session. After you enter the LEARN command, the keypad editor erases your last macro and then records the next functions and commands you use and the keyboard keys you type until you use the S or X functions.

The general form of the command to store a sequence of functions and commands is:

```
LEARN
```

The only functions and commands that you cannot include within a macro are:

1. The S function.
2. The X function.
3. The LEARN Command.

The keypad editor uses a special macro buffer which can hold up to 127 characters to record a macro. Each keyboard key generates one character and each arrow key and keypad key generates three characters. When you are recording a macro and you press a key that exceeds the capacity of the macro buffer or use a function that the keypad editor cannot execute, the keypad editor signals you that it is erasing all of the commands and functions that have been recorded. However, since the keypad editor also executes and displays the effect of each successful command and function as you use it, you can always identify the function or command that filled the macro buffer. The macro buffer is large enough for most macros, and therefore you may never have to consider the number of bytes that a function or command requires. However, the following suggestions will help you to reduce the size of any macro you want to use:

1. Use the abbreviated forms of commands and omit any optional command words. Each character in a command requires one byte in the macro buffer, as summarized in Table 5–1 below.

2. Specify a search model and the keypad editor settings, such as screen width and page definitions, before entering the LEARN command. Within the macro, use the FINDNEXT function instead of the FIND function whenever possible.
3. If the macro is to insert characters that you type, store them in the paste buffer before you enter the LEARN command, and then use the PASTE function within the macro.
4. Use the ADVANCE and BACKUP functions within a macro only when they are needed.
5. Use the special GOLD sequence for repeating three or more consecutive functions. For example, the sequence GOLD 3 Leftarrow requires less macro buffer space than the sequence Leftarrow Leftarrow Leftarrow.

Table 5–1 summarizes the number of bytes each function and command requires when you use it within a keypad editor macro. For each command, the table shows the number of bytes required for the shortest valid abbreviation.

Table 5–1: Macro Buffer Capacity for the Keypad Editor Commands and Functions

Commands and Functions	Bytes Required in Buffer
Arrow functions	3
Characters on the main keyboard, except as noted below	1
Commands	9 plus 1 byte per character typed
GOLD *integer*	3 for GOLD plus 1 byte per digit in the integer
Keypad functions	3; therefore, the lower function on each key requires a total of 6 bytes

Section 5.3.4 describes several typical macros in detail.

5.3.2 Terminating a Keypad Editor Macro: The S Function

VT100 Keys: The PF1 key on the keypad and the S key on the keyboard.
VT52 Keys: The blue key on the keypad and the S key on the keyboard.

Use the S function to end a macro without immediately executing the macro. After you use the function, you can use any keypad editor commands or functions. However, the LEARN command will erase the macro you have just recorded.

Examples: See Section 5.3.4.

Errors: The S function is valid only when you are recording a macro.

5.3.3 Executing a Keypad Editor Macro: The X Function

VT100 Keys: The PF1 key on the keypad and the X key on the keyboard.
VT52 Keys: The blue key on the keypad and the X key on the keyboard.

Use the X function to execute the latest macro that you have recorded. If you use the function while you are recording a new macro, the keypad editor terminates the macro and executes it immediately.

Examples: See Section 5.3.4.

Errors: The X function is valid only when the macro buffer contains a macro.

5.3.4 Examples of Keypad Editor Macros

This section lists several typical keypad editor macros that you may find useful or that you may want to modify for the special work that you do. Each example includes several special documentation conventions that you should be familiar with.

Each example has three parts:

1. The commands and functions that you must use before entering the LEARN command. For example, when you plan to use the FINDNEXT function within a macro, in most cases you will want to specify a search model before you begin to define your macro. To do that, you would use the FIND function before you enter the LEARN command.
2. The LEARN command and the combination of commands and functions that you want the keypad editor to store as a macro.
3. The commands and functions that would normally follow the S or X function that you use to end the macro.

Each example is described as a series of steps, with at least one command or function in each step. The descriptions use the following special symbols and conventions:

1. The GOLD function is printed only in the following cases:

   - When the GOLD integer sequence is used to repeat a function.
   - When the SPECINS function is used.

   For example, the string (ADVANCE)(GOLD)3(WORD) means "use the ADVANCE function and the GOLD sequence to repeat the WORD function three times." The string (GOLD)27(SPECINS) means "use the SPECINS function to insert the Escape character."

2. Strings of letters, digits, and other characters that are typed on the keyboard are not enclosed. For example, the string (FIND)09123(ADVANCE) means "use the FIND function to search forward for the string 09123." The string (COMMAND)OPEN OUTPUT REPORT.103(ENTER) means "use the OPEN OUTPUT command to open an auxiliary output file named REPORT.103."

5.3.4.1 Removing the Text in REM Statements from a BASIC-11 Program — You can decrease the size of a BASIC-11 program by removing the comments that are in the program. One recommendation is to remove only the text, however, because transfer statements, such as the GO TO statement, may specify the line number of a REM statement. You can remove only the text of a series of REM statements by executing the following sequence of keypad editor commands and functions. The example assumes that the BASIC-11 program does not contain compound statements in the following form:

```
10 REM This is a comment \ PRINT X$
```

- To set the keypad editor for exact search matching, use
  COMMAND SET SEARCH EXACT ENTER

- To specify that the keypad editor is to place the cursor after the target in a successful search, use
  COMMAND SET SEARCH END ENTER

- To specify the string REM as the search model, use
  FIND REM ADVANCE

- To start storing the combination of commands, use
  COMMAND LEARN ENTER

- To erase the text of each REM statement but leave the line number and the keyword REM, use
  DELEOL FINDNEXT

- To end the combination without executing it, use the S function.

- To execute the combination, use the X function. The combination ADVANCE GOLD X removes remarks that are to the right of the cursor and below it, and the combination BACKUP GOLD X removes remarks that are to the left of the cursor and above it.

5.3.4.2 Extracting Addresses That Contain a Given ZIP Code — When you have a mailing list with a systematic format for each address, you may be able to use a set of the keypad editor functions and commands to extract the addresses that contain a certain ZIP code. The following sequence of commands and functions assumes that the mailing list has two features in particular:

- A blank line before the first line of each address.
- No five-digit number except the ZIP code in any address.

The fact that a blank line precedes each address means that each address is preceded by two consecutive New Line terminators. The following macro uses this fact to guarantee that an address with any number of lines can be extracted.

- To define a page in terms of a marker string that has two New Line terminators, use
  COMMAND SET PAGE " RET RET " ENTER
- To open the auxiliary output file named ADDOUT.LST for the addresses that you extract, use
  COMMAND OPEN OUTPUT ADDOUT.LST ENTER
- To specify the model ZIP code 99999, use one of the following:
  FIND 99999 ADVANCE
  FIND 99999 BACKUP
- To start recording a combination of commands and functions, use
  COMMAND LEARN ENTER
- To find the next address that contains the ZIP code you have specified, use
  ADVANCE FINDNEXT
- To move the cursor back to the beginning of the address, use
  BACKUP PAGE
- To write the address to the auxiliary output file, use
  COMMAND WRITE 1 PAGE ENTER
- To end the combination without immediately executing it, use the S function.
- To execute the macro, use the X function.

You can use this macro to extract addresses for more than one ZIP code by using the FIND function again to specify a different model ZIP code.

5.3.4.3 Moving the Cursor to the End of a Word — The WORD function moves the cursor only to the first character in a word. However, in some cases you may want to change the endings of several different words. For example, you may want to change words from the singular to the plural.

In normal text, most of the words that the keypad editor detects end with one or more spaces, and you can record a macro that uses this fact to move the cursor directly to the first space that follows the printing characters in a word. One example of such a macro follows:

- To specify a space as a search model, use
  FIND SP ADVANCE
- To start storing a combination of commands and functions, use
  COMMAND LEARN ENTER

- To move the cursor to the beginning of a word and then to the first space in the word, use
  (WORD) (FINDNEXT)
- To end the sequence and execute it immediately, use the X function.

This macro moves the cursor to the end of a word only when the word has trailing spaces. The macro skips other words entirely. For example, the macro skips words that end with line terminators and Horizontal Tab characters.

5.3.4.4 Erasing Spaces from the End of a Text Line — When you create a file from card images, each line in the file may have several spaces immediately preceding the line terminator. The following sequence assumes that the New Line terminator is at the end of each line and erases any spaces that are between the last printing character and the terminator.

- To start storing a sequence of commands and functions, use
  (COMMAND) LEARN (ENTER)
- To search for a text line that has one or more trailing spaces, use
  (FIND) (SP) (RET) (ADVANCE)
- To move the cursor to the first trailing space in the line, use
  (BACKUP) (WORD) (FIND) (SP) (ADVANCE)
- To erase all trailing spaces, use
  (DELEOL)
- To end the sequence and execute it immediately, use the X function.

5.3.4.5 Reformatting Paragraphs: The FILL Function or Command — Section 5.2.5 describes how to use the VT52 FILL function and the VT100 FILL command to reformat the text lines in a select range. When the keypad editor finishes processing the select range, no new text line reaches beyond the right margin you are using.

When you want to reformat the text lines that are in separate paragraphs, you need to create a select range for each paragraph and then use the FILL command or function. The following example describes a macro to reformat paragraphs in a file when the paragraphs are separated by two consecutive New Line terminators that are followed by five spaces at the beginning of the first line. The macro assumes that the user will advance, not back up, through the file from paragraph to paragraph.

- To specify two consecutive New Line terminators as the search model, use
  (FIND) (RET) (RET) (ADVANCE)
- To start recording the combination of commands and functions, use
  (COMMAND) LEARN (ENTER)
- To move the cursor to the first printing character in the paragraph, use
  (GOLD) 7 (Rightarrow)

- To build a select range that includes the text lines that are in the paragraph, use
  (SELECT) (FINDNEXT)
- To reformat the text lines in the paragraph, use one of the following:
  With a VT100 terminal — (FILL)
  With a VT52 terminal — (COMMAND) FILL (ENTER)
- To end the combination without executing it, use the S function.
- To execute the combination, use the X function.

5.4 Reordering MACRO–11 Local Symbols: The LOCAL Command

DIGITAL's standards for MACRO–11 programs specify that local symbols within each local symbol block (LSB) start with 10$ and have an increment of 10. The keypad editor LOCAL command automatically reorders local symbols to satisfy the DIGITAL recommendations or the different standards that you specify.

The new local symbols that the keypad editor generates start with the symbol 10$ by default or with the starting value that you specify as part of the LOCAL command. After the first local symbol, the keypad editor generates other local symbols with a default increment of 10 or with the increment that you specify as part of the command.

For full descriptions of local symbol blocks, refer to the *MACRO–11 Reference Manual*.

The full form of the command to reorder MACRO–11 local symbols is:

```
LOCAL[starting value[increment]]
```

The starting value and increment that you specify must be integers in the range 1–65535.

The keypad editor restricts your use of the local symbol reordering feature in the following ways.

1. If the LSB is divided into pages by Formfeed characters, the cursor must be located at or above the first Formfeed within the LSB when you enter the LOCAL command. If the cursor is not at a valid location, the keypad editor displays the message `Valid start of LSB not found.`
2. The keypad editor detects and changes only the local symbols that are defined within the LSB in the following form:

   <Line terminator><Zero or more separators><Integer><$><:>

The separators that the keypad editor accepts are the space and the Horizontal Tab. The keypad editor changes all symbols whose definitions are in this form and also all of the references to those symbols that it can detect, including references that are within comments. The definition of a local symbol reference that the keypad editor uses is:

<non-Radix-50 character><integer><$>

3. The keypad editor detects only the LSBs that are defined as follows:

   - If an LSB begins with the .ENABL LSB directive and ends with either the .DSABL LSB directive or another .ENABL LSB directive, the keypad editor reorders all of the local symbols that are properly defined with the LSB.
   - If an LSB is not defined with an initial .ENABL LSB directive, the keypad editor reorders the local symbols that are properly defined between a non-local symbol definition and the next occurrence of one of the following:

     A non-local symbol definition.

     A .ENABL LSB directive.

     A .DSABL LSB directive.

MACRO–11 accepts other LSB definitions, but the keypad editor cannot detect them. For example, the keypad editor cannot detect the .PSECT directive as an LSB terminator, and therefore the keypad editor will continue to change local symbols below the .PSECT directive until it detects one of the features described above.

You can combine LSBs that use the same local symbols by first reordering each LSB with a local symbol starting value that does not overlap the other LSBs, then combining the LSBs, and finally reordering the combination. For example, to reorder the local symbols within two LSBs, use a procedure such as the following:

1. Reorder one of the LSBs with the LOCAL 1000 command.
2. Combine the LSBs by using the CUT and PASTE functions.
3. Reorder the combined LSB with the LOCAL 10 command.

5.5 Features for Editing Programs in Structured Languages

The keypad editor's structured tab features provide programmers who use structured languages with convenient ways to indent statements by different amounts. The features are especially useful when you want to show the nesting level of programming structures and statements.

The SET TABS command sets the keypad editor for the basic amount of indentation that you want the keypad editor to insert when you type the Tab key under certain conditions. After you set the basic indentation, use the keypad editor's three keyboard functions for structured tabs to align, extend, and decrease the total amount of indentation that the keypad editor is to generate each time you press the Tab key. The three functions and a summary of their effects are:

1. A (the alternate function of the A key) — Sets the total indentation to align with the column number of the cursor's current location.
2. D (the alternate function of the D key) — Decreases the total indentation by the basic amount that you specified in your last SET TABS command.
3. E (the alternate function of the E key) — Extends the total indentation by the basic amount that you specified in your last TABS command.

The SET NOTABS command disables the structured tab features.

The TABS ADJUST command adjusts the indentation of each line in a select range or optimizes the number of Tab characters and spaces at the beginning of a line, depending on the argument that you specify.

5.5.1 Enabling Structured Tabs: The SET TABS Command

The general form of the command to enable structured tabs is:

```
SET TABS[ indent]
```

indent is the basic amount of indentation you specify and must be an integer in the range 1–32767. When you omit an *indent* value, the keypad editor uses the default value 4.

When you specify an *indent* value, the keypad editor uses the value to calculate the total amount of indentation you want. The structured tab functions A, D, and E adjust an internal counter that is called the *level counter*. The keypad editor calculates the total indentation to create when you insert a Horizontal Tab character as follows:

*Total indent = level cntr * basic indent*

After you enable structured tabs and press the Tab key, the keypad editor provides the total indentation unless there are spaces or Horizontal Tab characters to the left of the cursor in the cursor's current line. If a space or a Horizontal Tab character is to the left of the cursor, the keypad editor responds to the Tab key as if structured tabs were disabled.

The keypad editor provides the total indentation by first inserting a Horizontal Tab character for each set of eight spaces in the total indentation and then by inserting spaces as required to match the total indentation.

Each time you enter a SET TABS command, the keypad editor resets the level counter to 1. The D function decreases the level counter by 1. The E function increases the level counter by 1. When the column number of the cursor's current location is evenly divisible by the basic indentation value, the A function sets the level counter equal to the quotient of the division.

5.5.2 Disabling Structured Tabs: The SET NOTABS Command

The general form of the command to disable structured tabs is:

```
SET NOTABS
```

The command disables all structured tab features, including the TABS ADJUST command. After the SET NOTABS command, the keypad editor processes the Tab character in the normal way, as a single ASCII character.

5.5.3 Aligning Structured Tabs to the Cursor: The A Function

VT100 Keys: The PF1 key on the keypad and the A key on the keyboard.
VT52 Keys: The blue key on the keypad and the A key on the keyboard.

The A function is disabled until you enable structured tabs by entering the SET TABS command. When structured tabs are enabled, the A function causes the keypad editor to calculate a value for the level counter from the column number of cursor's current location. The function is especially useful when you want to insert new programming statements within a group of statements that are already indented.

The keypad editor uses the following procedure to calculate the value for the level counter:

1. The keypad editor divides the column number of the cursor's current location by the basic indentation value that you specified with your last SET TABS command.

   For example, with a basic indentation value of 5 and the cursor at column 21, type the A function. The keypad editor calculates the level counter as 4. If you then move the cursor to the beginning of a line and press the Tab key, the keypad editor inserts two Horizontal Tab characters and four spaces to create the total indentation of 20. The keypad editor then leaves the cursor at column 21, ready for you to insert material.

2. When the cursor is not at column 1 and the column number is not evenly divisible by the basic indentation that you have specified, the keypad editor signals an error and ignores the new value for the level counter. In this case, an existing level counter value is not changed. If you use the HELP function in this case, the keypad editor displays the message:

   ```
   TAB Indent value wrong for align
   ```

For example, with a basic indentation value of 6 and with the cursor at column 11, the keypad editor calculates a level counter that is between 1 and 2, ignores the calculated value, and retains the previous level counter value.

Example: See Section 5.5.7.

5.5.4 Decreasing the Level Counter: The D Function

VT100 Keys: The PF1 key on the keypad and the D key on the keyboard.
VT52 Keys: The blue key on the keypad and the D key on the keyboard.

The D function is disabled until you enable structured tabs by entering the SET TABS command. When structured tabs are enabled, use the D function to subtract 1 from the level counter. One common use for the function is to decrease the total indentation at the beginning of a statement line so that your program format reflects the nesting level of structures within the program.

Example: See Section 5.5.7.

5.5.5 Extending the Level Counter: The E Function

VT100 Keys: The PF1 key on the keypad and the E key on the keyboard.
VT52 Keys: The blue key on the keypad and the E key on the keyboard.

The E function is disabled until you enable structured tabs by entering the SET TABS command. When structured tabs are enabled, use the E function to add 1 to the level counter. One common use for the function is to increase the total indentation at the beginning of a statement line so that your program format reflects the nesting level of structures within the program.

Example: See Section 5.5.7.

5.5.6 Adjusting Structured Tabs: The TABS ADJUST Command

With structured tabs enabled, the TABS ADJUST command allows you to:

1. Adjust the indentation of a multiline segment of a file.
2. Reduce to a minimum the number of spaces and tab characters at the beginning of indented lines.

Therefore, if you increase or decrease the logical nesting level of a part of a program, you can with one command also increase or decrease the amount of indentation at the beginning of a program line. And if the size of your program is critical, you can ensure that the indentations are taking as little space as possible.

The full form of the command to adjust structured tabs is:

```
[TABS ]ADJUST [[{±} ] number-of-levels]
```

The number of levels must be an integer in the range -50 to +50 and determines the amount that the current indentation is to be changed. Structured tabs must be enabled and the segment of your file that you want to adjust must be in a select range.

If the number of levels is zero or if you omit the value, the editor processes the TABS ADJUST command as follows:

1. Within the select range, the editor finds each text line that begins with a string of spaces or tab characters.

2. The editor then retains the text line's indentation but replaces the string with the most efficient combination of spaces and tab characters.

If the number of levels is less than or greater than zero, the keypad editor processes the TABS ADJUST command as follows:

1. The editor finds the first complete text line in the select range. Then the editor completes the following steps for each text line that is in the select range.

2. Within the text line, the editor finds the end of the first occurrence of:

   - A string of spaces.
   - A string of tab characters.
   - A combination of spaces and tab characters.

3. With cursor at the end of the string of spaces or tabs, the editor calculates the level counter for the cursor's position. The process is the same as for the A function that aligns the level counter to the cursor's position.

4. The editor adds or subtracts the number of levels that you specified in the TABS ADJUST command.

5. The editor adjusts the text line to the proper indentation and finds the beginning of the next text line in the select range.

At the end of the process, the cursor is located at the end of the select range.

When you adjust the indentation of a select range, the editor cannot maintain the position of comments that are to the right of executable statements in the select range. However, if the current increment with which you enabled structured tabs is a multiple of 8 and the comments are preceded by Tab characters, the comments will remain aligned when you adjust the indentation.

The TABS ADJUST command is invalid when you are inspecting a file.

5.5.7 Examples of Structured Tab Features

This section provides examples of the keypad editor structured tab features in the context of a short section of a FORTRAN program. By creating a new file and following the instructions for each step, you can see the processes that take place.

1. Enable structured tabs and set the basic indentation to four spaces by entering the SET TABS 4 command. Use the following sequence:

   GOLD COMMAND SET TABS 4 ENTER

2. With the cursor at column 1, press the Tab key where the TAB symbol appears and type the following FORTRAN statement:

   TAB DO 100 NUM = START , FINISH RET

   The keypad editor indents the statement to column 5 with 4 spaces because the basic indentation is 4 and the level counter is 1. The indentation is created with spaces because it is not large enough to permit any Tab characters to be used.

3. Increase the level counter to 2 by using the E function. Use the following sequence:

   GOLD E

4. With the cursor at column 1, press the Tab key and type the statements as shown (the programming error with the SUM = 0 statement will be corrected in a later step):

   TAB SUM = 0 RET
   TAB SUM = SUM + ARRAY (NUM) RET
   100 TAB CONTINUE RET

   The keypad editor indents the first two statements to column 9 with a Tab character (equivalent to 8 spaces) because the level counter is now 2. Each indentation is created with a Tab character because the keypad editor calculates that the Tab character will have the same effect as 8 spaces.

5. Decrease the indentation by using the D function. Use the following sequence:

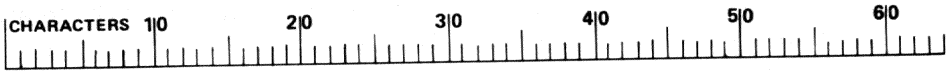
   GOLD D

6. With the cursor at column 1, press the Tab key and type the following statements as shown:

   TAB TYPE * , SUM
   TAB END

   The keypad editor indents the statements to column 5 by inserting 4 spaces because the level counter is now 1.

7. Your FORTRAN statements now look like the following illustration.



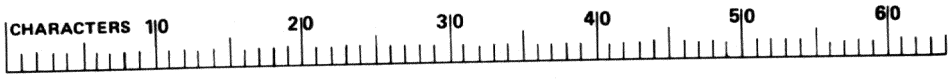

```
      DO 100 NUM = START, FINISH
          SUM = 0
          SUM = SUM + ARRAY (NUM)
100       CONTINUE
      TYPE *, SUM
      END
```

To indent the six statements 4 additional spaces, put the statements into a select range and enter the TABS ADJUST +1 command. With the cursor at column 1 of the DO statement, use the following sequence:

SELECT BOTTOM COMMAND TABS ADJUST +1 ENTER

The keypad editor increases each indentation by 4 columns by calculating and inserting the most efficient combination of Tab characters and spaces for each statement.

8. The SUM = 0 statement is inside the DO loop by mistake. Fix the error by moving the statement above the DO statement. The six statements should look like the following illustration:




```
          SUM = 0
      DO 100 NUM = START, FINISH
          SUM = SUM + ARRAY (NUM)
100       CONTINUE
      TYPE *, SUM
      END
```

9. Decrease the indentation of the SUM = 0 statement. Put the statement in a select range and enter the TABS ADJUST -1 command. With the cursor at column 1 of the SUM = 0 statement, use the following sequence:

SELECT ADVANCE BLINE COMMAND TABS ADJUST -1 ENTER

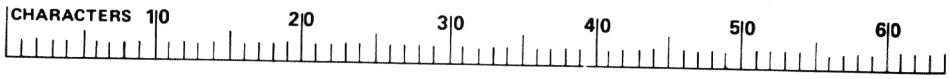
10. To insert a statement in the DO loop and align it with the other statements that are in the loop, use the A function to set the level counter properly. With the cursor on the C of the word CONTINUE, use the following sequence:

   GOLD A

11. Move the cursor to the beginning of the 100 CONTINUE statement and create an open line for the new statement by using the following sequence:

   BACKUP BLINE OPENLINE

   The six statements should look like the following illustration:




```
      SUM = 0
      DO 100 NUM = START, FINISH
          SUM = SUM + ARRAY (NUM)

100       CONTINUE
      TYPE *, SUM
      END
```

12. Type the following statement.

   TAB IF (ARRAY(NUM) .LT. 0) TYPE *,'Low value at ',NUM

   The keypad editor indents the statement to column 13. The total indentation is the equivalent of 12 spaces (3 times 4) because the A function set the level counter to 3 and the basic indentation is 4.

Appendix
PDP–11 Keypad Editor Messages

This appendix lists all of the messages that the keypad editor can display. The listing is arranged according to the first letter and later characters in each message. The prompts and informative messages the keypad editor displays, such as the `Command:` prompt and the `WORKING...` message, are also listed.

Each error message is documented in the following manner:

1. The error message appears in boldface.
2. Beneath the message are two columns. The left-hand column describes the conditions that caused the message to be generated. The right-hand column contains information about correcting the error condition.

The keypad editor displays messages in three general formats:

1. Command level messages in the form

   ?KED-Severity Code-Text

2. HELP function messages centered on the bottom screen line.
3. Prompts and informative messages.

Sections A.1, A.2, and A.3 summarize the ways the keypad editor uses the Command String Interpreter and HELP function messages. Section A.4 explains the procedure to follow if one of the keypad editor's messages about nonrecoverable conditions appear. Section A.5 provides suggestions that apply to any of the keypad editor's messages that an output file is full. Section A.6 is the full listing of messages.

A.1 Messages at the Command String Level

Command level messages have the following general form:

?KED–Severity Code-Text

KED and K52 both use the same messages. The messages usually appear if your response to the keypad editor prompt contains or causes an error. The two severity codes that you may see and their meanings are:

Severity Code	Meaning
F	A fatal error; the keypad editor returns either to the system level or to the keypad editor prompt.
W	A warning; the keypad editor continues executing unless you choose to stop it.

A.2 HELP Function Messages

When the keypad editor is displaying a file on the terminal screen, it signals whenever it cannot process a function or command in the normal way. The keypad editor's normal signal is to ring the bell on the terminal. With a VT100 you can also set the keypad editor for quiet signalling by using the `SET QUIET` command. The command is explained in Chapter 3.

When the keypad editor signals you in either way, you may be able to see immediately the cause of the signal by inspecting the cursor's position or the changes that have been displayed on your screen. You can also see a one-line message that explains the signal by using the HELP function. The function is explained in Chapter 3.

A.3 Prompts and Informative Messages

When the keypad editor is displaying a file on the terminal screen, it also displays prompts and informative messages by displaying text on the first line. The keypad editor automatically repaints the screen when it completes the process that is related to the prompt or informative message. For example, when you move the cursor the full length of a very long file, the keypad editor may temporarily display the message `WORKING...` on the top screen line. When the cursor reaches its target, the keypad editor removes the message and repaints the screen.

A.4 Messages for Nonrecoverable Conditions

Some of the messages displayed by the keypad editor identify nonrecoverable conditions. Digital Equipment Corporation uses these messages to diagnose the keypad editor malfunctions. Whenever a nonrecoverable condition arises, use the following procedure.

1. As accurately as possible, write down the functions and commands you used before the message appeared.
2. Save the programs or files you were editing.
3. Obtain a new copy of the keypad editor from your distribution kit and try to duplicate the functions and commands that caused the message.

4. If the message reoccurs, check your hardware or find someone to check it for you.
5. If the problem persists, consult someone in your area who is very familiar with the keypad editor.
6. If you qualify to receive a written reply under DIGITAL's Software Performance Report (SPR) service, follow the directions on the SPR form.

A.5 Messages When an Output File Is Full on RT–11

With RT–11 the keypad editor can display several messages when the file you are editing or creating is too large to allow more insertions during the current editing session. This section includes suggestions that may be useful when this condition arises.

The keypad editor uses the normal RT–11 procedures when it opens an output file. The output file is opened first to obtain one-half of the largest block of free space on the output volume, or the second largest block of free space, whichever is larger. After that step, if the output file space is not large enough for the input file you specify, the keypad editor uses the largest block of space available on the output volume.

In almost all cases the output file will be large enough for your input file and insertions. However, you may exceed its capacity, especially if you add the contents of several auxiliary input files to the main input file. If that should occur, try one of the following suggestions.

1. Exit and Start a New Session

   When you use the `EXIT` command, the keypad editor saves the output file and prompts you for another CSI file specification string. If you then begin another session and edit the file you just created, the keypad editor opens a new output file. In most cases, the new output file will be large enough to hold the additions you want to make.

   This suggestion may not be effective if the output volume you are using is almost full or if free space on it is not consolidated. (Use the RT–11 `SQUEEZE` command to consolidate free space.) The principal disadvantage of this suggestion is that the `EXIT` command causes the keypad editor to close all open files, discard the contents of the paste, line, word, and character buffers, and restore any special settings you made to it's default values.

2. Erase and Make Room for Insertions

   When the output file is full and valuable material is in the paste buffer or other buffers, you can make room for the valuable material by erasing part of your file. Each character you erase makes room for one additional character during the current session.

3. Use an Auxiliary Output File

When the output file is full, you can split it into two files. Use the OPEN OUTPUT command to open an auxiliary output file and the WRITE command to store as much of the output file as necessary. When the keypad editor completes the WRITE command, use the CLOSE command to save the auxiliary output file and then erase the section of the output file that you stored elsewhere.

In most cases, you can recombine the parts of the original file in another keypad editor session. If free space is still not sufficient, use a new output volume.

A.6 When a Function Seems to Insert Strange Characters

One hardware feature of your terminal is called the character silo. The silo holds characters that you type while waiting for your system to pass them to the keypad editor. When you use the VT100 autorepeat feature or the VT52 Repeat key to repeat a single-key function, you may overfill the terminal silo. When that occurs, the terminal signals you with the terminal bell (and no keypad editor message is available because the keypad editor has not caused the signal).

In this case, characters that you have not typed may appear on your screen. The question mark (?) and lowercase or uppercase letters are the most commonly reported insertions. The reason the characters are inserted is that the overfilled silo has disrupted the stream of characters being sent to the keypad editor, and the keypad editor interprets some of the incoming characters as insertions, not as editing functions.

To avoid this problem, avoid unusually long bursts of automatic repeating. To correct the problem, you can either erase the unwanted characters by using the keypad functions for erasing or you can stop editing without creating an output file by using the QUIT command.

A.7 PDP–11 Keypad Editor Messages

ADJUST illegal in inspect mode	The TABS ADJUST command is invalid when you are inspecting a file.	Use another command or function.
Advance char finds end of file	The CHAR function or the rightarrow function moved the cursor to the bottom of the file.	Use any valid function or command.
Advance line finds end of file	The BLINE function has moved the cursor to the end of the file. With line count definitions, the PAGE and SECTION functions can also cause this message.	Use another function.

Advance word finds end of file

The WORD function moved the cursor to the bottom of the file or the DEL-WORD function erased the last character in the file.

Use any valid function or command.

Argument error to INCLUDE or SKIP

The keypad editor did not complete an INCLUDE or SKIP command because you specified a negative line or page count.

Use any valid function or command.

Argument error to WRITE

The keypad editor did not complete a WRITE command because you specified a negative line or page count.

Use any valid function or command.

Arrow command finds extremity of file

The uparrow function moved the cursor to the top of the file, or the downarrow function moved the cursor to the bottom of the file.

Use any valid function or command.

Auxiliary input file contains records that will be truncated

The keypad editor loads this message for your information when you answer "no" to the following message.

Use any valid function or command.

Auxiliary input file contains records that will be truncated — Continue (Y,N)?

The largest record in the auxiliary input file that you have specified is larger than the keypad editor can read entirely.

To have the keypad editor continue, type Y and press the Return key. The overlong record will be truncated when the keypad editor reads it, and the truncated data will be lost. To stop the keypad editor, type N and press the Return key.

Auxiliary input file not open

The keypad editor did not complete an INCLUDE command because no auxiliary input file is open. The keypad editor automatically closes an auxiliary input file when an INCLUDE or SKIP command has processed the entire file.

To open an auxiliary input file, use the OPEN INPUT command. Otherwise, use any valid function or command.

Auxiliary input illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

Auxiliary output file exists

The keypad editor loads this message for your information when you answer "no" to the following message.

Use any valid function or command.

Auxiliary output file exists — Replace (Y,N) ?

The auxiliary output file name you specified is the same as an existing file on the output volume.

To erase the existing file and create a new one with the same name, type Y and press the Return key. To preserve the existing file, type any other answer.

Auxiliary output file full

No more space remains in the auxiliary output file.

Use the CLOSE command to close the auxiliary output file. Open another file for auxiliary output and merge the files later in a separate editing session or with the RT-11 COPY command.

Auxiliary output file not open

The keypad editor did not complete a WRITE command because no auxiliary output file is open. The CLOSE and EXIT commands close an auxiliary output file. The PURGE and QUIT commands discard an auxiliary output file, if one is open.

To open an auxiliary output file, use the OPEN OUTPUT command. Otherwise, use any valid function or command.

Backup char finds beginning of file

The CHAR function or leftarrow function moved the cursor to the top of the file.

Use any valid function or command.

Backup line finds beginning of file

The BLINE function has moved the cursor to the top of the file. With line count definitions, the PAGE and SECTION functions can also cause this message.

Use another function.

Backup word finds beginning of file

The WORD function or LINEFEED function moved the cursor to the top of the file.

Use any valid function or command.

Bounded search reached bound

You have used the SET SEARCH BOUNDED command to limit searches to one page. Then a FIND or FINDNEXT function moved the cursor to the top or bottom of a page without finding a string that matches the model you have specified.

Use any valid function or command.

CHNGCASE finds end of file

In the ADVANCE directional mode, the CHNGCASE function changed the case of the last character in the file.

Use another function or command.

CHNGCASE is illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

Command canceled

The keypad editor loads this message when you type CTRL/Z or CTRL/C to cancel a keypad editor command and the Command: prompt.

Use any valid function or command.

Command:

The keypad editor displays this prompt after you use the COMMAND function.

Type the command you want the keypad editor to execute and terminate the command with the ENTER function.

CTRL/C entered to stop operation

The keypad editor loads this message when you type CTRL/C CTRL/C to cancel the command or function that the keypad editor is processing. The keypad editor repaints your terminal screen and shows where the cursor moved. If you typed commands or functions while the keypad editor was displaying the `WORKING...` message, the keypad editor ignores all of them.

Use any valid function or command.

CTRL/C or CTRL/Z ignored — use QUIT

While you are working with a file, the keypad editor ignores the CTRL/Z and CTRL/C combinations.

To insert CTRL/Z or CTRL/C in your file, use the SPECINS function. To cancel a keypad editor session without saving any open output files, use the `QUIT` command.

CTRL/U finds beginning of file

The CTRL/U function erased the first character in the file.

Use any valid function or command.

Cursor not at target

With no existing select range, the keypad editor could not complete a CUT, APPEND, PASTE, or SUBSTITUTE function because the cursor is not at a valid search target.

Use any valid function or command.

DELCHAR finds end of file

The DELCHAR function erased the last character in the file.

Use any valid function or command.

DELEOL finds end of file

The DELEOL function erased the last character in the file.

Use any valid function or command.

DELETE finds beginning of file

The DELETE function erased the first character in the file.

Use any valid function or command.

DELLINE finds end of file

The DELLINE function erased the last character in the file.

Use any valid function or command.

Empty select range specified to CHNGCASE

After using the SELECT function, you used the CHNGCASE function before building a select range.

To change the case of several letters, build a select range that includes them and then use the CHNGCASE function.

End of file reached on auxiliary input file

A `SKIP` or an `INCLUDE` command finished reading to the bottom of an auxiliary input file and the keypad editor automatically closed the file.

Use any valid function or command.

EOL finds beginning of file

The EOL function moved the cursor to the top of the file.

Use any valid function or command.

EOL finds end of file

The EOL function moved the cursor to the bottom of the file.

Use any valid function or command.

Erasures are illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

Error reading auxiliary input file

The keypad editor could not read the auxiliary input device. The problem is not due to reaching the end of the file.

If the file name was correct, check that the device is on-line and that it contains the proper volume. Use another device if the device is faulty.

Error writing auxiliary output file

The keypad editor may have detected a bad block or faulty device.

Check that the device is on-line, write-enabled, and properly formatted. Try using another auxiliary output device or volume. Use the RT-11 DIRECTORY/BADBLOCKS command to check for bad blocks.

EXIT stopped by CTRL/C

The keypad editor loads this message when you type CTRL/C to cancel the EXIT command. The results are unpredictable because they depend on several variable factors, such as the output file size, the speed of the hardware components you are using, and when you type CTRL/C during the EXIT process.

If you type CTRL/C CTRL/C too late to cancel the EXIT process, the keypad editor displays the command level prompt again. In this case, the output files you originally specified have been saved. If you also specified automatic backup of input files, they have also been renamed. If CTRL/C CTRL/C cancels the EXIT process, the keypad editor repaints the terminal screen with the part of the file that contains the cursor. In this case, continue editing by using any valid function or command.

File full during FILL

The keypad editor could not complete the FILL command or function because the output file or temporary file is full.

Refer to Section A.5.

File full during INCLUDE

The keypad editor could not complete the INCLUDE command because the output file is not large enough for the material you want to copy. The cursor is to the right of the last character the keypad editor copied from the auxiliary input file.

Refer to Section A.5 of this appendix for suggestions that may be useful when your file is full. You can also use the CLOSE or QUIT commands or any functions or commands that erase material from the file.

FILL illegal during inspect

The FILL command or function is invalid while you are inspecting a file.

Use another function or command.

I/O error closing auxiliary output file

The keypad editor could not write to the auxiliary output file during a close operation. The keypad editor may have detected a bad block.

Try using another auxiliary output volume. Use the RT-11 DIRECTORY/BADBLOCKS command to check for bad blocks.

Illegal command

The keypad editor could not recognize the command you typed.

Use any valid function or command.

Illegal definition of PAGE or SECTION

The keypad editor could not process a SET ENTITY PAGE or SET ENTITY SECTION command for one of the following reasons:

- the string marker was not enclosed in matching single quotes or double quotes.
- the string marker was enclosed in illegal delimiters.
- in a line count definition, you did not specify a value greater than zero.

Use any valid function or command.

Illegal file specification

The file specification you typed in an OPEN INPUT or OPEN OUTPUT command was not a valid file specification.

Use any valid function or command.

Illegal function

The keypad editor could not recognize the function you specified and also could not directly insert the characters you typed. The most common cause is typing the Escape key and then almost any other key on the keyboard or the keypad.

To insert an Escape character or another character that you cannot insert directly, use the SPECINS function. You can also use any valid function or command.

Illegal right column for wrap

You specified a right margin that is greater than 256 in the SET WRAP command.

Enter the command again, or use another function or command.

Illegal tab indent value

You specified an indentation value of 0 or less in the SET TABS command.

Enter the command again, or use another function or command.

Illegal terminating key

The keypad editor could not complete a FIND function for one of the following reasons:

- you included an illegal character in the model.
- you did not terminate the model with the ADVANCE or BACKUP function.

Use any valid function or command.

Illegal terminating key to command prompt

The keypad editor could not process a command for one of the following reasons:

- you included an illegal character in the command.
- you did not terminate the command with the ENTER function.

Use any valid function or command.

Illegal to insert null

With the SPECINS function, you specified the NULL character (ASCII code 0). The keypad editor does not allow you to insert nulls in any file.

Use another command or function.

INCLUDE and SKIP illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

INCLUDE finds end of auxiliary input file

Your last INCLUDE command copied the last character of the auxiliary input file to your main file. The keypad editor closed the auxiliary input file.

Use another function or command.

Insert finds file full

The keypad editor could not insert the character you typed because the output file is not large enough.

Refer to Section A.5 of this appendix for suggestions that may be useful when your file is full. You can also use the `CLOSE` or `QUIT` command or any functions or commands that erase material from the file.

Insert failure in local

You stopped the processing of a LOCAL command by typing CTRL/C or the keypad editor could not finish processing for some other reason.

Insert is illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

Invalid parameter to ADJUST

In the TABS ADJUST command, you specified an adjustment that is outside the range of -50 to +50.

Enter the command again, or use another command or function.

Invalid parameter to LOCAL

In the LOCAL command, you specified a starting value or increment that is 0 or greater than 32767.

Enter the command again, or use another command or function.

?KED-F-Bad call to .VVV.V

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

?KED–F–Directory file does not exist

On an RSX–11 system, the keypad editor cannot access the directory file for one of the UICs in the file specification string.

Check for a typing error. Reenter the file specification string.

?KED–F–Illegal command line

?KED–F–Illegal file specification

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

?KED–F–Insufficient memory

The keypad editor may have malfunctioned or cannot run because too little memory is available. (The keypad editor requires approximately 16K words.)

RT–11:
Switch to a system that provides more memory resources or switch to a smaller RT–11 monitor. In the *RT–11 System Message Manual*, the section entitled Increasing Storage and Memory Resources offers other suggestions that may be useful. If the message continues to appear, refer to the procedure in Section A.4 of this appendix.

RSX–11 and IAS:
Ensure that the keypad editor has been properly built and installed, and that there is a large enough task size increment.

?KED–F–Invalid device

The main input or output device is not a random-access device. This restriction does not apply to auxiliary files, which may reside on sequential-access devices.

Use a random-access device for your main input and output files.

?KED–F–I/O or device error — Continue (Y,N) ?

The keypad editor detected an I/O or device error. Examples of situations that can cause this message are:

- a device you are using has been switched off-line.
- an output device you are using has been write-locked.
- a volume you are using has a bad block.

Before typing any response, check that the device is on-line, and if it is an output device, check that it is write-enabled. To make the keypad editor try the same function or command once more, type `Y` and press the RETURN key. The keypad editor then tries the function or command once more. If the command or function works, continue editing. If the command or function does not work, or if you type any character other than `Y`, the keypad editor quits without saving any of your editing work and returns to the RT–11 monitor.

?KED–F–I/O error while trimming output

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

?KED–F–Output file shorter than input file

The space available for the output file is smaller than the input file.

RT–11:
Use the RT–11 SQUEEZE command to consolidate free space on the output volume or use a different volume.

RSX–11 and IAS:
Specify the output filesize correctly or use the default. There may not be sufficient storage space on the volume.

?KED–F–RDBKW — No space to read into
?KED–F–RDFWD — No space to read
?KED–F–RDFWD — Pointer corrupt
?KED–F–SETUP — Allocation error
?KED–F–SPCFRE — Logic error, character count wrong

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

?KED–F–Temporary file shorter than input file

The number of blocks that you specified with the /BL option is less than the size of the input file that you specified.

Enter the command line again with a temporary file size that is greater than the input file size.

?KED–F–Too big to edit

The input file is too large for KED to handle. KED cannot edit or inspect a file that is larger than 32767(decimal) blocks. This restriction does not apply to auxiliary files or devices.

Split the file into smaller segments using auxiliary output files (described in Chapter 4), and edit each segment separately.

?KED–F–TTY??? Logic error

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

?KED–F–Unable to access input device

The keypad editor could not access the auxiliary device for one of the following reasons:

- the device is illegal.
- the device is not ready.
- the device is not installed.
- the volume has too many bad blocks.
- no room for the device handler.

RT–11:
Check the device abbreviation. If it is correct, check that the device is on-line. Use the RT–11 SHOW/DEVICES command to see which devices your system includes, and use the RT–11 INSTALL command to install the device you need, if necessary. If the message continues to appear, use the RT–11 DIRECTORY/BADBLOCKS command to check the volume you are using for bad blocks, and if there are many bad blocks on the volume, use another copy. If you cannot install the handler for the device you want to use, try the suggestions listed in the documentation for the message "?KED–F–Insufficient memory."

RSX–11 and IAS:
Correctly specify the input device.

?KED–F–Unable to access output device

The keypad editor probably could not load the output device handler.

If the device name was correct, check that the system volume contains the device handler. If the message continues to appear, try the suggestions for "?KED–F–Unable to access input device."

?KED–F–Unable to attach terminal

The keypad editor cannot attach your terminal and therefore cannot proceed.

Check with your system manager.

?KED–F–Unable to copy temporary file

With an RSX system and after you have used the EXIT command, the system cannot properly convert the keypad editor's temporary file. (Chapter 2 explains temporary file usage.) The system locks both the temporary file and the output file.

If you must try to salvage either the locked temporary file or the output file, use the PIP UNLOCK command which is described in the *RSX–11 Utilities Manual* or the *IAS Utilities Manual*. Or, use the PDS commands UNLOCK and SET END-OF-FILE, described in the *IAS PDS User's Guide*.

?KED–F–Unable to create output file

?KED–F–Unable to create temporary file

With RSX or IAS systems, the keypad editor cannot create the specified file on the output volume and in the directory that you have specified. Three causes for this message are:

- A protection violation.
- Too little space on the output volume.
- An I/O error.

Check that the output device is online and write-enabled. Check that your output volume contains sufficient free space as described in Chapter 2. Check that the current UIC has the correct file protection attributes.

?KED–F–Unable to edit input file

The input file that you specified is too long to be handled by the keypad editor. With the keypad editor, you cannot edit a file that is larger than 16383. blocks.

Find some other way to edit the file. If necessary, consult an experienced colleague about the problem of splitting the file into separate parts that are small enough for the keypad editor.

?KED–F–Unable to open input file

The keypad editor could not open the input file for one of the following reasons:

- the volume does not contain the file.
- the volume has not been initialized.
- the input device is not on-line.

Check the file specification. If it is correct, check that the correct volume has been installed and that the input device is on-line.

?KED–F–Unable to open output file

The keypad editor could not open the output file for one of the following reasons:

- the device is not ready or is write-locked.
- the volume has not been initialized.
- the volume does not have sufficient free space for the output file.
- the output file exists and has been protected.

Check that the output volume is properly installed and that it has been initialized. If the volume is too full, use another volume. To edit a protected file, use a different output file specification.

?KED–F–Unable to trim output file
?KED–F–WRBKW — Bad block number
?KED–F–WRBKW — Block overrun
?KED–F–WRBKW — Plygnd not empty
?KED–F–WRFWD — Bad block number
?KED–F–WRFWD — Block overrun
?KED–F–WRFWD — Not enough data

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

?KED–W–Cannot set terminal options — Continue (Y,N)?

With KED running on an RT–11 multiterminal system, the terminal KED is running on has been attached (.MTATCH) by another job. KED requires that the multiterminal equivalent of SET TERM NOCRLF be performed, but cannot do so itself.

To continue processing, type Y and press the RETURN key. To stop KED, type N and press RETURN. When KED returns to the system prompt (.), type the command SET TERM NOCRLF. Then restart KED.

?KED–W–File not found — Create it (Y,N)?

KED could not find the input file on the volume you specified.

Make sure you typed the input volume name and file name correctly. To continue, type Y and press the RETURN key. KED will create the file you specified on the volume you specified. To stop KED, type N and press RETURN.

KED–W–Input file contains records that will be truncated — Continue (Y,N)?

The largest variable length record in the input file is larger than the space that the keypad editor has available to read records. Records larger than that space will be truncated when read, and the truncated data will be lost.

To have the keypad editor continue processing, type Y and press the RETURN key. To stop the keypad editor, type N and press the RETURN key.

?KED–W–Input file not properly closed — Continue (Y,N)?

The keypad editor finds conflicting information about the input file that you specified.

To have the keypad editor continue, type Y and press the RETURN key. The keypad editor will try to read the file, but the file may contain invalid data that you may have to delete. If data at the end of the file appears to have been deleted because the end of file symbol appears earlier than you expected, the file may contain a record that is too long for one of the keypad editor's buffers.

To stop the keypad editor, type N and press the RETURN key.

?KED–W–Only MMMMMM blocks available for insertions — Continue (Y,N) ?

Free space on the output volume is less than 10 blocks larger than the input file. The keypad editor reports (in decimal) the number of blocks available.

To continue the editing session, type Y and press the RETURN key. The keypad editor will signal when the file becomes as large as possible. To cancel the editing session and return to the RT–11 command level prompt (*), type any other answer. Use the RT–11 SQUEEZE command to consolidate free space on the output volume, or specify an output volume that has more free space.

?KED–W–Output file exists — Continue (Y,N)?

The output file name you specified is the same as an existing file on the output volume.

To erase the existing file and create a new one with the same name, type Y and press the RETURN key. To preserve the existing file and return to the system or keypad editor prompt, type any other answer.

?KED–W–Output file or device protected — Inspect (Y,N)?

The output file you specified for creating or editing already exists as a protected file, the input file you specified for editing is protected, or the output device you specified is write-protected.

To continue by inspecting the file you specified, on the device you specified, type Y and press the RETURN key. To stop KED, type N and press RETURN.

?KED–W–Output files purged

The keypad editor displays this message when you use the QUIT command. If an auxiliary output file is open when you type the command, the keypad editor does not save the file. If you are editing or creating a file, the keypad editor does not save any of your editing work.

The message is for information only.

?KED–W–Unable to rename input file to BAK file type

The original input file was not saved with the BAK file type. However, the keypad editor attempted to save the output file.

The message is primarily for your information. If you do not have another copy of the original file, there is no way you can recover it.

?KED–W–Unable to truncate or close output file

With an RSX system and after you have used the EXIT command, the system cannot properly store your output file. The system locks the output file because the file may be faulty.

If you must try to salvage the locked output file, use the PIP UNLOCK command. The command is described in the *RSX–11 Utilities Manual* or the *IAS Utilities Manual*.

?KEX–F–Insufficient memory

You attempted to run KEX, but there is not enough available memory. Or, you applied the customization in Section 2.7 of the *RT–11 Installation Guide* to limit the amount of memory KEX requests with a .SETTOP, but the amount of memory you requested was too small.

Reapply the KEX customization, specifying a larger value for the highest address KEX requests with a .SETTOP.

Learn buffer filled

The functions and commands that you have used since you entered the LEARN command require too many bytes for the learn buffer. The keypad editor immediately stops storing the macro and erases the contents of the learn buffer.

Check that you are not using unnecessary ADVANCE or BACKUP functions in your macro. Use abbreviations for commands within the macro, and use the shortest possible search models within the model. Table 5–1 shows the minimum number of bytes each function and command requires.

LOCAL illegal during inspect

You entered the LOCAL command while inspecting a file.

Use another command or function.

Local symbol definition too long

In processing a LOCAL command, the keypad editor detected a local symbol that is more than 6 digits long.

Check that the local symbol is correct.

Logic error in ADJUST
Logic error in CHNGCASE
Logic error in CUT
Logic error in FILL
Logic error in PASTE
Logic error in SUBSTITUTE
Logic error in undelete
Logic error in undelete setup
Logic error in WRITE SELECT

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

Model:

The keypad editor displays this prompt after you use the FIND function.

Type the string you want the keypad editor to search for and then specify the direction of search with the ADVANCE or BACKUP function. You can specify models from 1 to 46 characters long.

Move failure in local

You stopped the processing of a LOCAL command by pressing CTRL/C, or the keypad editor could not finish processing the command for some other reason.

Move PAGE or SECTION finds extremity of file

A PAGE or SECTION function moved the cursor to the top or bottom of the file.

Use any valid function or command.

Move to bottom when at bottom

The BOTTOM function is not valid when the cursor is already at the bottom of the file.

Use any valid function or command.

Move to top when at top

The TOP function is not valid when the cursor is already at the top of the file.

Use any valid function or command.

No macro to execute

The macro buffer is empty either because you have not used the LEARN command to store a macro or because the last macro that you tried to store failed while you were defining it.

Use the LEARN command to store the macro that you want to use.

No model defined

The keypad editor could not process the FINDNEXT, REPLACE, or SUBSTITUTE function because you have not yet specified a search model.

To search for a string, use the FIND function, type a model of the string you want the keypad editor to search for, and then specify the direction of search with the ADVANCE or the BACKUP function. You can also use any other valid function or command.

No select range defined

You attempted a REPLACE, APPEND, or CUT function without previously defining a select range and the cursor was not located at a search target.

Use any valid function or command.

No select range define for FILL

There were no characters in a select range or you were not building a select range when you used the FILL function or command.

Build a select range that contain the segment of your file that you want to reformat before using the FILL command or function.

No select range for ADJUST

You are not building a select range or there are no characters in the select range that you are building.

Build a select range that includes the segment of your file that you want to adjust before using the TABS ADJUST command.

No select range for WRITE SELECT

You have used the SELECT function and then entered the WRITE SELECT command. However, there are no characters in the select range that you are building.

Build a select range that includes the material that you want to copy to the auxiliary output file, and then enter the WRITE SELECT command.

Not enough file space to do PASTE

The keypad editor could not complete the PASTE function because the output file is not large enough for the material that is in the paste buffer. The cursor is located to the right of the last character the keypad editor copied from the paste buffer.

Refer to Section A.5 of this appendix for suggestions that may be useful when your file is full. You can also use the EXIT or QUIT command or any functions or commands that erase material from the file. If you use the EXIT or QUIT command, the keypad editor preserves the material in the paste buffer for your next session until you stop the keypad editor completely and return to the system level.

Not enough space to undelete

The keypad editor could not complete an UNDELLINE, UNDELWORD, or UNDELCHAR function because the output file is not large enough for the added material. The cursor is located to the right of the last character the keypad editor copied.

Refer to Section A.5 of this appendix for suggestions that may be useful when your file is full.

Nothing to undelete

The keypad editor could not process an UNDELLINE, UNDELWORD, or UNDELCHAR function because the corresponding line, word, or character buffer is empty.

Use any valid function or command.

OPENLINE finds file full

The keypad editor could not process the OPENLINE function because the output file is not large enough for a Carriage-Return/Linefeed pair.

Refer to Section A.5 of this appendix for suggestions that may be useful when your file is full.

OPENLINE illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

PASTE is illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

Repeat:

The keypad editor displays this message when you press the GOLD key and then type a digit on the keyboard. Therefore, the message appears when you use the sequence to repeat a function and when you use the SPECINS function to insert a character that you cannot insert directly.

The message is primarily for information, but it is also a warning that the special effects of repeated functions and the SPECINS function are enabled.

REPLACE is illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

Search canceled

The keypad editor loads this message when you type CTRL/C or CTRL/Z to cancel the FIND function and `Model:` prompt.

Use any valid function or command.

Select range too large to CUT

The keypad editor cannot complete a CUT or an APPEND function because the paste buffer is too small for all of the material you want to store in the buffer.

Process the selection as smaller select ranges, or use auxiliary files for the same purpose as the CUT, APPEND, and PASTE functions. For example, write the selection temporarily to an auxiliary output file and close it, erase the selection, move the cursor to where you want the selection to be, and copy the material from the temporary file as auxiliary input file. You can also use any valid function or command.

TAB Indent value wrong for align

With the A function, the column number of the cursor's position is not evenly divisible by the current setting for the indentation.

Check that the cursor's position and the indent setting are both correct. To align the structured tab feature, use the A function again.

Tabs not enabled for ADJUST

You used the TABS ADJUST command before using the SET TABS command.

Enable structured tabs with the SET TABS command before using the TABS ADJUST command.

Target not found

A FIND or FINDNEXT function moved the cursor to the top or bottom of the file without finding a string that matches the model you have specified.

Use any valid function or command.

Too many arguments for command

The keypad editor did not recognize a command because you added extra characters on the command line after the last valid word or argument in the command.

Use any valid function or command.

Unable to access auxiliary device

See "?KED-F-Unable to access input device."

Unable to close auxiliary file

The keypad editor could not close the auxiliary output file. The output device may have caused the problem, and the keypad editor may have malfunctioned.

Check that the output device is on-line. Also try the suggestions that are listed for the message "?KED-F-Unable to access input device." If the message continues to appear, refer to the procedure in Section A.4 of this appendix.

Unable to delete local symbol marker
Unable to delete target
Unable to insert tab

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

Unable to open auxiliary input file

The keypad editor could not open the auxiliary input file for one of the following reasons:

- the volume does not contain the file.
- the volume has not been initialized.
- the device is not ready.

Check the file specification. If it is correct, check that the correct volume is installed and that the auxiliary input device is on-line.

Unable to open auxiliary output file

The keypad editor could not open the auxiliary output file for one of the following reasons:

- the volume has not been initialized.
- the device is not ready.
- the device is write-locked.

Check that the auxiliary output volume has been initialized. Also check that the auxiliary input device is on-line and write-enabled.

Unable to replace symbol

The keypad editor may have malfunctioned.

Refer to the procedure in Section A.4 of this appendix.

Undelete buffer full

The keypad editor could not finish processing a DELLINE or a DELWORD function for one of the following reasons:

- DELLINE would have erased more than 132 characters, the capacity of the line buffer.
- DELWORD would have erased more than 80 characters, the capacity of the word buffer.

To erase larger strings than the line and word buffers can hold, use repeated DELLINE or DELWORD functions (and ignore the error signal), or use the paste buffer or auxiliary files. You can also use any valid function or command.

Undeletes are illegal during inspect

When you are inspecting a file, the keypad editor cannot complete functions and commands that would modify the file.

Use any valid function or command.

Valid start of LSB not found

No global symbol or .ENABL LSB directive exists between the cursor's position and the preceding Formfeed character or the top of the file.

Check that the cursor is located in the first page of an LSB before you use the LOCAL command.

WORKING. . .

When a process takes longer than normal, the keypad editor flashes this message at the top of the terminal screen until the process is complete. Several interrelated factors, such as the length of the file you are editing, the devices you are using, and the functions you are using, determine when the `WORKING. . .` message appears.

The message is for information only.

WRITE finds end of file

Your last WRITE command copied the last character of your main file to the auxiliary output file.

Use another command or function.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

PDP–11
Keypad Editor
User's Guide
AD–H853A–T2

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- __ Assembly language programmer
- __ Higher-level language programmer
- __ Occasional programmer (experienced)
- __ User with little programming experience
- __ Student programmer
- __ Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

— — Do Not Tear — Fold Here and Tape — — — — — — — — — — — — — — — — — —

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG/ML PUBLICATIONS, MLO5–5/E45
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MA 01754–2571

— — Do Not Tear — Fold Here — — — — — — — — — — — — — — — — — —

Cut Along Dotted Line