IBM

Personal Computer
Professional Series

Personal Editor

[redacted]

Several people have already mentioned to me that having to remember to say "notabs" every time a file is saved is a pain. I agree (although I can't help pointing out that most of the problem would be gone if Epson had handled runs of tab characters properly). What follows is a patch that may be applied to pe.exe to permanently disable compressing runs of blanks to tabs. The syntax of the File and Save commands is not changed; but effectively "notabs" will always be enabled when writing files. The notabs option of the Edit command remains unchanged.

1. On a work disk, rename 'pe.exe' to 'xx'.
2. Enter 'debug xx'.
3. Enter 'u 70f2'. DEBUG should reply as follows:

```
   MOV W,<BP-12>,0009
   JP  70FE
   MOV W,<BP-12>,3FFF
   MOV DX,<F05C>
```

   If the output you get is different, do not continue.
   (Angle brackets have been substituted for square brackets.)
4. Enter 'e 70f5'. When DEBUG replies '09.', type 'ff '.
   Don't forget the space after ff. DEBUG will output '00.'.
   Type '3f' and enter.
5. Enter 'u 70f2'. DEBUG should reply:

```
   MOV W,<BP-12>,3FFF
   JP  70FE
   MOV W,<BP-12>,3FFF
   MOV DX,<F05C>
```

   If the output you get is different, do not continue.
6. Enter 'w'.
7. Enter 'q' to exit DEBUG.
8. Rename 'xx' to 'pe.exe'.

First Edition (November, 1982)

Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication.

International Business Machines Corporation provides this manual "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time and without notice.

Products are not stocked at the address below. Request for copies of this product and for technical information about the system should be made to your authorized IBM Personal Computer dealer.

A product comment form is provided at the back of this publication. If this form has been removed, address comment to: IBM Corp., Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

Preface

How To Use The Book

After reading this Preface, read Chapter 1 Introduction and Chapter 2 Getting Started before going on to any other section.

If you are a beginner with computer editors, read and do Lesson 1 thru Lesson 4 of Chapter 3. These lessons teach the basics of the Personal Editor and then build to a more difficult level.

If you are experienced with computer editors, go to Lessons 3 and 4. These lessons describe the Personal Editor commands and how to redefine the keyboard.

Here is how the Personal Editor book is organized:

Chapter 1 "Introduction" gives you an overview of the product with its features and requirements.

Chapter 2 "Getting Started" describes how to start the Personal Editor and what is on the Personal Editor screen.

Chapter 3 "Tutorial" consists of four lessons that teach you the fundamentals of how to use the Personal Editor.

Chapter 4 "Special Topics" describes what an internal file is, how changed or erased text can be restored, and what you should know about your computer memory limitations.

Chapter 5 "Reference" is divided into two parts. It lists alphabetically the commands and then the functions. The Reference chapter gives more details about the Personal Editor commands and functions introduced in the the Tutorial chapter.

Appendix A lists over 60 messages that the Personal Editor gives for different situations.

Appendix B lists the functions assigned to the keyboard when you first receive the Personal Editor.

Appendix C lists a suggested set of functions that you may want to assign the keyboard for word processing.

Appendix D gives four sample sheets of the 10 function keys for each set of possible function key assignments.

Appendix E describes how to make backup copies of your program and data diskette, as well as show how to put the DOS start-up procedure on the Personal Editor diskette.

Contents

Notes

Chapter 1. Introduction

Contents

Notes

Overview

The Personal Editor is a full-screen editor that lets you redefine most keys on the keyboard. This means that if you want to customize the keys to your editing style, you can.

Otherwise, the Personal Editor assigns the key definitions from a separate key definition file called PE.PRO. This is the Personal Editor's profile. You can make minor or major changes to the Personal Editor's profile, or build your own.

The Personal Editor has three internal files. One lists what the keys currently do. The second stores the most recent changes from the text so you can recover them if needed. The third gives you a directory of the files on a diskette.

The Personal Editor creates a file called the spill file that holds information on diskette that could not fit in memory. It also has an ERASE command that allows you to remove files from the diskette if the diskette is full.

The Personal Editor is quick; it scrolls pages almost instantly; it searches and changes text in large files in seconds.

Features

The Personal Editor gives you the following features:

- Full-screen editing
- Paragraph, line, and sentence markers to let you move, copy, erase, and overlay text
- Word processing features like tab and margin settings, word wrapping, and reformatting of text
- Editing of up to 20 DOS ASCII files at one time in memory
- Ability to redefine 99 keys on the keyboard
- Support for color and black and white monitors (40 or 80 column), and the IBM Monochrome Display
- ? MEMORY command that shows memory space still available anytime in the Personal Editor
- Help Menu
- Block shifting
- Diskette directory display
- Reference card and Keyboard Guide

What You Need

These are the minimum requirements of the Personal Editor:

- IBM Personal Computer
- One diskette drive (single or double-sided)
- 64KB of memory
- A monitor
- IBM Disk Operating System (DOS)
- IBM DOS formatted diskettes for storing files

The Personal Editor also supports the following:

- Two diskette drives
- IBM Monochrome Display
- Color monitor
- Television with RF modulator
- IBM Matrix Printer or any compatible printer

Notes

Chapter 2. Getting Started

Contents

Notes

Welcome to the Personal Editor

This chapter teaches you how to start the Personal Editor, how to get to the Help Menu, and how to name a file. It also introduces you to the special keys the Personal Editor uses on the IBM Personal Computer keyboard.

Starting the Personal Editor

You need your DOS diskette to start the Personal Editor. First, make a backup (see Appendix E).

The following procedure requires only one drive. You can use an IBM Monochrome Display, color, or black and white monitor (40 or 80 column).

The Personal Editor detects what type of display is connected to the IBM Personal Computer. (See the SET DISPLAY command in the command section of the Reference chapter to change from the IBM Monochrome Display to either a color or black and white monitor.)

Put the DOS diskette in drive A and start DOS by either switching on the IBM Personal Computer or holding down the Ctrl, Alt, and Del keys.

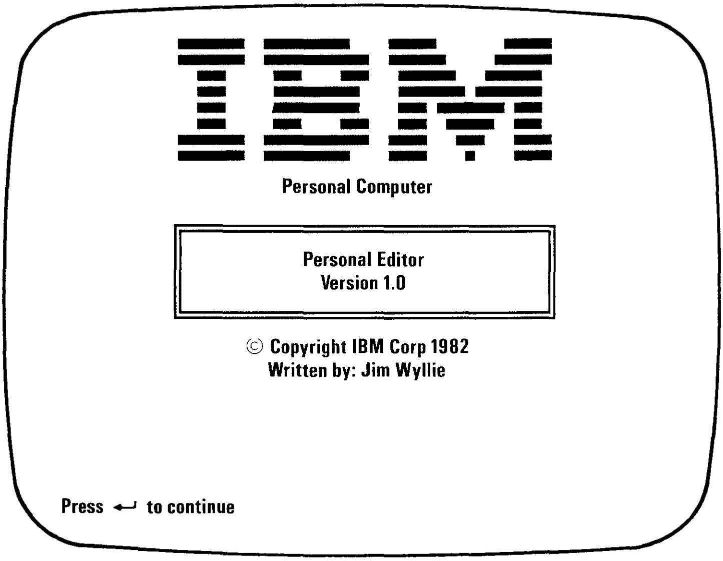
After you enter the date and time, the DOS prompt appears on the screen:

```
A>
```

Take out the DOS diskette and put the Personal Editor diskette in drive A. Type PE (the abbreviation for the Personal Editor) after the DOS prompt and press the Enter key ↵ .

```
A>PE
```

The IBM Personal Computer reads the Personal Editor program into its memory, and in a moment the Personal Editor copyright appears on the screen. Read this information.



IBM

Personal Computer

Personal Editor
Version 1.0

© Copyright IBM Corp 1982
Written by: Jim Wyllie

Press ↵ to continue

As the message says, press the Enter key ↵ to continue. The red light comes on for a brief moment and then the Personal Editor screen appears on the screen ready for you to insert text. This empty file appears each time you start the Personal Editor this way.

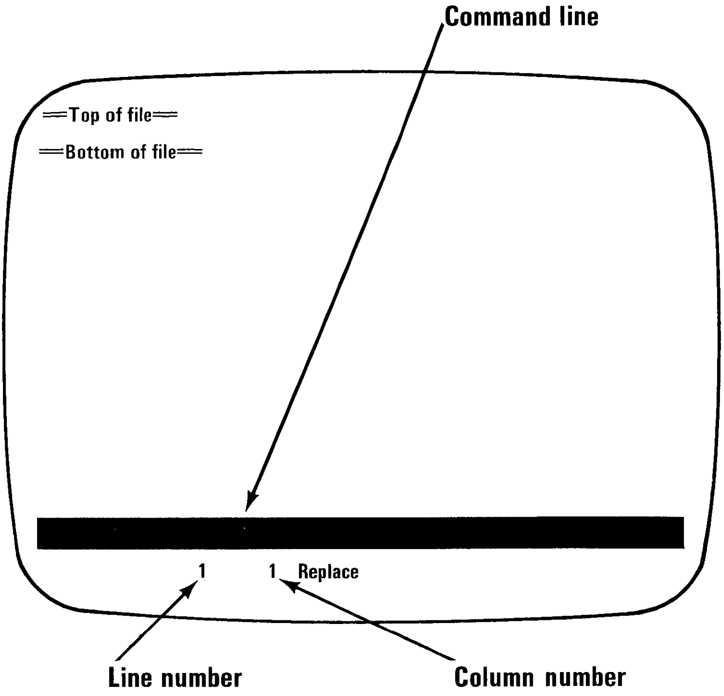
A second way is explained later.

You must always leave the Personal Editor diskette in drive A until the Personal Editor screen appears. The Personal Editor goes back and reads more information from the diskette when you press the Enter key after the copyright screen.

The Personal Editor is now ready to go.

The Personal Editor Screen

How the Personal Editor screen looks depends on the type of display device you have. The remainder of the book uses the 80 column IBM Monchrome Display for the examples, as shown here:



The Personal Editor uses 25 lines on the screen.

The Top of file and Bottom of file markers are on the first and third line each time you start a new file. You insert lines between these markers which separate them to make room for your text.

You insert all your text, whether it is program code or paragraphs of information, between these two markers.

Text Area

When the two markers are on the screen, you have a maximum of 20 lines for text. As you add lines, these markers scroll off the screen. You then have 22 lines for text. These 22 available lines are called the text area.

Command Line

The highlighted bar on line 23 is called the command line. This is where you type all the commands of the Personal Editor.

The blinking underscore (_) is called the cursor and always first appears on the command line. When the cursor is on the command line, you can type only commands.

To move the cursor between the command line and the text area, you press the Esc key. The Esc key is the third key in from the left on the top row of the keyboard.

Try it. Press the Esc key once. The cursor moves from the command line to the one blank line in the text area. You may have to look carefully because the cursor is small. (Shortly, we'll see how the cursor can become bigger and easier to see.) To get the cursor back to the command line, press the Esc key again.

Status Line

Put the cursor on the command line.

Below the command line, on line 24, is the status line. Three pieces of information can appear on the status line: the name of the current file, the line and column numbers showing the placement of the cursor, and either Replace or Insert.

Naming the First File

Let's identify our blank file "practice" using the Personal Editor NAME command so the status line can be complete. With the cursor on the command line type:

```
name practice
```

then press the Enter key ↵ .

The status line reads:

```
practice                      1      1   Replace
```

The word "name" typed on the command line is one of the seventeen Personal Editor commands.

The two numbers to the right of the filename give the line and column location of the cursor in the text area at all times. When you move the cursor around in the text, the line and column numbers change.

Try it. Move the cursor up to the text area with the Esc key. Press the Space Bar three times. The second number, the column number, changes from 1 to 4.

The words Replace and Insert indicate the two ways you can enter your text. Let's first discuss Replace since it's already on the status line.

With Replace, when you type on a line of characters that already exists, the new characters type over (or "replace") the old. When you use Replace, the cursor is a blinking underscore.

Now press the Ins key, located below the PrtSc and End keys. The cursor becomes a blinking block, which is easier to see, and the word Insert appears on the status line.

For the Personal Editor, Insert is the opposite of Replace. When you type on an existing line, the old characters move to the right to make room for the new.

Leave Insert on the status line now for the lessons in the next chapter.

Message Line

The last line is line 25. This is where you get the Personal Editor messages. All Personal Editor messages are listed in Appendix A.

To show you what a message looks like, put the cursor on the command line, type the word "message." Then press the Enter key ↵ .

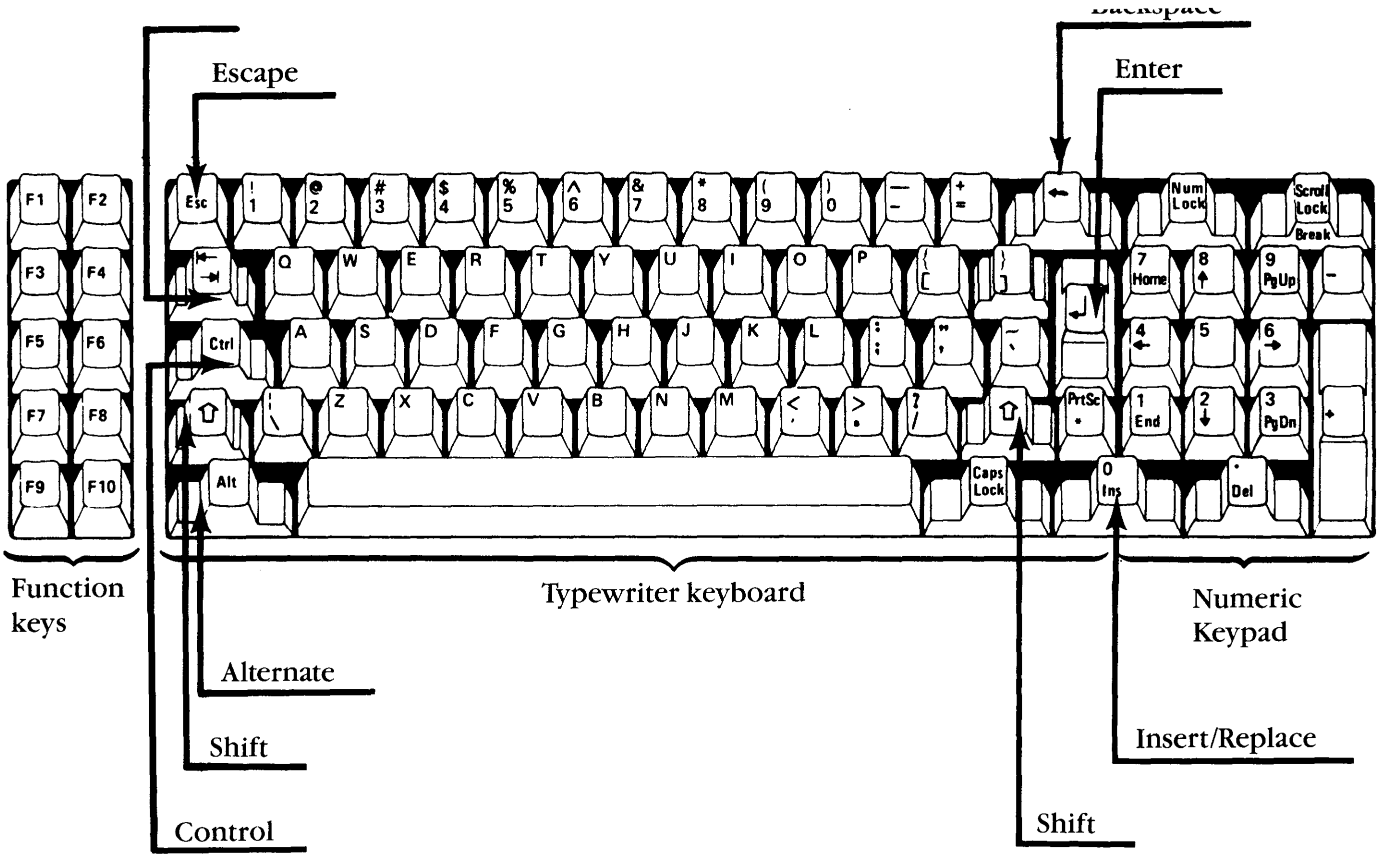
Lines 23, 24, and 25 look like this:

```
message
practice                1      4   Insert
Unknown command
```

Pressing the Enter key when the cursor is on the command line tells the Personal Editor to do the command on the command line. This happened when we named the file.

The Personal Editor expects only commands on the command line. Since the word you typed, "message," was not a command, you got a message. Press the F5 function key to remove the word "message" and the message itself.

We'll see what the ten function keys do in the Help Menu section. But first, let's look at the keyboard to see what special keys the Personal Editor uses.


Backspace
Escape
Enter
Function keys
Typewriter keyboard
Numeric Keypad
Alternate
Shift
Control
Insert/Replace
Shift

The Personal Editor makes special use of the IBM Personal computer keyboard. Let's begin by learning about the 10 function keys.

Function Keys

When you receive the Personal Editor, the 10 function keys on the left side of the keyboard are each assigned a specific function.

You can have, however, 3 more functions for each key at the same time. To do this, you would hold down one of three special keys while pressing one of the ten function keys. This would give you three more sets of function keys. Don't press any combinations yet.

Special Personal Editor Keys

These three keys are the Ctrl (Control), Shift (large up arrow), and the Alt (Alternate) keys. See the Keyboard drawing for their location.

You hold down the Ctrl, Alt, or Shift keys with one finger and press one of the 10 function keys. In the book, the combinations are depicted with dashes between the special key and another key. Ctrl-F1 is an example.

Adding the normal function key settings with the Alt, Shift, and Ctrl key combinations gives you 40 possible function keys to use.

Other Combinations

The Ctrl and Alt keys can be combined with other keys, too. The Ctrl key works with 8 cursor movement keys, plus the Enter key and the Backspace key. You can combine the Alt key with letters A to Z, and with the 10 numbers 0 to 9 on the top row of the keyboard.

The total number of keys and key combinations is 99.

Here are a few examples you'll see used in the book:

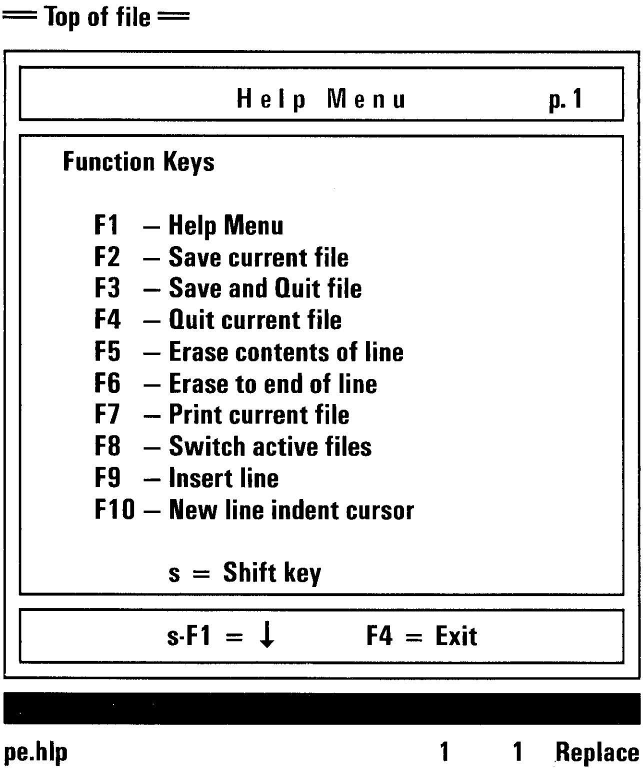
— Ctrl-Home
— Alt-Z
— Shift-F1

We'll learn how to assign functions to all 99 keys in Lesson 4 of the next chapter.

Let's use two of these key combinations in the Help Menu.

Help Menu

Press the F1 function key. The first page of the Help Menu appears:



```
== Top of file ==

              Help Menu              p. 1

  Function Keys

    F1  – Help Menu
    F2  – Save current file
    F3  – Save and Quit file
    F4  – Quit current file
    F5  – Erase contents of line
    F6  – Erase to end of line
    F7  – Print current file
    F8  – Switch active files
    F9  – Insert line
    F10 – New line indent cursor

           s = Shift key

           s-F1 = ↓        F4 = Exit

pe.hlp                             1      1   Replace
```

Scrolling the Help Menu

The Personal Editor Help Menu is 5 pages long. Page 1 gives you the list of what the 10 function keys do. Page 2 lists the steps to leave the Personal Editor. Pages 3 and 4 give the cursor movements. Page 5 lists the text markers.

Hold down the Shift key (located either above the Caps Lock key or between the Ctrl and Alt keys) and press the F1 key to see past the first page of the menu.

When you've seen page 5, hold down the Shift key and press the F2 key to move up through the menu. Try moving both up and down.

The Help Menu is a good learning tool when you are beginning to use the Personal Editor. Press the F4 function key when you are ready to leave the Help Menu.

When you press the F4 function key, the practice file returns to the screen.

You can see the Help Menu any time the Personal Editor program diskette is in the default drive.

If you only have a one-drive system and you want to see the Help Menu, you can either put the Personal Editor program diskette back into the drive, or copy the Help Menu on your data diskette.

You could use the DOS COPY command to copy the file "PE.HLP" onto your data diskettes. You would then be able to see the Help Menu with the F1 function key.

This concludes the Getting Started chapter.

If you want to put the IBM DOS start-up procedure on your Personal Editor program diskette in order to start from the Personal Editor diskette, see Appendix E.

You should, however, make a copy of your Personal Editor program diskette and use the copy instead of the original. If the copy is damaged, you have the original to make another copy. This also is described in Appendix E.

Before we begin the lessons in Chapter 3, if you have a one-drive system, remove the Personal Editor program diskette and put it in a safe place. Put the blank DOS formatted diskette in drive A and we'll use it for the next chapter.

If you have a two-drive system, leave the Personal Editor diskette in drive A and put the blank diskette in drive B and use drive B for the next chapter.

In either case, we won't be using the DOS formatted diskette until Lesson 3 of the next chapter.

Leaving the Personal Editor

If you wanted to leave the Personal Editor at this point, refer to that section in the Help Menu. For this time, use the F4 key to leave the Personal Editor and NOT the F3 key.

When you press the F4 function key, a question appears asking if you are sure you want to leave.

Typing Y removes the file from memory. Since this practice file is the only file in memory, removing it from memory returns the Personal Editor back to the DOS prompt.

Typing N clears the QUIT command from the command line so you can continue.

To return to the Personal Editor, follow the steps in the "Starting the Personal Editor" section again.

Chapter 3. Tutorial

Contents

This chapter which is broken into four easy lessons teaches you the fundamentals of the Personal Editor so you can quickly begin to create your own text.

The Personal Editor is designed to be easy to learn, yet it offers many capabilities for those who want to take advantage of them.

The Personal Editor lessons are cumulative; what you learn in one lesson carries into the next lesson.

You begin with Lesson 1 which teaches how to move the cursor. Lesson 2 shows you how to mark areas of text to move, copy, or erase. Lesson 3 explains to you how the Personal Editor commands work, and Lesson 4 discusses how to redefine the functions of the keyboard to fit your own needs.

Lesson 1. Moving the Cursor

In the Getting Started chapter, you learned about the Personal Editor screen, keyboard, and Help Menu.

In this short first lesson, you will learn how the Personal Editor uses the eight cursor movement keys pictured on the third page of the Help Menu.

Getting Started Again

If you left the Personal Editor, here is a review of how to start again:

1) At the DOS prompt, type PE and press the Enter key ↵ .

2) When the program copyright screen appears, press the Enter key ↵ again.

3) For a one-drive system, remove the Personal Editor diskette and put in the DOS formatted diskette. For a two-drive system, put the formatted diskette in drive B.

Let's begin Lesson 1.

Moving Left and Right

For the first lesson, we'll use Insert on the status line. The blinking block cursor is easier to see. Insert should already be on the status line from the last chapter. If it isn't, press the Ins key.

Look at the screen again.

Place the cursor in the single line text area by pressing the Esc key. If the blinking block cursor is not in column 1 which is the furthest left position, move it there. Do this by pressing the Cursor Left key which is located on the number 4 key of the numeric keypad. Press the Cursor Left key as many times as it takes to move the cursor to column 1.

Moving Right 40 Characters

Now hold down the Ctrl key and press the Cursor Right key (Ctrl-Right).

The cursor moves right 40 columns. The column indicator in the status line shows column 41. When you add a move of 40 to 1, you get 41.

Type a letter, "A" for example, in column 41. The letter is in column 41 and the cursor moves to column 42.

Returning to Column 1

Now move the cursor back to column 1 by pressing the Home key. The Home key returns the cursor to the first column of the line.

Moving to End of Line

Press the End key now. The cursor moves to column 43. The End key moves the cursor after the last character in a line, which for our example was column 42.

Incidentally, the blank space that you get when you press the Space Bar is considered a character. This is helpful to know when you start moving the cursor with the End key, for example, and you notice the cursor doesn't seem to go where you expect. You may have pressed the Space Bar, inserting blank characters at the end of a line of text.

Moving Left 40 Characters

Back to cursor movement. The Ctrl-Left arrow key combination moves the cursor 40 characters to the left. Go out to the last column by pressing Ctrl-Right arrow keys six times.

The cursor moves out to column 255. A Personal Editor line is 255 columns wide. Now return to column 1 by pressing the Ctrl-Left arrow keys seven times.

To demonstrate how the other cursor movement keys work, we have to insert some more blank lines with the F9 function key. Remember that the list of functions for the function keys are on the Help Menu.

With the cursor anywhere on the first line, hold down the F9 key until the Top of file and Bottom of file markers move off the screen then release the F9 key. Don't worry when the text area goes blank. Read on.

From Top to Bottom

Press the Ctrl-Home combination keys. The Top of file marker appears. Keep an eye on the line indicator on the status line.

Now press the Ctrl-End keys. The Bottom of file marker appears. The Ctrl-Home and Ctrl-End keys move the cursor to the top and bottom lines of the file.

When the cursor is on the last line of the file, the column indicator tells you how many blank lines you just inserted with the F9 function key.

Top and Bottom of Screen

Go to the top line of the file with the Ctrl-Home keys. Press the Ctrl-PgDn keys. Look where the cursor goes. The cursor moves to the bottom of the current text area.

Now press the Ctrl-PgUp keys.

Ctrl-PgUp and Ctrl-PgDn are similar to Ctrl-Home and Ctrl-End except they work within the current screen.

Ctrl-PgUp and Ctrl-PgDn move the cursor to the top and bottom lines of the current screen. The information on the screen does not change.

This ends the first lesson. Review what we've learned. Go back and practice with the blank file. The cursor can go anywhere on the screen regardless if text is on the line or not. The lines must be available, however.

Use the chart on the next page as a reference for the cursor movement keys.

The next lesson uses what you've learned here to practice marking blocks of text and doing things with the marked areas like moving and erasing them. We'll use this blank file in the next lesson.

Again, if you want to leave, press the F4 function key.

Key(s)	Moves Cursor
Cursor Up	up one space
Cursor Down	down one space
Cursor Left	left one space
Cursor Right	right one space
Ctrl-Left	40 columns left
Ctrl-Right	40 columns right
Home	to first column of line
End	after last character of line
Ctrl-Home	to top line of the file
Ctrl-End	to bottom line of the file
PgUp	none (page of text moves up)
PgDn	none (page of text moves down)
Ctrl-PgUp	to top of current screen
Ctrl-PgDn	to bottom of current screen

Notes

Lesson 2. Marking Areas of Text

Now that you know how to move the cursor around, let's learn how you can mark blocks, lines, sentences, phrases, or words of text. You move, copy, overlay, and erase marked areas.

Marking text is a basic tool of the Personal Editor. Learn this tool before you go on to the more difficult commands and functions.

To learn how to use the tool, we'll create our own sample text.

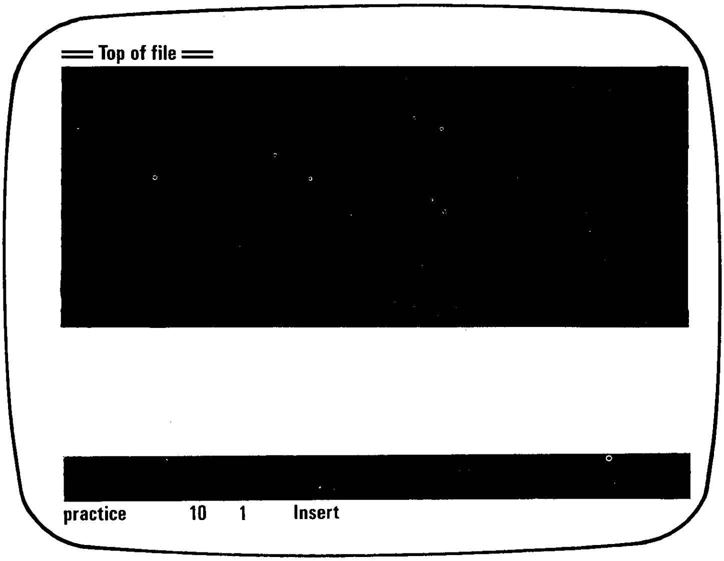
Use the screen of blank lines from the last lesson. Put the cursor in the text area with the Esc key. Press the Ctrl-Home keys to move the cursor to the top of the file, then press the Home key to go to the first column of the top line. This will be our starting point for Lesson 2.

Marking Lines

You use line marks when you want to erase, copy, or move several lines: for example, an entire paragraph or a figure.

With the cursor on the top line, hold down the Alt key and type the letter "l" (L for "line"). The first line of the text area becomes highlighted like the command line. Move the cursor down ten lines. Hold and type the Alt-l keys again.

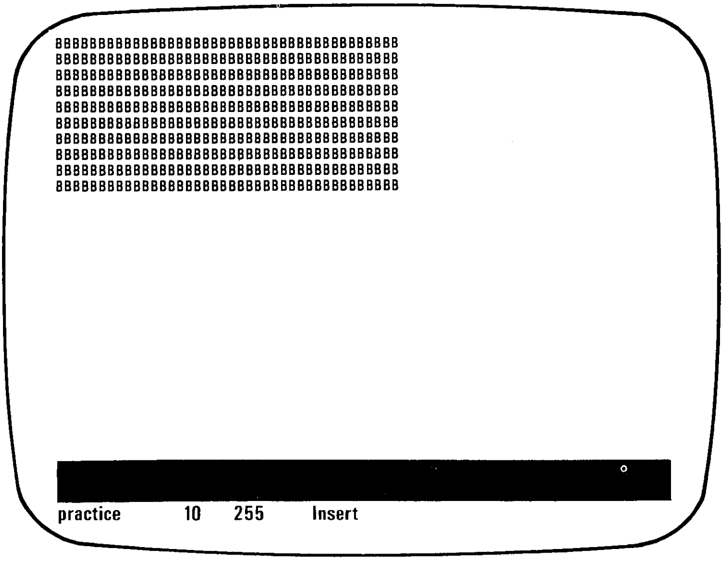
Look at the screen. Highlighting is how the Personal Editor marks all text areas. Below is the example of the line marked area.



```
== Top of file ==

practice        10    1        Insert
```

To see how text looks in the empty marked area, press the Alt-f (for "fill") keys. The Personal Editor gives the message TYPE A CHARACTER on the message line. The next letter you type will fill that marked area. Type the first letter of your name.

Now, move the cursor out to the last column of the line marked area, column 255, using the End key.



```
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

practice        10    255      Insert
```

You can see that the Personal Editor copies that one character throughout the marked area. We'll use these characters as our "text" for this lesson.

Turn off the highlighted area with the Alt-u (for "unmark") keys. The Alt-u keys unmark any type of area. The fill function turns off after the block is filled.

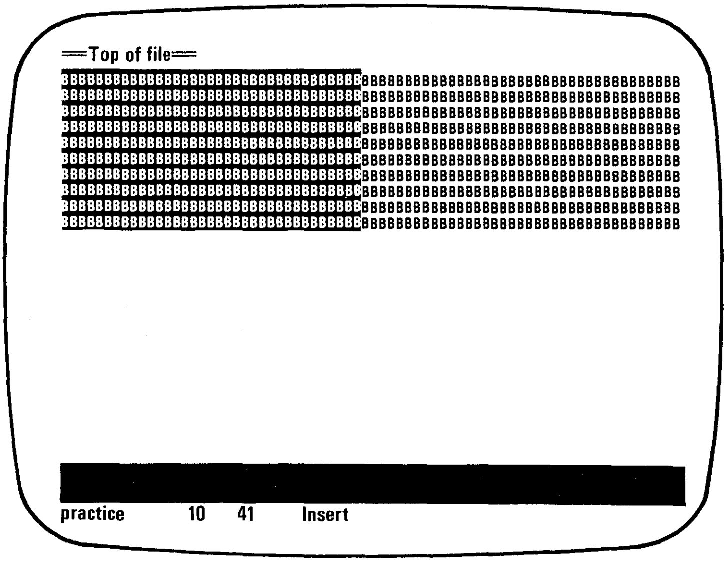
Marking Blocks

You can be more specific about what text you want to mark by using block marks.

You highlight a block by marking either the top left and the bottom right corners, or the top right and bottom left corners.

Go to the first line and column of the text area again and press the Alt-b (for "block") keys. That character becomes highlighted.

Move the cursor down to the last line of the text, then move the cursor 40 columns to the right with the Ctrl-Right arrow keys. Press the Alt-b keys again. A 41-column 10-line block forms on the text.

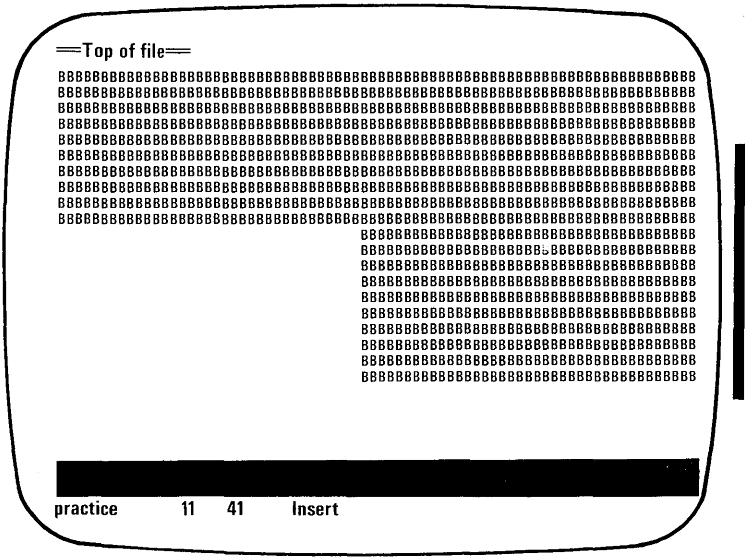


```
==Top of file==
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB



practice        10   41       Insert
```

You can also mark two points on a horizontal or vertical line with block marks. This is helpful if you want to copy a line from a chart or figure.

Marked blocks of text can be deleted, moved, copied or overlayed. Block marked areas are the only areas that can be overlayed. Let's try moving it.

Move the cursor down one line from the bottom right corner of the highlighted box. Press the Alt-m (for "move") keys. Moving is one of the four operations you can perform on marked areas.



```
==Top of file==
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB



practice        11    41      Insert
```

I hope you're not getting tired of that letter. It now covers about 75% of the screen. The Personal Editor moved the block of letters that we marked to the location below the cursor.

Without us realizing it, though, the marked area was erased. The remainder of the letters in the first ten lines moved left toward the first column, making it look like nothing happened to the marked area.

Check this by moving the cursor out to the last character of one of the original ten lines. The last character is no longer in column 255, but 215, or the original length minus the 40 moved columns.

Notice that the position of the cursor was important for moving the block. For block moves, the location of the cursor is the top left corner of the moved area.

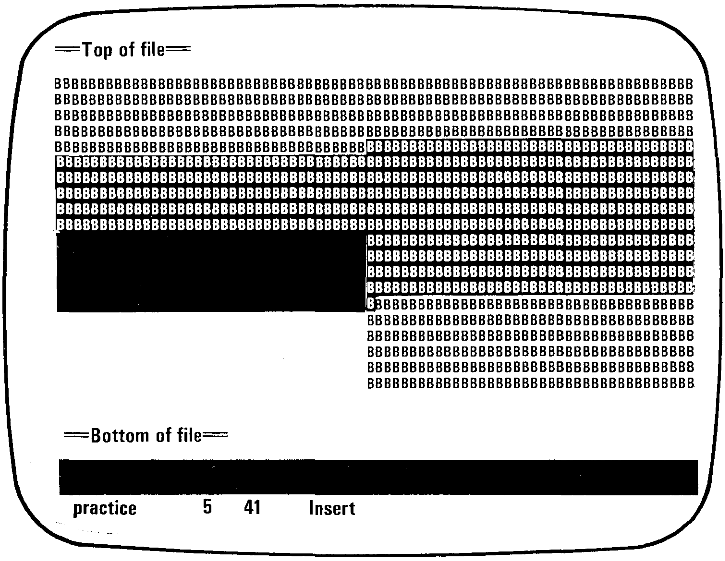
Press the Home key to get back to the first column.

Marking Sentences, Phrases, or Words

There is a third way to specify what text you want to highlight: by marking the beginning and ending of a sentence, phrase, or word.

Put the cursor in column 41 again with the Ctrl-Right keys, and move the cursor to line 5 of the original 10 lines.

Press the Alt-c (for "character") keys. Move the cursor down to the middle of the recently moved area, line 15. (We're using this area as a long sentence.) Press Alt-c again.



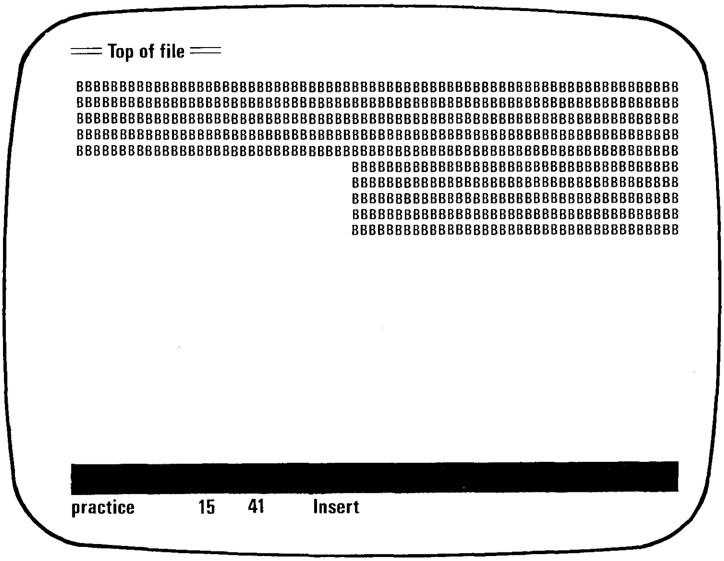
```
==Top of file==
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

==Bottom of file==

 practice        5    41       Insert
```

The highlighted area begins at the first character mark and ends at the last character mark.

Like the other two types of text markers, you can move, erase, and copy this sentence. This time, let's erase the marked text.

Press the Alt-d (for "delete") keys. Deleting marked areas of text is the second marked text operation.

The highlighting goes away, the text in the marked area is gone, and the text below the marked area moves up. The text below the deleted area moves up to fill in the gap.



```
== Top of file ==

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                                BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

practice       15     41       Insert
```

When you copy the sentence, phrase, or word to another location in the file, the Personal Editor inserts the text following the cursor position exactly as it appears in the marked area. It moves any text below the cursor down to make room.

This ends Lesson 2.

We could have used the move, copy, or erase operations with each of the three marked areas. The overlay operation works only with block marks. Overlay puts a block marked area on top of an area erasing the original text.

You can try all the combinations if you want. Practice on your own. It's a good idea to review what you have just learned.

Use the chart on page 3-18 as a quick reference of where to put the cursor to position the marked area in your text.

If you get any messages while practicing, read the explanation of the message in Appendix A. In any case, you should not have to start the lessons over. Pressing any key usually clears the message so you can continue.

The Reference chapter gives more details of how to handle text with these three text markers and the four text operations. All seven are Personal Editor functions.

After you're finished practicing, leave the practice file on the screen. We'll use it in Lesson 3 which follows.

Lesson 3 teaches you the steps to maintain your files like saving, changing, erasing, and renaming them.

Again, if you wanted to leave the Personal Editor, you would press the F3 function key followed by the Enter key ↵ . The FILE function would save any work you've done and remove the work from memory. It also would quit the Personal Editor because the PRACTICE file is the only file in memory.

To start again, type PE, a space, the drive letter, a colon, followed immediately by the filename PRACTICE.

After you press the Enter key the second time for the copyright screen, the Personal Editor searches for the PRACTICE file and brings it to the screen.

Where To Position Cursor In Your Text
For Three Operations*

Type of Operation	Type of Mark		
	Line Alt-L	Block Alt-B	Char Alt-C
Copy Alt-Z	Line above	At line	At line
Delete Alt-D	Any position	Any position	Any position
Overlay Alt-O	—	Top left corner	—
Move Alt-M	Line above	Top left corner	Top left corner

*Alt-u unmarks all marked areas.

Lesson 3. Learning About Commands

Welcome to those who are just starting.

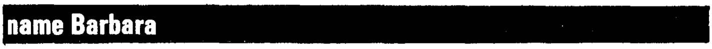
We've covered how to move the cursor. We saw what text looks like in the text area and how to move that text around in the file. With this lesson, we'll learn how to maintain a file with Personal Editor commands.

Starting with the practice file from the first two lessons, we'll rename the file, and save it on the diskette.

First, move the cursor to the command line by pressing the Esc key.

Naming A File

Type the NAME command on the command line again, followed by your first name. If you're using two drives, type "name b:*yourname*" (eight letters maximum).



```
name Barbara
```

Press the Enter key ↵ .

Your name now appears on the status line. All filenames appear in lowercase letters.

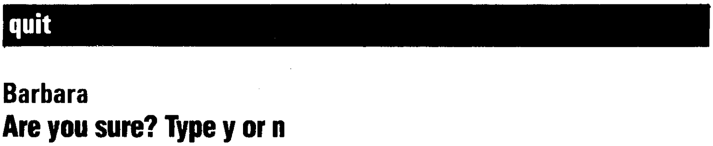
We have used this command once before in Chapter 2 Getting Started.

When you type any command on the command line, the word does not have to begin in the first column. You can type a command anywhere on the command line. Make sure, however, the command and what goes with the command are the only things on the command line.

Quitting A File

Let's try quitting our practice file. Quitting a file means that you erase the file from the IBM Personal Computer's memory. The first step is to press the F4 key.

The word "quit" appears on the command line and a message appears on the message line:



```
quit

Barbara
Are you sure? Type y or n
```

The Personal Editor asks you if you are sure before it removes the file from memory. It only asks this if you've made changes to the file since the last time you saved it. Because you have not yet saved this file, the Personal Editor considers it a changed file.

If you type the letter N for NO, the QUIT command leaves the command line, the message leaves the status line, and the cursor remains on the command line. Do it. Type the letter N.

If you had typed the letter Y for YES, you would have removed all the text in the practice file as well as the practice file itself. Also, when you quit from the last Personal Editor file, you return to DOS.

Saving A File

Let's save the file instead with the SAVE command. Press the F2 function key.

The word "save" appears on the command line. If it didn't, press the F5 function key to erase the line and press the F2 key again. Press the Enter key ↵ when the word "save" appears on the command line.

When the red light of the drive comes on, your file is being saved on the diskette under your name.

The Personal Editor uses the filename and device information from the status line when it saves your file.

The Personal Editor "executed" the SAVE command; it did the command on the command line when you pressed the Enter key.

The cursor does not have to be on the command line to execute a command. If the cursor is in the text and you want the Personal Editor to do the command on the command line, press the Ctrl-Enter keys. These keys make the Personal Editor do what's on the command line no matter where the cursor is.

It's a good idea to save your files periodically. If a power failure occurs, the text you typed after your last save is lost. Saving a file should become more of an updating procedure.

Because you have not made any changes to the file since you saved it a few seconds ago, if you were now to use the QUIT command, your file would be removed and you would return to DOS. Don't use the QUIT command now.

File Command

If you wanted to save a file and then remove it from memory in one step, you would use the FILE command to save yourself time and keystrokes.

If you would have pressed the F3 function key, the word "file" would have appeared on the command line. Pressing the Enter key would have caused the Personal Editor to save and quit the file.

You would not get a message asking you if you're sure you wanted to remove the file because you would not have made changes since the SAVE command.

Think of the FILE command as filing a piece of information away for later use.

Editing A File

Now it's time to create two more files. On the command line, type "edit program.bas". For two drives, type "edit b:program.bas".

```
edit program.bas
```

Press the Enter key ↵ .

The Personal Editor looks on your diskette for the file. Since the file is not on the diskette, the Personal Editor creates a new file with that name. This is how you create all your new files.

Let's create another file. Don't worry about the first file. You can create up to 20 new files on the Personal Editor each time you start just by typing the EDIT command and a filename.

The Personal Editor can hold old and new files in memory to a combined total of 20.

On the command line, type "edit text.doc". For two drives, type "edit b:text.doc".

```
edit text.doc
```

```
program.bas                 1    1           Insert
```

Press the Enter key ↵ .

The abbreviation ".bas" is a file extension that stands for the programming language, BASIC. The extension ".doc" stands for documentation.

The Personal Editor does the same procedure with this file as with the first file, PROGRAM.BAS.

Now you have three active files. Active means that the file is in memory. It also means that you can look at it with the touch of a button. Try it. Press the F8 function key. This function key switches from one active file to another on the screen.

Our first practice file with your name on it appears. Press the F8 key again. The PROGRAM.BAS file appears. Press the F8 key a third time and the TEXT.DOC file appears.

Since the last two files are empty, only the name changed. There is no text to display.

Switching files with the F8 function key is like looking at photographs in your hand. You slide the top picture to the bottom of the stack. Eventually, if you look through all the pictures, the first one returns to the top.

With the Personal Editor, if you had five files in memory and pressed the F8 key five times, the first file would return.

This is what happened when you pressed the F8 key the third time. The PROGRAM.BAS file returned to the screen.

Before we go any further, let's remove the practice file with our name on it from memory. Press the F8 key again and the practice file appears. With the practice file on the screen, press the F4 function key. The file is removed. Remember, the file is already saved on the diskette.

To learn the next commands, we need text in the two new files again.

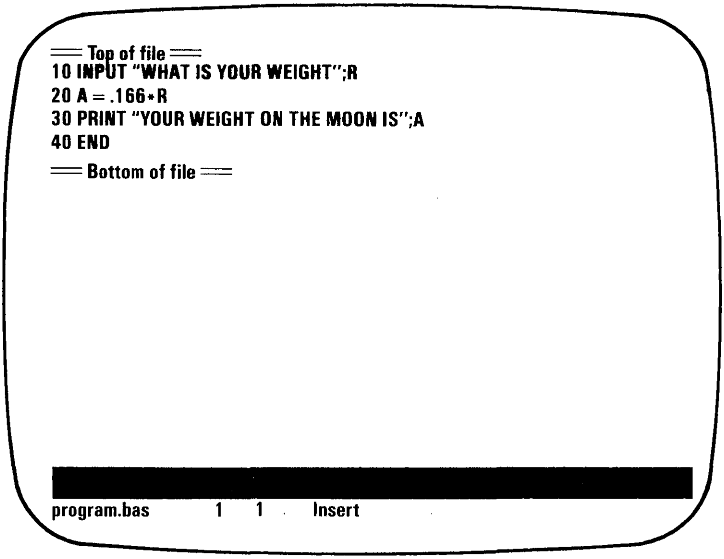
This time we'll type a simple BASIC program in the PROGRAM.BAS file and a paragraph in the text file. (You don't need to know how the program works.)

The two files show the variation of Personal Editor files you can create.

Switch back to the PROGRAM.BAS file with the F8 key, move the cursor to the text with the Esc key, and type the program shown below. Press the F9 function key at the end of each line to take you to the beginning of the next line.

```
10 INPUT "WHAT IS YOUR WEIGHT";R
20 A = .166*R
30 PRINT "YOUR WEIGHT ON THE MOON IS";A
40 END
```

Your screen should look like this:



```
══ Top of file ══
10 INPUT "WHAT IS YOUR WEIGHT";R
20 A = .166*R
30 PRINT "YOUR WEIGHT ON THE MOON IS";A
40 END
══ Bottom of file ══





program.bas        1    1     Insert
```

Switch back to the TEXT.DOC file with the F8 key.

Type the following sentence on the first line of the text area (do not press the F9 function key like you did for the program):

> The BASIC program in the PROGRAM.BAS file asks you how much you weigh then calculates your weight on the moon.

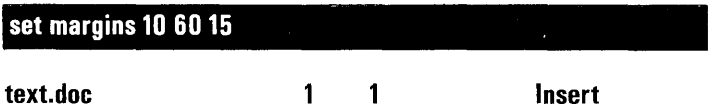
If you typed the sentence exactly as it is on one line, the cursor should be at column 111. The Personal Editor adjusts the screen to show you the rest of the sentence once you pass the right edge of the screen.

Changing Margins

The program in the PROGRAM.BAS file could have been written all on one line and it still would have calculated your weight. Breaking it up as we did makes it easier to read.

Documentation, on the other hand, requires a maximum of 80 columns for an 8½ by 11 page. The Personal Editor allows you to set the left and right margins to make your documentation fit on a page.

For example, type the following on the command line:



```
set margins 10 60 15

text.doc                  1    1            Insert
```

Press the Enter key ↵ .

Nothing seems to happen except that the command is gone.

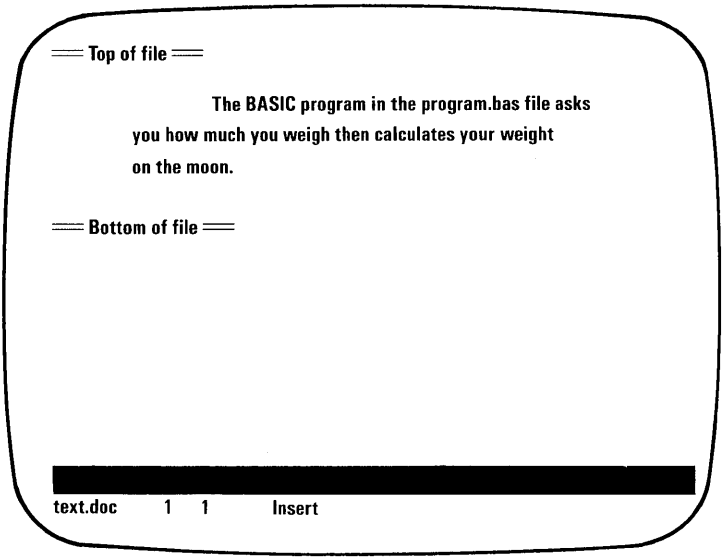
Whenever you start the Personal Editor, the margins are 1, 254 and 1. What you have just done is change the left margin of 1 and right margin of 254 to a left margin of 10 and a right margin of 60. The third number is for a paragraph identation of 15.

To get the one line of text to fit within the new boundaries, move the cursor to the one line of text and press the Alt-p keys (for "paragraph reflow").

The paragraph reformatting is very quick.

Press the Home key with the cursor in the text area to see the reformatted paragraph.

The screen should look like this:



```
== Top of file ==
                    The BASIC program in the program.bas file asks
          you how much you weigh then calculates your weight
          on the moon.

== Bottom of file ==





text.doc      1   1        Insert
```

The Personal Editor remembers what margins you set for each file. If you go to the PROGRAM.BAS file and begin typing without pressing the F9 function key, the screen will adjust to show the characters typed past the right margin. The margins for the program file are 1, 254, and 1.

In the TEXT.DOC file, if you move the cursor to column 10 and begin typing on a new line, the words will all stay within the margin settings of 10 and 60.

Anytime you want to find out what margins you have set for a file, use the QUESTION MARK command and the word MARGINS on the command line. Then press the Enter key. Try it now.

This is handy when you can't recall the margin for an active file. Press the Enter key to remove the information.

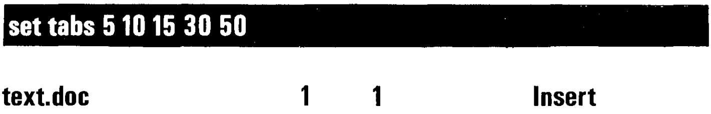
Changing Tabs

There are two more SET commands with the Personal Editor: SET TABS and the SET DISPLAY.

The Personal Editor allows you to set tabs like on a typewriter. You can set a maximum of 20 tabs between 1 and 255.

The format for the SET TABS command is similar to the SET MARGINS command except that you type "tabs" instead of "margins".

Again for this example, type the following on the command line:



```
set tabs 5 10 15 30 50

text.doc                    1     1              Insert
```

Press the Enter key ↵ .

With the cursor in column 1 of the text area, press the Tab key once. The cursor goes to the fifth column. Press it four more times and the cursor goes to column 10, then to 15, 30 and 50.

When you start the Personal Editor, the tab settings are in columns 9, 17, 25, and so on, moving 8 columns to the right each time you press the Tab key.

Like the margin settings which apply to the file in which they are set, any time you change the tab settings, these new tab settings apply to the new file and any file edited after that.

As with margins, you can see what tabs you've set for a current file. Type ? TABS on the command line and press the Enter key ↵ .

Changing Display Devices

The third SET command is the SET DISPLAY command.

The Personal Editor allows you to edit files on the IBM Monochrome Display, a color, or black and white monitor.

If the monitors are properly connected, you can use the SET DISPLAY command to change from one monitor to another. If they are not connected properly, the Personal Editor gives you a message.

If you have another monitor connected to the IBM Personal Computer, read the description of the SET DISPLAY command in the Reference chapter.

Locating Text

There are three more commands I want to cover in this lesson with you. The first two are related.

Suppose you have a long program or document, and you want to see a particular section of that file quickly. You can either use the PgDn or PgUp keys, or you can use the LOCATE command.

Let's use the LOCATE command to find a word in the PROGRAM.BAS file. Switch to that file with the F8 key. Put the cursor in the first column of the first line.

Go to the command line with the Esc key. Type the following:

```
locate/40

program.bas                    1      1            Insert
```

The "/" character is on the question mark key.

Press the Enter key ↵ .

For programmers, this is a way to find a line number or variables in a program. The Personal Editor finds the line and brings it to the bottom line of the current screen.

You can search for words too. If a word appears more than once in a file, you can quickly and easily search for each occurrence.

After you type the command, slash, and the word you want to find, press the Enter key ↵ . The command, slash, and the word remain on the command line. The cursor, however, goes to the first occurrence of the word in the text.

To find the next occurrence of the word in the file, you can return the cursor to the command line with the Esc key and then press the Enter key ↵ . A quicker way is to press the Ctrl-Enter keys. Remember, pressing these two keys does whatever command is on the command line regardless of the cursor position.

You also can search backwards. You would type a second slash after the word followed by the minus sign (dash).

For example, "locate/dog/-".

After pressing the Enter key, the search would begin to the left of the cursor and continue to the top of the file.

From now on we'll use the term "string" instead of "word". Not all characters that you may want to look for are words. You can look for numbers (as before), or special characters like parentheses or exclamation points. These are called character strings.

Changing Text

The command related to LOCATE is the CHANGE command. It finds strings like the LOCATE command, but it also replaces the found string with a new one.

Switch to the TEXT.DOC file. Put the cursor on the command line and type the following:

```
change/the/THE/
```

text.doc

Don't press the Enter key yet.

What this example of the CHANGE command says is change the first occurrence of the word "the" to uppercase "THE".

Let's try it. Move the cursor to the first column of the first line with Ctrl-Home and Home, then press the Ctrl-Enter keys.

The cursor moves over to the first occurrence of the word "the". The message CONFIRM CHANGE appears on the message line. The Personal Editor waits for you to tell it to replace the string it found with the new string. You do this by pressing the Shift-F5 keys.

The Personal Editor exchanges the strings and then moves the cursor to the last character of the new string. The command and the two strings remain on the command line.

If you had typed any other character and then pressed the Shift-F5 keys, you would have gotten a message saying that no change was going to be made to that string.

This time we purposely won't change the next occurrence of the string. Press the Ctrl-Enter keys. The cursor moves to the second occurrence of the word and gives the CONFIRM CHANGE message again.

Press the Esc key to get the cursor back to the command line. Now press the Shift-F5 keys. You get the message NO CHANGE PENDING. Now press the Enter key ↵ . The cursor goes to the third occurrence of the word. Press the Shift-F5 keys. The strings exchange.

You can change or not change a string throughout the file as long as the string is in the file. Otherwise, you would get a message saying that the string was not found.

Press the Esc key to get back to the command line. Press the F5 function key to clear the command line. Leave the cursor on the command line.

There is a way to change all occurrences of an old string to a new string from the cursor position to the bottom of the file. You type an asterisk after the slash.

If you include the minus sign (dash) with the asterisk, the changes occur from the cursor position to the top of the file. (The minus sign always goes before the asterisk if you combine them.)

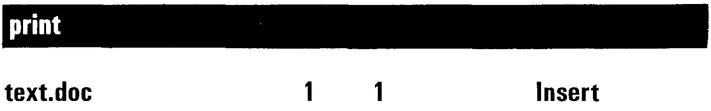
If you use only the minus sign as you did with the LOCATE command, the Personal Editor would search for the first occurrence of the string from the cursor position to the top of the file.

Printing Files

The last function we'll cover in this lesson is the PRINT command. Let's print out the two files that we've been using.

First, the IBM Matrix Printer (or any other compatible printer) must be properly connected, have paper in it, and the printer must be ready to print. Check your printer before you continue reading.

To use the PRINT command, press the F7 function key. The following appears on the command line:



```
print

text.doc                  1    1              Insert
```

Again, the Personal Editor command requires that you press the Enter key to do the command. This two-step process allows you to stop for a moment to see if you want that command.

If your printer is ready, press the Enter key ↵ .

The file is printed without the Top of file and Bottom of file indicators, and without the information on the status line.

When the first file is printed, switch files with the F8 key and press the F7 key to print the PROGRAM.BAS file. Printing the file does not change it; printing does not save or remove it from memory either.

This ends the third lesson.

Let's end the lesson by saving and removing one of the two active files from memory with the FILE command. We'll use the PROGRAM.BAS file that's on the screen.

Press the F3 function key to get the FILE command, then press the Enter key ↵ .

In a moment, the first file is saved on the DOS formatted blank diskette and then removed from the computer's memory. The second file appears. We'll use this in our final lesson.

Remember, you can use the F3, F2, and F4 function keys to file, save, and quit a file anytime the cursor is either in the text area or on the command line.

Also remember that for more details on the editing features in this and any other lesson, read the Reference chapter.

The next lesson discusses two commands we haven't covered yet. These two commands let you customize your own Personal Editor to make it do what you think an editor should do.

Lesson 4. How to Design Your Own Editor

Up to now, we have told you what each of the keys do for the Personal Editor. When you're finished reading this lesson, you will be able to decide, based on your own style, what keys do what functions for you.

The two remaining commands allow you to customize your Personal Editor.

Define Command

The DEFINE command is the most powerful command of the Personal Editor. It allows you to change the definition of a key. Definition refers to what the key does when you press it.

For example, when you press the Enter key ↵ , the cursor moves to the beginning of the next line if it's available.

Let's change the definition. Type the following on the command line exactly as it appears:

```
define enter = [insert line]
```

Make sure you include the square brackets.

Press the Enter key ↵ . The command line clears.

What you have just done is redefine the Enter key to insert a new blank line below the current line moving the cursor to the new line. This is similar to what happens when you press the Return key on a typewriter.

Try the new definition of the Enter key anywhere in the TEXT.DOC file from the last lesson. Watch the Bottom of file marker move down the page like it did in the last lesson with the F9 function key.

This is just one example.

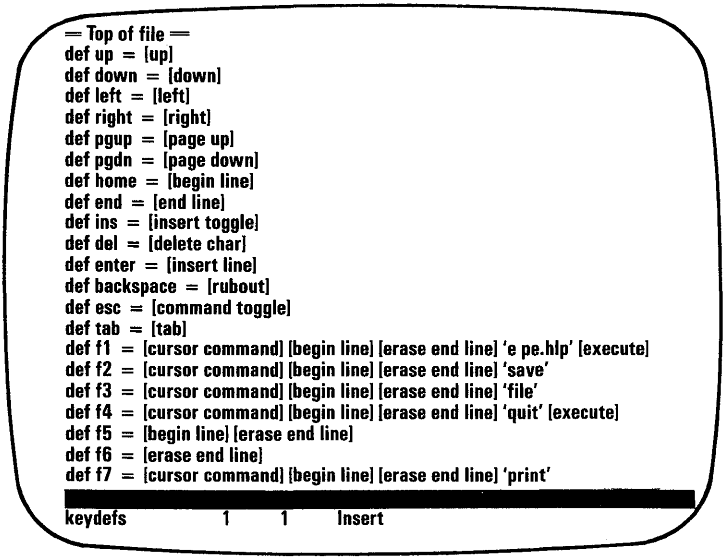
A benefit of being able to redefine keys like this is so you can reduce some repetitive editing tasks to a few keystrokes. You do this by assigning a series of Personal Editor functions and commands to one key or a combination of keys.

For example, when you typed Alt-p in Lesson 3 to reformat the paragraph within the new margins, you used eleven different Personal Editor functions at one time. You only pressed two keys.

To see the list of functions for this and other keys, type the following on the command line (including the period):

```
edit .keydefs
```

You see the following screen:



```
= Top of file =
def up = [up]
def down = [down]
def left = [left]
def right = [right]
def pgup = [page up]
def pgdn = [page down]
def home = [begin line]
def end = [end line]
def ins = [insert toggle]
def del = [delete char]
def enter = [insert line]
def backspace = [rubout]
def esc = [command toggle]
def tab = [tab]
def f1 = [cursor command] [begin line] [erase end line] 'e pe.hlp' [execute]
def f2 = [cursor command] [begin line] [erase end line] 'save'
def f3 = [cursor command] [begin line] [erase end line] 'file'
def f4 = [cursor command] [begin line] [erase end line] 'quit' [execute]
def f5 = [begin line] [erase end line]
def f6 = [erase end line]
def f7 = [cursor command] [begin line] [erase end line] 'print'

keydefs           1     1     Insert
```

The word ".keydefs" stands for KEY DEFINITIONS. It is one of the three internal files of the Personal Editor. That is why the red light on the drive did not come on. Internal files are explained in the next chapter.

The .KEYDEFS file gives you the current key definitions for the available keys.

Temporary Changes

Look at the .KEYDEFS file. It shows the change that we made to the Enter key. Let's change the definition of the Enter key back to the way it was.

On the command line, type:

```
define enter = [begin line] [down] [indent]

.keydefs                    1    1            Replace
```

Press the Enter key ↵ . (Remember, pressing the Enter key on the command line does the command.)

Nothing seems to happen. To see the change to the original definition of the Enter key, edit the .KEYDEFS file again.

In a moment the definition of the Enter key changes on the screen. The Personal Editor gives a new snapshot of the current settings.

Using the DEFINE command on the command line is how you temporarily change key definitions. The .KEYDEFS file is where you find the temporary changes. How you make permanent changes is discussed later in this section.

One reason for making temporary changes is to see how a function or a series of functions work before you decide to make a permanent change. The definition only remains permanent until you change it again.

Use the Enter key in the text again. It should work as it first did; that is, move the cursor to the next available line. If there is a blank line below the cursor, when the Enter key is pressed twice, the cursor indents to the paragraph margin setting.

Combining Functions

Back to the combined functions of the Alt-p keys. Press the PgDn key three times to see the functions assigned to the Alt-p keys.

The Personal Editor uses lowercase "a" for the Alt key, "c" for the Ctrl key, and "s" for the Shift key. You cannot use uppercase characters when you define a key or type a function.

With the cursor on the command line, move the cursor up to the definition of the "a-p" keys with the Cursor Up key. The cursor leaves the command line and moves up the page. Press the End key to see the last functions assigned to that combination of keys.

Press the Home key to get back.

Each one of the bracketed words or groups of words is a function. Combining functions is like gathering people with different talents to do one job more efficiently.

Let's discuss a simpler list of combined functions assigned to a key. Use the LOCATE command from the last lesson to find the F1 key. On the command line, type "/def f1   /-" and press the Enter key ↵ .

If you didn't put a space after F1, the cursor moved up to the F10 key. You must remember to be specific when you type strings. If the cursor is at F10, press the Ctrl-Enter keys to move the cursor to F1.

The F1 key is assigned to get you to the Help Menu.

Briefly, this is what happens when you press the F1 function key: the cursor moves to the beginning of the command line, erases whatever is on the command line, puts the command string on the command line, and does the string.

The command string consists of the abbreviated form of the EDIT command and the name of the Help Menu file.

Pressing the F1 key represents typing the EDIT command and Help Menu filename on the command line followed by the Enter key.

If you haven't noticed yet, you can change the Help Menu. You can move the cursor around inside the file and change any or all of the information. This allows you to design your own Help Menu if you decide to change the definitions of keys.

You may not want to change anything also. It's up to you.

Macro Command

The MACRO command takes the DEFINE command one step further. The MACRO command executes a list of DEFINE commands. The list must appear in a separate file called a command file.

The DEFINE command is not the only command that can appear in a command file. You can use all the commands in a command file. Here is an example of what commands can be in a command file:

```
edit program.bas
print
quit
edit text.doc
print
quit
```

This short command file with its six commands brings the PROGRAM.BAS file into memory, prints it, and removes it from memory. It does the same thing with the TEXT.DOC file.

Here is how you create a command file from these six commands:

1) use the EDIT command to create a new file,

2) type the six commands in the text area of the new file (one command per line),

3) and save it.

Let's try it.

First, create a new file. Type the EDIT command followed by a name you'll remember. For example,

```
edit print2
```

Don't forget the "b:" if you have a two-drive system.

Move the cursor to the text area and type the six lines the way they appear above.

Press the F2 key to get the SAVE command, then press the Enter key ↵ to save the command file.

Now, to use the command file, type the MACRO command on the command line followed by the command filename.

On the command line type:

```
macro print2
```

Don't press the Enter key yet.

You could also have assigned the MACRO command and the command filename to a key enclosing them in quotes.

For example:

```
define a-5 = 'macro print2' [execute]
```

Do it this way the next time you create a command file.

Check the printer to see if it's ready. If so, press the Enter key

The MACRO command makes the Personal Editor go into the command file and do the commands one at a time in the order they are typed.

The two files print out one after the other.

Whenever you want to print these two files again, type the MACRO command and the command filename on the command line again.

If you assign the MACRO command and the filename to two keys, you reduced 53 keystrokes to 2.

Also, with the command file, you don't have to wait for the first file to print before you tell it to print the second file.

The Personal Editor not only helps reduce typing of repetitive tasks but saves time as well.

Saving New Key Definitions

Now you know how the DEFINE and MACRO commands work.

Next we'll learn three ways to save your changes of the key definitions.

We won't be typing anything for the remainder of the tutorial, so just read along.

To begin, when we assigned the Insert Line function to the Enter key above, the change was temporary, not because we changed it back, but primarily because we didn't save the change.

The purpose of saving new definitions is so you do not have to type them again the next time you start the Personal Editor. All temporary changes are erased when you turn off the Personal Editor.

Two ways to make save changes involve the .KEYDEFS file that we were introduced to above.

You came close to saving the new key definition of Enter when you changed its definition above. You typed the DEFINE command, the key, and the function on the command line. Then you used the EDIT command to see the changed .KEYDEFS file.

You needed two more steps to complete the change.

After you used the EDIT command to see the new temporary key definition, you could have used the NAME command to rename the .KEYDEFS file, then saved the file under the new name with the SAVE command.

With these two additional steps, the new filename would have appeared on the status line, and a copy of the changed file would have been made on the diskette.

Since the .KEYDEFS file is an internal file, you cannot permanently change it. That is why you would have had to rename the file before you saved a copy of the changes.

If you had tried to save the .KEYDEFS file without renaming it, you would have gotten a messsage saying you can't save an internal file.

If you used the EDIT command to get the .KEYDEFS file again, the two files would look alike.

What you would have done was create a new command file that could be used anytime you are in the Personal Editor. Typing the MACRO command, the filename and pressing the Enter key would assign the Enter key the Insert function.

The MACRO command would assign the new definition to the Enter key until you turn off the Personal Editor, or until you use another command file that changes the key definition again.

The second way to save a definition change is similar to the first way. Instead of typing the DEFINE command, the key, and the function on the command line, you could move the cursor into the file and change the functions in the file itself.

You would then rename the file and save it. When you do it this way, however, you do not get the temporary changes that you can check to see if they work before you save them.

PE.PRO

The third way to save a changed key definition is to change the PE.PRO file.

In Chapter Two, when you pressed the Enter key after the Personal Editor copyright screen appeared, the Personal Editor read a command file called PE.PRO.

This command file assigned the initial key definitions to the keyboard. See Appendix B for this PE.PRO file.

If you use the EDIT command to see the .KEYDEFS file just after you start the Personal Editor, there would be no difference between the .KEYDEFS file and the PE.PRO file.

Since the .KEYDEFS file displays the current key definitions, the initial PE.PRO key definitions would be the most current.

To make definition changes in the PE.PRO file, you must move the cursor into the text area. You can erase and add functions and strings throughout the PE.PRO file.

Once you have made all your changes to the PE.PRO file, save it under the same name.

Do not change the filename, PE.PRO. The Personal Editor looks for a file called PE.PRO when it starts, and if it cannot find it, it gives you only a minimum set of key definitions with which to operate.

After you save the changes, type "macro pe.pro" on the command line and press the Enter key. The Personal Editor redefines the keys to the new key definitions.

Appendix C is an example of a command file designed for word processing. You could type the word processing changes into the .KEYDEFS file, rename the file to something that reminds you it's a word processing profile, then save it. Then when you wanted a word processing profile you would type the MACRO command and the filename on the command line.

If you wanted the original profile of the Personal Editor, which is more for programming, you would then type the MACRO command and PE.PRO.

You can make as many command files as you have different purposes for them.

You can design one to help you type a structured programming language, or type a memo. You can design one to keep facts and figures, or to write letters to Mom.

Before you attempt to construct these command files, however, try to understand why the key definitions in the PE.PRO file and the word processing command file are organized the way they are.

Become familiar with how individual functions work and how they work together. You then can combine them to do the "job" you want.

This concludes Lesson 4 and the tutorial.

You can do a lot with the Personal Editor. I wrote this book using the Personal Editor using the command file in Appendix C.

I recommend you read the Reference section before you change the key definitions. Learn what tools you have before you begin to use them.

The next chapter discusses special Personal Editor topics like how to recover erased or changed text, and what to do when you run out of memory.

Notes

Chapter 4. Special Topics

Contents

Notes

Internal Files

The Personal Editor has three internal files, the .KEYDEFS file, the .UNNAMED file, and the .DIR file. Internal means that the file is generated inside the Personal Editor program and does not occupy space on the diskette. Internal files provide information you can use when editing.

The .KEYDEFS file holds the current key definitions. (See Lesson 4 for details.) The .UNNAMED file holds the last five changes you made to active files. (See the next section of this chapter for details.) The .DIR file gives the name, byte size, date and time of DOS files on one diskette. (See the DIR command in the Reference chapter.)

You view the .KEYDEFS file and the .UNNAMED file with the EDIT command. You view the .DIR file by typing DIR on the command line with no period.

Here is a list of characteristics for internal files:

- All three files, once brought to the screen, become active files.
- Active internal files use a portion of available memory.
- You can line, block, and character mark any information from an internal file to another file and save that information on the other file.
- You cannot use the SAVE or FILE command on internal files unless you supply a valid filename with the command.
- You must use the QUIT command to remove the internal files from memory.

Restoring Erased or Changed Text

There are two ways to put back erased or changed text into a file.

One way to recover a lost line of text is with the [undo] function. This function is initially assigned to the Shift-F4 keys when you start the Personal Editor.

The [undo] function restores a line to the way it appeared before the cursor was moved to it. This includes if the line was blank before you began to type on it.

The [undo] function restores characters that were erased or changed by:

- the [delete char] function (Del key),
- typing characters in Insert or Replace style,
- [erase end line] function (F5 and F6 function keys),
- [erase begin line] function.

The [undo] function will not restore lines that were

- deleted with the [delete line] function (Ctrl-Backspace keys), or
- words that were erased by a series of functions assigned to a key.

Once the cursor leaves the line that you've changed, you cannot use the [undo] function to recover the original line. The cursor must remain on the changed line to recover the text.

The second way to recover the text is using the .UNNAMED file. It allows you to recover text once the cursor leaves the current line.

The .UNNAMED file is a Personal Editor internal file that holds the last five changes you made to active files.

If you change a line in one active file, then make single changes in four other active files, the .UNNAMED file shows all five changes. If you make one more change, the first of the five changes is removed from the .UNNAMED file.

The five changes are separated by blank lines.

If many lines are block shifted, you may get a memory full condition because of the space required in the .UNNAMED file to hold the change.

The following is a list of the type of changes that appear in the .UNNAMED file:

- paragraphs before you used the paragraph [reflow] function
- truncated lines that went past the right column limit of 255
- a one space block shift (each block shift is a single change to the .UNNAMED file)
- changed sentences up until the line had to split to fit into the margin settings
- deleted lines

- same changes or deletions covered in the above mentioned [undo] function

- all global and individual changes with the CHANGE command

- deletions with the [rubout] function (Backspace key) and the [delete char] function (Del key).

Storage Considerations

The Personal Editor operations make use of available memory. You must consider your IBM Personal Computer memory size when you use the Personal Editor.

Minimum Memory Size

If your IBM Personal Computer has the minimum Personal Editor memory requirement of 64KB and has two disk drives, loading the Personal Editor program and the default PE.PRO file into memory leaves you approximately 8KB of memory to use.

Exactly how many files you can edit with this amount of memory depends on the size and type of operations you do in the file.

If you load more bytes of text into memory than your system can hold, these extra bytes "spill" back out to the diskette. You can still edit these files. This is explained in the "Disk Spill" section of this chapter.

What Takes Memory?

The following require memory:

- the number of active files in memory at one time
- the number of lines in a file
- bringing in new files to be edited including internal files
- doing things with marked text
- copying and shifting lines

- adding or inserting lines in a file
- the number of functions assigned to keys in the PE.PRO file

Disk Spill

When you use all the available memory, the Personal Editor uses the diskette on the default drive as a place to store more information.

There are five important concepts about the disk spill process that you should know:

1) Disk spill occurs automatically when the Personal Editor requires more memory than is available — no text is lost.

2) The Personal Editor creates a temporary file called PE.TMP on the diskette to hold additional information from active files.

3) The PE.TMP file can take up to 128KB of space on your diskette.

4) The Personal Editor always requires a minimum amount of memory to operate normally.

5) The Personal Editor performs quickest when all files are able to fit into memory — retrieving from and putting information on the diskette requires more time than doing the same in memory.

The Personal Editor automatically creates on the default drive the PE.TMP file when you either load a file that is too big for memory, or when you do normal editing operations, like copying lines, that fill the available memory. When this happens, the message SPILL FILE CREATED appears on the status line. Do not remove the spill file diskette.

The PE.TMP spill file can be up to 128KB long if there is enough room on your diskette. The spill file uses less if you have less. The PE.TMP file becomes like any other file in that it takes up space on the diskette.

The Personal Editor constantly uses memory. When it needs more memory than is available, it writes a block of lines from an active file to the spill file. It then uses that memory space freed up by the transfer to operate normally.

That is why if you check your memory with the QUESTION MARK command during the spill disk condition, you will see the memory size remains about the same. It needs this minimum amount of memory.

If you use most of your memory, you may notice the Personal Editor performs slower for operations like block shifting.

When you fill the spill file, you get a message saying the SPILL DISK IS FULL. You then have only the minimum amount of memory available. When that is used up, the Personal Editor goes into a memory full condition where you are restricted from normal operation. This is explained in the next section.

When Memory Is Full

When you have used all the available memory, the Personal Editor cannot operate normally. The message, MEMORY IS FULL — REMOVE FILES, appears on the message line. The ϕ character (a circle with a line through it) appears on the status line.

The message leaves after a keystroke, but the special character remains on the status line until memory is available.

The Personal Editor does not allow you to use operations that require additional memory. It gives you a message if you try to do so. Here are the restrictions:

- you cannot move the cursor into the text area

- you cannot use any command that requires memory

Here is what the Personal Editor allows you to do:

- use the EDIT command (without a new filename) to see which files you want to remove from memory

- use the QUIT or FILE command to remove active files from memory

- use the PgUp and PgDn keys to see the bottom of the file that caused the memory to be full. If that file was not entirely loaded into memory, do not use the FILE command or the text that did not appear on the screen will be erased.

You must remove one or more files to free memory. When memory is available, the Personal Editor returns to normal operation. The special character goes away and the message MEMORY FULL CONDITION ENDED appears on the message line.

Chapter 5. Reference

Contents

The Reference chapter is broken into two sections.

The first section describes the Personal Editor commands. The second section describes the Personal Editor functions. Descriptions of the cursor movements and the text mark keys of Lesson One and Lesson Two are in the function section.

The remainder of the book contains appendixes.

The Personal Editor supplies over 60 messages on the message line. What the messages mean and how you can recover from them is explained in Appendix A.

Appendix B gives the Personal Editor PE.PRO file. This lists the keys and the functions assigned to the keys when you first start the Personal Editor.

Appendix C gives you a suggested PE.PRO file that is designed for word processing. If you plan to do more word processing than programming, change your PE.PRO to contain these functions and strings. If you plan to do word processing infrequently, you can create a command file with only these function changes. You would continue to be able to use the keys that are not assigned in this small command file.

Appendix D helps you remember what functions are assigned to the 40 possible function keys (normal function keys, Ctrl function keys, Shift function keys, and Alt function keys). You can fill in the function you assign. This may be helpful if you are an infrequent user of a particular command file. Go to these pages or even remove them when you want to recall your key definitions.

Finally, Appendix E teaches you how to put the DOS startup procedure on your Personal Editor program diskette. When you start the IBM Personal Computer, you then can put the Personal Editor diskette in the drive instead of the DOS diskette. This saves pulling out the DOS diskette and putting in the Personal Editor diskette. It also covers how to back up your program diskette.

Commands

The Personal Editor commands are listed alphabetically.

All commands must be typed on the command line.

The description of each command includes a brief purpose statement, its format, then more detailed remarks. The description ends with references to related commands.

The characters between the angle brackets < > are the options of the Personal Editor commands.

Except for the SAVE, ERASE, RENAME, DIR, and FILE commands, you can abbreviate all commands by their first letter. (The format gives the first letter abbreviation.) Otherwise, you must type out the entire word of the command. Also, for the DEFINE command, you can type "d," "def," or "define."

Commands may be typed in uppercase or lowercase, or in a mixture of cases.

Except fòr the LOCATE and CHANGE commands, all commands must be separated from the next part of the command by one or more blanks.

CHANGE Command

Purpose: The CHANGE command replaces a character or string of characters with another string. You can selectively replace strings or replace them all at one time.

Format: c/*presentstring/newstring/*<—> <*>

Remarks: To replace a string throughout a file (global change), type the asterisk * following the second slash. When you press the Enter key, the Personal Editor finds all occurrences of the present string and replaces them with the new string.

To selectively replace some strings and not others, do not include the asterisk. When you do not include the asterisk then press the Enter key on the command line, the Personal Editor moves the cursor to the first occurrence of the string past the current cursor position.

To change that present string to the new string use the [confirm change] function (Shift-F5).

After you replace the first string, the CHANGE command and the two strings remain on the command line.This allows you to search for the next string without typing the command again. If you do not want to change the next found string, press the Ctrl-Enter keys; the Personal Editor skips over the string and looks for another occurrence of the string.

A character must follow the CHANGE command, the present string, and the new string for the CHANGE command to work. If you do not type the entire word, the CHANGE command reads the first character after the "c" as the separator. It may be a letter, number, or a non-letter character like a parenthesis, a plus sign, or a slash. The same character must follow each part of the command.

The optional minus sign in the CHANGE command controls the direction of the search. If you do not type a minus sign, searching begins from the cursor to the end of the line then to the bottom of the file.

If you type the minus sign, the search begins from the cursor to the beginning of the ~~file~~ line then to the top of the file.

The Personal Editor ignores whether you type the present string in upper, lower or mixed case. For example, if you type the string /MiXeD/, the Personal Editor finds any of these strings: mixed, MIXED, or mIxEd.

The Personal Editor, however, does insert the new string exactly as you type. If the new string is MiXeD, it will appear in the text the same.

Example: Here are some examples of how to use the CHANGE command:

Command	Comments
c(xyz(abc(	Changes xyz to abc individually with a forward search. Use Shift-F5 to replace, and Ctrl-Enter to repeat the command.
c/xyz//*	Deletes all occurrences of xyz (by changing them to the empty string) from the current file position to the end of the file.
C!xyz! xyz !-*	Capitalizes all occurrences of xyz and puts spaces on both sides from the cursor position and the top of the file.
c'/'('*	Changes all occurrences of / to (from the current position in the file and the end of the file.

See also the LOCATE command.

DEFINE Command

Purpose: The DEFINE command assigns a meaning to a key.

Format: def key = <*'literal'*> <*[function]*>

Remarks: The equals sign is required, but the blanks surrounding it are optional. The key may be any one of the keys listed in Appendix B.

Examples of key names are “s-F1” (for Shift-F1), “Ins,” and “C-PgUp” for Ctrl-PgUp.

Key names must all be in lowercase.

You can asign functions or literal strings or a combination of the two.

Functions are the bracketed names in the function reference section such as [left] and [page down]. Literals are character strings enclosed in quotes that by themselves do not do anything except appear either in the text or on the command line.

You may use either single (’) or double (’’) quotes around a literal. Single (double) quotes may not appear inside a character string enclosed in single (double) quotes. (Use a different quote mark inside another quote mark.)

The number of functions plus each literal string character may not exceed 120. For example, there are twelve characters in ’e b:practice’.

Once you assign a function or literal string to a key, typing the key performs the sequence of literals and functions.

The order that you type the functions and literals is the order in which the Personal Editor performs them. Because of the speed at which the Personal Editor operates, the functions and literals seem to perform at the same time. This not only makes your work go faster, but the Personal Editor is able to avoid extra redisplays of the screen.

If you want to remove the definition(s) assigned to a key, type the DEFINE command and the key without a function or literal. For example:

```
d a-t =
```

See also the MACRO command.

DIR Command

Purpose: The DIR command lists either all files from a diskette or specified files.

Format: dir <device:> <*filename* <*.ext*> >

Remarks: The DIR command gives a snapshot of the current files on a diskette just as the .keydefs file gives a snapshot of the current key definitions.

The .DIR file is an internal file that you can rename and save.

You can use the global characters ? and * in the filename and the extension.

Example: If you had a group of files that had similar names except for a few characters, you could type the ? where the filenames vary and receive those files. For example,

```
dir   b:appendx?
```

lists all the files that begin with "appendx" and end with a different character. Here is an example of what might appear in the .DIR file:

```
APPENDXA    14946   11-01-82    4:41p
APPENDXB    13107   11-02-82    3:40p
APPENDXC     2557   11-03-82    3:45a
APPENDXD      187   11-04-82   10:45a
```

An * in a filename or in a filename extension indicates that any character can occupy that position and all the remaining positions in the filename or extension. For example,

```
dir *.bas
```

may give the following:

```
JONATHAN  BAS  20880    11-04-82
CAROLYN   BAS   1720    11-07-82    12:00p
JIMBO     BAS   1880    11-03-82
DONNA     BAS   1865    12-11-82    12:00p
```

EDIT Command

Purpose: The EDIT command switches between one active file and another.

Format: e <device:> <*filename* <.*ext*> <notabs> >

Remarks: If you want to look at or change a file while another file is on the screen, type the EDIT command and the name of the file on the command line.

The Personal Editor first removes the current file from the screen (not memory), remembering the current file's cursor position and Insert or Replace setting. It then displays the new file on the screen.

If the file you want is already active, type "e" and the filename. The Personal Editor restores that file to the screen as it was before you edited the new file.

If the file is not active, the Personal Editor looks for it on the diskette and reads it into memory if it exists. Otherwise, the Personal Editor creates a new file with that name. Remember to include the disk drive if the file you want is not on the default drive.

The Personal Editor assigns the file you get with the EDIT command the same margins, tab settings, and monitor as the current file.

Active files are organized like slides in a slide projector. Each time you use the EDIT command to get a file, that file becomes one slide. If you have more than one file in the “slide projector,” typing “e” on the command line switches from one active file to the next in the order they were loaded.

To quickly scan through active files (slides) one at a time, press the F8 function key. The F8 key is the button you use to advance through the active files.

If you have a two-drive system, you can load a file into memory when you start the Personal Editor. After typing the abbreviation PE, type a space then “b:” followed by the name of the file. When you press the Enter key, the Personal Editor looks for that file if it exists. Otherwise, it creates a new file with that name and drive.

If you have a one-drive system, you can still load a file when you start the Personal Editor but it must be on the Personal Editor program diskette.

When the Personal Editor starts, it expands tab characters into the correct number of blanks to reach the next tab stop, where tab stops are at file columns 9, 17, 25, 33, etc. The Personal Editor does this to fill out the blank spaces that were compressed with the SAVE command.

To defeat tab expansion, type “notabs” after the filename. This is not the usual procedure unless you want compressed text.

ERASE Command

Purpose: The ERASE command deletes a file from diskette.

Format: erase <device:>*filename*<*.ext*>

Remarks: You can use the global characters ? and * in the filename and in the extension. (See the DIR command for details on the global characters.)

You must be careful when you use global characters with the ERASE command. Check to see what files are covered with the global characters. You may unintentionally erase files with similar filenames.

FILE Command

Purpose: The FILE command saves a copy of the current file to diskette and removes it from memory.

Format: file <*filename* <notabs> >

Remarks: The FILE command writes the current file on the screen to the diskette and removes it from memory.

The Personal Editor allows you three options when you quit and save a file with the FILE command.

First, when you type only the word "file" on the command line and press the Enter key, the Personal Editor compresses the file as it saves, then removes the file from memory. (File compression will be explained later.)

Second, when you type the word "file" followed by a new name for the file, the file is then compressed, saved, and quit with that new filename. Otherwise, the FILE command uses the information on the status line as the current filename.

Third, when you type the word "file," a filename, and the word "notabs," the file is saved, quit, but not compressed. In this case, you must include a filename after the FILE command, otherwise, the Personal Editor reads the word "notabs" as the filename.

Compression means eliminating sequences of more than one blank from the text in order to save diskette space. (The Space Bar blank character takes up room on the diskette like any other character.) The Personal Editor compresses the file unless you tell it not to with the notabs option.

When you edit the file again and want to print it, the PRINT command inserts the spaces back into the file. The file is then printed as you typed it originally. This is the usual procedure.

An example of why you would not want to compress the file is if you want to print the file in DOS with the DOS TYPE command.

Here is a chart to help see the relationships:

		CHANGE
	COMPRESSED	NAME
file	YES	NO
file <*filename*>	YES	YES
file *filename* notabs	NO	YES/NO

You may not use the FILE command on internal files (.KEYDEFS, .UNNAMED, or .DIR). With the .KEYDEFS file, you can include a name with the FILE command.

The Personal Editor compresses programs too. To avoid moving literal string constants that would change the performance of the program, for example, the Personal Editor does not compress characters beyond the first quote in a line.

Errors can still occur, however, using Hollerith constants in Fortran FORMAT statements. Use the notabs option to prevent the errors.

See also the NAME and SAVE commands.

LOCATE Command

Purpose: The LOCATE command finds the next (or previous) occurrence of a string from the cursor position.

Format: <1>/*pattern*</<->>

Remarks: The Personal Editor searches for the string and moves the cursor to the first place it is found, displaying a new screen if necessary.

The LOCATE command and the string remain on the command line after you press the Enter key (Ctrl-Enter if you're in the text). This allows you to repeat the search without retyping the command.

The command name or letter abbreviation is optional. You must type the letter L or the word LOCATE if the first character of the search string is a slash (/). If so, type L and use some other character like a dash or period before and after the string.

The optional minus sign in the LOCATE command controls the direction of the search. If the minus sign follows the separator, the search begins at the current file position and searches toward the top of the file. If you do not type it after the second slash, the search direction is forward, toward the bottom of the file.

LOCATE Command

The LOCATE command, like the CHANGE command, ignores whether you type the string in uppercase or lowercase when it searches for the string.

For example, if the string is /MiXeD/, the LOCATE command finds any of the strings: mixed, MIXED, or mIxEd.

Each entry in the following table is a legal LOCATE command.

Command	**Comments**
/xyz	Searches forward for xyz.
/xyz/-	Searches backward for xyz.
L'/xyz'	Searches forward for /xyz.
l'/'-	Searches backward for /.
/xyz abc/	Searches forward for xyz abc.
/-xyz--	Searches backward for xyz.

See also the CHANGE command.

MACRO Command

Purpose: The MACRO command reads in a file of commands and performs each of the commands in the order entered.

Format: m <device: >*filename*<*.ext*>

Remarks: To understand how the MACRO command works, let's compare it with a similar process the Personal Editor does.

When you press the Enter key after the Personal Editor copyright screen, the Personal Editor searches for a file called PE.PRO (for Personal Editor profile) on the default drive. The Personal Editor reads this file of DEFINE commands and assigns the key definitions to specific keys. (See Appendix B for the PE.PRO file.)

The PE.PRO file does not have to be limited to DEFINE commands. You can put all the Personal Editor commands in the PE.PRO file. For example, you can put the SET MARGINS and SET TABS command in to get the margins and tabs you want without having to type them each time.

The PE.PRO file may not contain other MACRO commands.

With the MACRO command you can design your own PE.PRO file to help edit different types of jobs.

The three ways you can do this are listed in Lesson 4.

For example, here is a profile for writing memos:

def a-f1 = [cursor command] 'set margins 1 80' [execute] (sets margins for Date, To, From and Subject)

def enter = [insert line] [insert line]
(defines Enter to skip two lines to start next paragraph)

def a-F2 = [cursor command] 's margins 10 70 15' [execute] (sets margins for the body of the memo)

Save the file. Go to the file where you want to type your memo and type "macro memo" on the command line. The Personal Editor reads these three new definitions overlaying other key definitions if necessary. These functions now become active.

Remember you can have only one DEFINE command per line. Also, if the Personal Editor finds an error in any of the commands in the file, it stops reading the file and displays an error on the message line.

To see if the key definitions you asked for with the c-f1 keys are assigned, type the EDIT command followed by the internal filename, .keydefs. This file displays the current key definitions.

See also the DEFINE and EDIT commands.

NAME Command

Purpose: The NAME command changes the name of the current file.

Format: n *filename*

Remarks: The Personal Editor checks to see if the name you typed is the same as another active filename, or if the name begins with a period. If so, you get an error message. If not, the new name appears on the status line.

You also get a message if you try to save a file with an invalid DOS name.

When you save a file, the updated file replaces the old file with the same name.

See also the EDIT command.

PRINT Command

Purpose: The PRINT command prints the current file to the printer.

Format: p

Remarks: You can pause the printer by holding down the Ctrl key and pressing the Num Lock key. The printer pauses after the current line is printed. Press any key to continue.

You can stop the printer by holding down the Ctrl key and pressing the Scroll Lock-Break key. The printer stops after the current line is printed.

If the printer is turned off, out of paper, or not on-line, the following message appears:

No printer; type r or c (resume/cancel)

Check for these possible conditions and correct them.

Position the paper before you use the PRINT command. The file is printed as it appears on the screen, including blanks at the top and bottom of the file.

QUESTION MARK Command

Purpose: The QUESTION MARK command displays the requested information.

Format:

```
? tabs
? margins
? key <keyname>
? memory
```

Remarks: You type the question mark and the word on the command line.

When you type "? tabs", or "? margins" and press the Enter key, the current tab settings or margin settings display on the command line. For example:

```
set tabs    9  8  25  33  41  49  57  65  73  81  89
set margins    1  254  1
```

When this information appears on the command line, you can change it on the command line. Move the cursor to the number and change it.

When you press the Enter key, the command leaves the command line and that new information becomes the current setting. Pressing the F5 function key clears the information from the command line.

QUESTION MARK Command

With "? key <*keyname*>", you type the key after the word KEY. For example:

```
? key enter
? key a-f4
? key pgup
```

The key and the function assigned to it (if any) is displayed on the command line as it appears in your .KEYDEFS file.

You can also change this information as long as the changes are allowable Personal Editor words.

Typing "? memory" on the command line gives you the current amount of bytes available in memory. The information appears on the message line. Pressing any key removes the message. You cannot change this information.

QUIT Command

Purpose: The QUIT command stops editing of a file and removes the file from memory without writing it to diskette.

Format: q

Remarks: If the file was not changed and you use the QUIT command, the Personal Editor erases the copy of the current file in memory and shows another file in memory (if any) without using the EDIT command.

If you modified the current file or have changed its name since the last time the file was saved, the Personal Editor writes the messages "Are you sure? Type y or n" in the status area.

If you want to start over by removing the command, type N for NO. Typing Y for YES erases any changes you made since the last save.

When you use the QUIT command with the last active file, the Personal Editor returns to DOS.

RENAME Command

Purpose: The RENAME command changes the name of a file on the specified disk drive.

Format: rename < device: > *oldname.ext newname.ext*

Remarks: The RENAME command is different than the NAME command in that the RENAME command changes the name of a file without you having to save the changed name.

The new name does not appear on the status line. You must use the NAME command or use the EDIT command to have the new filename appear on the status line.

You can use the global characters * and ? with the RENAME command. (See the global character details in the DIR command.)

Example: The command

```
rename program.bas weight.bas
```

renames the file PROGRAM.BAS on the default drive to WEIGHT.BAS.

The command

```
rename b:text.doc    *
```

renames the file TEXT.DOC on drive B to TEXT.

SAVE Command

Purpose: The SAVE command writes a copy of the current file to the diskette.

Format: save <*filename*<*.ext*> <notabs> >

Remarks: If you do not type a filename after the command, the Personal Editor writes the current file on the screen to the diskette.

If you do give a filename whether it's new or the one on the status line, the current file is saved under that name. In either case, the file is not removed from memory.

The Personal Editor allows you three options, like the FILE command, when you save a file with the SAVE command.

First, when you type only the word "save" on the command line, the Personal Editor saves and compresses the current file on the diskette. (File compression will be explained later.) Nothing changes on the screen.

Second, when you type the word "save" followed by a new name, the file is saved and compressed with that new filename.

Third, when you type the word “save,” a filename, and the word “notabs,” the file is saved, but not compressed. You must include a filename after the SAVE command, or else the Personal Editor reads the word “notabs” as the filename.

Compression means eliminating blank spaces from the file in order to save diskette space. The Personal Editor does compress Space Bar “blank characters.” It does not compress single blanks.

The Personal Editor compresses the file unless you tell it not to with the notabs option.

When you want to print the file using the PRINT command, the PRINT command inserts the blank spaces back into the file. The file is then printed as you typed it originally. This is the usual procedure.

An example of why you would not want to compress the file is if you wanted to print the file in DOS with the DOS TYPE command.

Here is a chart to help see the relationships:

	COMPRESSED	CHANGE NAME
save	YES	NO
save *filename*	YES	YES
save *filename* notabs	NO	YES/NO

When the Personal Editor compresses programs, it avoids moving literal string constants that would change the performance of the program. It does this by not compressing characters beyond the first quote in a line.

Errors can still occur, however, when using Hollerith constants in Fortran FORMAT statements. Use the notabs option to prevent the errors.

You may not use the FILE command on internal files (.KEYDEFS, .UNNAMED, or .DIR). With the .KEYDEFS file, you must change the name before you can use the FILE command.

See also the FILE command.

SET DISPLAY Command

Purpose: The SET DISPLAY command changes the display of the current file.

Format:

```
s display mono
s display color 40
s display color 80
s display b/w 40
s display b/w 80
```

Remarks: The Personal Editor moves the current file to another monitor. If you have two monitors connected to your system, the Personal Editor switches the current file from the monochrome display to or from a display attached to the color/graphics adapter.

Each of the active files remembers on which monitor it is active. As you scroll through the active files using the EDIT command with no filenames (or F8), the Personal Editor changes to the necessary monitor to display each file.

Only one display can be active at one time. The active display contains the blinking cursor.

SET DISPLAY does not allow you to change monitors if you do not have the correct adapter installed on your IBM Personal Computer.

SET DISPLAY Command

If you have a monitor other than the IBM Monochrome Display connected to the IBM Personal Computer when you start the Personal Editor, the Personal Editor uses that display.

The way the Personal Editor shows marked areas, the command line, and so on, depends on which monitor the file is active. The table below describes individual characteristics of the different monitors:

Field type	Mono	Color 40 or 80	B/W 40 or 80
Data area	white on black	white on black	white on black
Command line	black on white	blue on white	black on white
Status area line	white on black	white on black	white on black
Message line	bright white	red on black	white on black
Marked area	white on black	white on black	white on black
Current position	bright underline	blue on black	blinking white on black
Current position in mark	black underline	blue on white	blinking black on white

See also the EDIT command

SET MARGINS Command

Purpose: The SET MARGINS command marks the boundaries so you can type without pressing the Enter key for a carriage return. You also reformat text inside these boundaries.

Format: s margins n n <n>

Remarks: The three numbers in the command are the left margin number, the right margin, and the optional paragraph margin. If you do not give the paragraph margin, the Personal Editor uses the same value as the left margin.

The right margin must be larger than both the left margin and the paragraph margin. The paragraph margin can be less than the left margin if the left margin is greater than one.

The Personal Editor starts with a left margin of 1, a right margin of 254, and a paragraph margin of 1. To change the margins, type the SET MARGINS command and the margin numbers on the command line and press the Enter key.

When you use the EDIT command to create a new file or get an old one, the margin settings for that file will be the margin settings for whatever file was on the screen when you typed the EDIT command.

You cannot save the margin settings of a file. They have to be reset each time you start the Personal Editor. To save you time from typing, you can either assign margins to a key, or put the SET MARGINS command in the PE.PRO file.

For example, you can assign a key,

```
def s-f10 = 's margins 5 65 10' [execute]
```

or on one line in the PE.PRO file,

```
set margins 5 65 10
```

The Personal Editor has word wrapping (also referred to as power typing). If you are typing text and there is a nonblank character past the right margin as part of a word or the beginning of a word, the Personal Editor moves that word or character to either:

1) a new line, or

2) the beginning of the next line if there is room on the line for additional characters.

This is helpful if you're typing documents.

When you reflow a paragraph, all three margin settings control the reformatting.

SET TABS Command

Purpose: The SET TABS command allows you to change the tab markers from the default setting.

Format: s tabs t1 t2 t3 t4 . . . t20

Remarks: You can assign 20 tabs to a file between column 1 and 255.

Those 20 tab settings become the tab settings for all active files.

The numbers must follow in increasing order. The second number has to be larger than the first, and the third larger than the second, and so on.

Pressing the Tab key moves the cursor to the tab settings. Holding down the Shift key and pressing the Tab key moves the cursor left along the tabs.

If the Tab key is assigned the [tab word] function and the Shift-Tab keys are assigned the [backtab word] function, the tab settings are ignored.

To see the current tab settings, use the QUESTION MARK command and the word "tabs" on the command line.

See [tab], [backtab], [tab word], [backtab word], and the QUESTION MARK command.

Notes

Functions

The Personal Editor functions are listed alphabetically.

The description of a Personal Editor function is similar to the Command Reference section.

Purpose describes the function briefly. Default means that is the key the function is assigned when you start the Personal Editor for the first time. (Later you may want to change the default settings for the functions.) Remarks is a more detailed description of the function.

At the end of each description are other related functions.

The names of the default keys are spelled out for easy reading. Use "s" for Shift key, "c" for Ctrl key, and "a" for Alt key with the DEFINE command.

You must include the square brackets when you use a function or you will get an error message.

See Appendix B and Appendix C for examples of combining Personal Editor functions to do one task.

BACKTAB Function

Purpose: The [backtab] function moves the cursor left to the previous tab stop.

Default: Shift-Tab keys

Remarks: Tab stops are initially set at every 8 columns beginning after column 1 at column 9, 17, 25, 33, 41, 49, etc.

You may also redefine the tab settings with the SET TABS command.

When the cursor moves left with the [backtab] function, it stops at the first tab setting it finds. Continuing to use the [backtab] function moves the cursor to the other tab settings.

For example, on the default settings, if the cursor is at column 94 and you press Shift-Tab, the cursor stops at column 89, or 5 columns left. After that the cursor moves eight columns left or at column 81, 73, and so on.

If you press the Shift-Tab keys to move the cursor left off the screen (when possible), the Personal Editor displays approximately an additional half screen of columns.

To see what the tab settings are, use the QUESTION MARK command for tabs on the command line.

See also [tab], [backtab word], SET TABS command and the QUESTION MARK command.

BACKTAB WORD Function

Purpose: The [backtab word] function moves the cursor to the left, stopping at the first character in the next word.

Default: none

Remarks: A word in this context is a group of characters separated by blanks.

The cursor moves to the first column if the line is blank or if the first word in the sentence is indented.

If necessary, the Personal Editor displays 20 or 40 new columns (depending on the monitor size) if the cursor moves to the left off the screen.

To see what the tab settings are, use the QUESTION MARK command for tabs on the command line.

See also [tab], [tab word], and [backtab].

BEGIN LINE Function

Purpose: The [begin line] function moves the cursor to the first column of the current line.

Default: Home key

Remarks: If the cursor is not in the first 80 columns (40 for a 40 column screen) of a file, the Personal Editor displays a new screen so that the cursor appears in the first column of the file in the first column of the screen.

If the cursor is already in the first column of the current line, [begin line] does nothing.

See also [end line].

BEGIN MARK Function

Purpose: The [begin mark] function moves the cursor to the beginning of the marked area.

Default: Alt-Y keys

Remarks: For block marks and character marks, the [begin mark] function moves the cursor to the top left position of the marked area.

For line marks, the [begin mark] function moves the cursor vertically to the top line of the marked area without changing its column position. This includes if the cursor was on the command line.

If necessary, the Personal Editor changes the display of the current file so that the marked area in that file appears on the screen.

This function also searches other active files to find the marked area. Once found, that file and its marked area appear on the screen and become the current file.

One use for the [begin mark] (or [end mark]) function is as a search marker. Mark a character (line or block) in a file you refer to often. Edit other files. Whenever you want to go to that marked file immediately, press the Alt-Y keys. If you do this, however, other files cannot be marked.

See also [end mark].

BOTTOM
Function

Purpose: The [bottom] function moves the cursor to the last line of the file.

Default: Ctrl-End keys

Remarks: When you press the Ctrl-End keys, the Personal Editor displays the Bottom of file marker on the last line of the screen.

If the last line of the file is already on the screen, the cursor moves down to the last line.

If the cursor is on the command line, the cursor will move to the last line of the file.

See also [top].

BOTTOM EDGE Function

Purpose: The [bottom edge] function moves the cursor down to the last row of the current text area.

Default: Ctrl-PgDn keys

Remarks: When the cursor moves to the last line of the text area, the contents of the current screen do not change.

If the cursor is on the command line, it moves to the last line of the text area.

See also [top edge].

CENTER LINE
Function

Purpose: The [center line] function moves the line with the cursor to the middle of the screen.

Default: none

Remarks: The line with the cursor moves to row 10 of the screen.

Regardless if the cursor is closest to the Top of file or Bottom of file markers, that line is moved to row 10.

The row and column indicators on the status line do not change.

The [center line] function does not work on the command line.

COMMAND TOGGLE Function

Purpose: The [command toggle] function switches the cursor from the text area to the command line and back again.

Default: Esc key

Remarks: If the cursor is on the command line when you press the Esc key, the cursor moves to the current file position in the text area.

If the cursor is in the text and you press the Esc key, the cursor moves to the command line leaving a highlighted underscore (and letter) where the cursor was.

If the cursor leaves a marked area, the location of the cursor in the area becomes unmarked.

See also [cursor command] and [cursor data].

CONFIRM CHANGE Function

Purpose: The [confirm change] function replaces a character or a group of characters with another character or group of characters when you search through a file with the CHANGE command.

Default: Shift-F5 keys

Remarks: This function works only when you search for individual occurrences of the strings with the CHANGE command, not for global changes.

The Personal Editor displays an error message if you try to use this function any time other than immediately after an [execute] function of the CHANGE command.

See the CHANGE command.

COPY MARK Function

Purpose: The [copy mark] function copies a marked area at the location of the cursor leaving the original copy marked.

Default: Alt-Z keys

Remarks: For marked lines, the Personal Editor inserts a copy of the lines below the line with the cursor.

For block marks, the cursor position marks the top left corner of the inserted block. Any text to the right of the cursor moves to the right of the inserted text.

For character marks, the Personal Editor inserts a copy of the marked area at the cursor location.

One of the many uses for the [copy mark] is for printing part of a file. If you have a long file, but only want to print a part of it, mark a copy of the text you want, insert it into a new file, then print the new file.

Another example is copying text from one file to another. This is convenient if you have similar text you want in more than one file. Mark an area in one active file, then switch to another file. Move the cursor to the place where you want to include the text from the first file, then press the Alt-Z keys.

For all cases of [copy mark], you are not allowed to copy a marked area directly onto or overlap itself.

See also [mark block], [mark char], [move mark] and [mark line].

CURSOR COMMAND Function

Purpose: The [cursor command] function unconditionally moves the cursor to the command line.

Default: none

Remarks: If the cursor is already on the command line when you use the [cursor command], nothing happens.

If the cursor is in the text, this function moves the cursor to the command line. The Personal Editor highlights the cursor's original text position on the screen.

The [cursor command] function is primarily intended for use when you combine functions and assign them to a key. See Appendix B and Appendix C for examples.

This function works exactly the opposite of the [cursor data] function.

See also [command toggle] and [cursor data].

CURSOR DATA
Function

Purpose: The [cursor data] function moves the cursor from the command line to the highlighted cursor position in the text area.

Default: none

Remarks: If the cursor is on the command line, the [cursor data] function returns the cursor to its original highlighted location on the screen. The highlighting for that location is then turned off and the cursor resumes its normal blinking mode.

If the cursor is already in the text area when you use [cursor data], nothing happens.

The [cursor data] function is primarily for combining functions and assigning them to a key. See Alt-P in Appendix B for an example.

This function works exactly the opposite of the [cursor command] function.

See also [command toggle] and [cursor command].

DELETE CHAR Function

Purpose: The [delete char] function deletes the character at the cursor and shifts the remaining characters to the right of the cursor one position left.

Default: Del key

Remarks: The [delete char] is most useful for correcting mistakes in existing text.

The [rubout] function (on the Backspace key) which deletes the character to the left of the cursor location is intended more for correcting typing mistakes immediately after they occur.

See also [rubout].

DELETE LINE Function

Purpose: The [delete line] function deletes the current line and shifts the text below up one line.

Default: Ctrl-Backspace keys

Remarks: The Personal Editor shifts all lines below the deleted line up one line. The cursor remains stationary.

This function does nothing if the cursor is on the command line.

It is possible to delete all but one line of a file. The Personal Editor always provides one line between the Top of file marker and the Bottom of file marker.

Use the [delete mark] function to erase many lines of text.

See also [insert line] and the ERASE command.

DELETE MARK Function

Purpose: The [delete mark] function deletes the text in the marked area and unmarks the file.

Default: Alt-D keys

Remarks: The [delete mark] function erases text from the three types of marks differently. For line marks, Personal Editor erases the marked line or group of lines and closes the gap.

For block marks, the [delete mark] acts as if you performed the [delete char] function for each character inside the mark. If any text is to the right of the marked block, it is shifted left by the same number of columns of text that were erased.

Lines of text below deleted marked blocks do not move up.

For character marks, lines contained within the marked area are erased. The marked areas on the first and last lines of the marked area are deleted as with the [erase end line] and [erase begin line] functions.

If characters remain on the beginning and end lines of the marked area, the Personal Editor joins the two line fragments together.

See also [unmark], [mark block], [mark char], and [mark line].

DOWN
Function

Purpose: The [down] function moves the cursor one row towards the bottom of the file.

Default: Cursor Down key

Remarks: If you press the Cursor Down key when the cursor is on the last line of the screen, the Personal Editor scrolls the screen up by one line, bringing a new line into view.

When the cursor is on the command line, pressing the [down] function key scrolls the file up by one line, then moves the cursor to the new line.

You must use the [command toggle] or [cursor command] functions to return the cursor to the command line. If the cursor is already on the last line of the file when you press the Cursor Down key, nothing happens.

See also [up] and [down4].

DOWN4 Function

Purpose: The [down4] function moves the cursor four rows toward the bottom of the file.

Default: none

Remarks: If you use the [down4] function when the cursor is located on the command line or in the bottom three rows of the current screen, the function shifts the appropriate number of lines up into view. The cursor appears on the bottom row of the text area.

If the cursor is within the bottom three lines of the file, the Personal Editor moves the cursor to the bottom line of the file, shifting the text up if necessary.

See also [up4] and [down].

END LINE Function

Purpose: The [end line] function moves the cursor after the last character on the current line.

Default: End key

Remarks: If the last column of a line is not on the screen, Personal Editor adjusts the file so that the last column of the line is shown in approximately the middle of the screen.

If the cursor is beyond the end of the current line when you press the End key, the cursor moves back to the last character on that line.

If the cursor is already at the end of the current line, the [end line] function does nothing. If the cursor is on a blank line, it moves to the first column.

See also [begin line].

END MARK Function

Purpose: The [end mark] function moves the cursor to the end of the marked area.

Default: Alt-E keys

Remarks: For block and character marks, the [end mark] function moves the cursor to the exact character that is at the end of the mark.

For line marks, the cursor does not change its column location. The cursor just jumps vertically to the row containing the last line of the marked area.

The Personal Editor displays a new screen if necessary to show the end of a marked area. The [end mark] function also switches from a current file to another active file to display a marked area.

See also [begin mark].

ERASE BEGIN LINE
Function

Purpose: The [erase begin line] function erases characters from the cursor to the beginning of the current line.

Default: none

Remarks: The cursor does not move. The remaining text moves to the beginning of the line.

See also [erase end line].

ERASE END LINE
Function

Purpose: The [erase end line] function erases characters from the cursor to the end of the current line.

Default: F6 key

Remarks: The new length of the current line becomes one less than what is indicated by the column number because of the cursor position.

The cursor does not move.

See also [erase begin line].

ESCAPE Function

Purpose: The [escape] function allows you to type ASCII characters on the screen.

Default: Alt-X keys

Remarks: If you use the [escape] function followed by any other character, Personal Editor inserts or replaces the character according to the current setting of the Insert toggle, even if that character has some control function assigned to it.

For example, to put form feed characters in your document to break pages for printing, you would press the Alt-X keys then continue to hold down the Alt key and type 012. When you lift your finger off the Alt key, the form feed character appears at the cursor position.

You can assign ASCII strings to keys. For example,

```
def a-4 = ’ √ ’
```

The square root figure is an ASCII 251 character.

You have to use two right or two left quotes, or the double quotes. You cannot mix quote marks.

See the IBM BASIC manual for a list of ASCII characters and the decimal numbers for each.

EXECUTE Function

Purpose: The [execute] function does the command on the command line, even if the cursor is not currently on the command line.

Default: Ctrl-Enter keys

Remarks: The Enter key always performs the [execute] function when the cursor is on the command line, even if some other function has been assigned to it.

The [execute] function is very useful when you use it with the CHANGE and LOCATE commands. Since these commands do not erase from the command line after working, you can press the Ctrl-Enter keys to repeat the operation.

FILL MARK Function

Purpose: The [fill mark] function fills a marked area with a character.

Default: Alt-F keys

Remarks: When you press the Alt-F keys, the Personal Editor waits for you to type a character which it uses as a fill character.

The Personal Editor then duplicates that one character throughout the marked area.

You may use the [escape] function before entering the fill character in order to use a control key as the fill character.

See also [escape], [mark block], [mark char], and [mark line].

FIND BLANK LINE Function

Purpose: The [find blank line] function searches forward in the file for a blank line.

Default: none

Remarks: If the cursor is on a blank line, the Personal Editor does not move the cursor.

If the current line is not blank, the Personal Editor searches for the next line that is blank. If the blank line is not found, you get a message indicating that the remainder of the file contains no blank lines.

When using the Personal Editor for word processing, paragraphs must be separated by blank lines. Otherwise, when you reflow a paragraph, the entire text becomes one paragraph.

See also [reflow].

FIRST NONBLANK Function

Purpose: The [first nonblank] function moves the cursor to the first character of the current line that is not a blank.

Default: none

Remarks: If there are no nonblank characters on the current line, [first nonblank] moves the cursor to the beginning of the current line. This function is helpful for properly indenting program text.

INDENT Function

Purpose: The [indent] function moves the cursor to the paragraph margin.

Default: none

Remarks: The cursor indents to the paragraph margin only if the line above the cursor is blank, or the cursor is on the first line of the file. Otherwise, the cursor moves to the left margin.

If there is no paragraph margin, the cursor moves to the left margin.

One way to take advantage of the [indent] function is to combine it with some other function and assign it to the Enter key.

When you first use the Personal Editor, the Enter key is assigned the [begin line] [down] [indent] functions to move the cursor down to the beginning of the next line. This is designed for programmers.

If you want to do word processing with paragraph margins, assign the [insert line] and [indent] functions to the Enter key. (See Appendix C.) You would use the Enter key then to end a paragraph. Pressing the Enter key a second time indents the cursor to the paragraph margin.

INSERT LINE Function

Purpose: The [insert line] function adds a new line below the current line.

Default: F9 key

Remarks: Inserting a new line causes the Personal Editor to shift down any text below the current line. The cursor moves to the first column of the new line.

A way to insert a blank line above text that is at the top of the file is to use the [split] function with the cursor in the first column of the first line.

The [insert line] function does nothing if the cursor is on the command line.

See also [delete line].

INSERT MODE
Function

Purpose: The [insert mode] function unconditionally sets the status line to Insert.

Default: none

Remarks: Regardless of its previous setting, the [insert mode] function puts the Personal Editor into Insert mode, expands the cursor to a rectangle, and puts the word Insert on the status line.

The [insert mode] function is primarily intended for use in macros.

See also [insert toggle] and [replace mode].

INSERT TOGGLE Function

Purpose: The [insert toggle] function switches the typing style from Insert to Replace and back again.

Default: Ins key

Remarks: If in Replace, the [insert toggle] function switches you to Insert. If in Insert, the [insert toggle] function switches you to Replace.

The Personal Editor displays the typing style in two ways: by the message in the status line and by the size of the cursor.

While in Insert, the cursor expands from its usual appearance as a blinking underscore to a blinking rectangle.

The [insert toggle] function controls how characters are entered. In Replace, characters replace the old characters at the cursor position by typing over them.

In Insert, characters are inserted at the cursor position and characters to the right of the cursor are shifted one position to the right for each character inserted.

See also [insert mode] and [replace mode].

JOIN Function

Purpose: The [join] function connects the line below the current line to the current line.

Default: Alt-J keys

Remarks: The cursor may be located anywhere on the current line and does not move after you connect the two lines.

The [join] function erases a blank line between two lines.

This function does nothing if the current line is the command line or is the last line of the current file.

If the combined line is longer than 255 characters, the Personal Editor erases any character after 255 and displays "Line(s) truncated" on message line.

You may recover the old contents out of the .UNNAMED internal file.

See also [split].

LEFT Function

Purpose: The [left] function moves the cursor one position to the left.

Default: Cursor Left key

Remarks: If moving left causes the cursor to move off the screen, the Personal Editor displays a new screen where the cursor position is near the middle column of the screen.

If the cursor is already in column 1 of the current line, the [left] function does nothing.

See also [right], [left8], and [left40].

LEFT8 Function

Purpose: The [left8] function moves the cursor eight positions to the left.

Default: none

Remarks: This function allows faster horizontal scrolling than the [left] function.

If moving the cursor left 8 spaces causes the cursor to move off the screen, the Personal Editor displays a new screen with the cursor approximately in the middle column of the screen.

If the cursor is already in the first 7 columns of the current line, the [left8] function moves the cursor to column 1.

See also [right8] and [left].

LEFT40 Function

Purpose: The [left40] function moves the cursor 40 positions to the left.

Default: Ctrl-Left key

Remarks: This function allows faster horizontal scrolling than the [left] and [left8] functions.

If moving the cursor left 40 spaces causes the cursor to move off the screen, the Personal Editor displays a new screen with the cursor approximately in the middle column of the screen.

If the cursor is already in the first 40 columns of the current line, [left40] moves the cursor to column 1.

See also [right40], [left8], and [left].

LEFT EDGE Function

Purpose: The [left edge] function moves the cursor to the leftmost column of the current screen.

Default: none

Remarks: The cursor moves straight to the left and the Personal Editor does not display a new screen.

Since the command line scrolls independently of the text area, the left edge column number of the command line may not be the left edge column number of the text area.

See also [right edge].

LOWERCASE
Function

Purpose: The [lowercase] function makes all uppercase letters within a marked area lowercase.

Default: none

Remarks: No other arrangement of the text occurs.

If you try to change text without marking an area, you will get a "No marked area" message.

See also [uppercase].

MARK BLOCK Function

Purpose: The [mark block] function tags one corner of a block mark.

Default: Alt-B keys

Remarks: Tag means to indicate one corner of a rectangle of text.

To mark a block, you must tag two corners of an area.

When you hold down the Alt key and press "b" for the [mark block] function, the Personal Editor displays the character at the cursor in reverse video. (The cursor must be in the text area when you use the [mark block] function.)

When you press the keys for the second [mark block], the Personal Editor highlights the rectangle marked by the two diagonally opposite corners.

You may tag an area by either the top left and lower right corners, or the top right and lower left corners.

Two block marks also may be in the same column or line. Marking lines and columns is convenient for moving columns of a table.

An error message appears if there are already two marks in the current file or another active file and you use the [mark block] function. An error message also occurs if you use a different second mark function, like the [mark line] function.

See also [unmark], [delete mark], [copy mark], [move mark], and [overlay block].

MARK CHAR Function

Purpose: The [mark char] function tags one position of a sentence or character.

Default: Alt-C keys

Remarks: Tag means to highlight a character or the beginning or ending of a sentence, phrase, or word.

You need two marked characters to highlight a sentence, word, or phrase.

If there are no marks in any active file and you use the [mark char] function, the Personal Editor highlights the character at the cursor in reverse video. (The cursor must be in the text area when you use [mark char].)

When the second character mark is placed in the current file, the Personal Editor highlights all characters between and including the two character marks. No position after the last character of a line is highlighted.

Contrast this with the highlighting for line marks, which extends to the end of the screen regardless of the line length.

Gaps appear in the marked area if a blank line is between character tags. This unmarked line is included in any operation you do to the marked area.

MARK CHAR Function

An error message appears if there are already two marks in the current file or in another active file and you use the [mark char] function. An error message also occurs if you use a second different mark function, like the [mark block] function, instead of a [mark char] function.

Character marks are most useful for working with text files. For example, you may mark a phrase of a sentence (or an entire sentence) that may extend across several lines and copy it into the middle of another sentence.

See also [unmark], [delete mark], and [copy mark].

MARK LINE
Function

Purpose: The [mark line] function tags a line of the current file.

Default: Alt-L keys

Remarks: Tag means to indicate a highlighted line.

If no marks exist in any active file and you use the [mark line] function, the Personal Editor displays the current line in reverse video. (The cursor must be in the text area when you use the [mark line] function.)

When you tag a second line in the current file (above or below the first line), the Personal Editor displays in reverse video the current line and all lines between it and the other marked line out to column 255.

An error message occurs if you use a second different mark function with [mark line], like the [mark block] or [mark char] function.

You will probably find line marks particularly useful for moving and copying pieces of program source.

See also [unmark], [delete mark], and [copy mark].

MOVE MARK Function

Purpose: The [move mark] function inserts a marked area below the cursor position, deleting the original marked area.

Default: Alt-M keys

Remarks: For line marks, the [move mark] function inserts the marked text below the cursor and moves any text below the cursor down to make room for the text. The original line marked area is erased.

For block marks, the [move mark] function uses the cursor position as the top left corner of the area to insert. If any text exists at the new location, it moves to the right.

For character marks, the [move mark] function inserts the marked area in front of the cursor. If the marked area was above the moved location, the newly rearranged text moves up to fill in the erased lines. If below the moved area, the text below the erased lines moves up.

OVERLAY BLOCK Function

Purpose: The [overlay block] function puts a block marked area on top of any existing text.

Default: Alt-O keys

Remarks: The [overlay block] function works for block marked areas only.

For block marked areas, the Personal Editor places the marked rectangle on top of an area of the same size, deleting any characters in that area.

The position of the cursor marks the top left corner of the block to be overlayed.

If you hold down the Alt key then press "o" for the [overlay block] function and the marked area is not a block marked area, you will get a message telling you a block mark is required.

PAGE DOWN Function

Purpose: The [page down] function scrolls the display down to the next page.

Default: PgDn key

Remarks: The Personal Editor brings up 20 new lines from the text below the current screen, overlapping the bottom 2 lines from the current screen to the top of the new screen. The total is 22 lines of text.

If there are less than 20 lines below the last line of the current screen, the Personal Editor displays the last 21 lines of the file and the Bottom of file marker.

The cursor does not move when you use the [page down] function. (Unless the file is less than 22 lines long.) This means that the sequence of the [page down] function followed by [page up] function leaves the cursor in its original position unless the cursor is at the top or bottom of the file.

See also [page up].

PAGE UP Function

Purpose: The [page up] function scrolls the display up to the previous page.

Default: PgUp key

Remarks: The Personal Editor brings down 20 new lines from the text above the current screen overlapping the top two lines from the current screen to the bottom of the new screen. The total is 22 lines of text.

If there are less than 20 lines above the top row on the screen, the Personal Editor displays the first 21 lines of the file and the Top of file marker.

The cursor does not move when you use the [page up] function (unless the file is less than 22 lines). This implies that the sequence of the [page up] function followed by [page down] function leaves the cursor in its original position unless the cursor is at the top or bottom of the file.

See also [page down].

REDRAW Function

Purpose: The [redraw] function redraws the screen.

Default: Alt-R keys

Remarks: The Personal Editor erases the entire screen, then completely redraws it.

This function is useful to recover the proper screen contents after a message from DOS rearranges the screen format.

REFLOW Function

Purpose: The [reflow] function reformats text to the set margins within a line marked area.

Default: Shift-F3 keys

Remarks: The [reflow] function reformats line marked areas only.

For the first line of the reformatted area only, the Personal Editor uses the current paragraph margin setting as the left margin.

The Personal Editor fills lines with words where possible without splitting them at the end of the line. If you are typing a word that extends beyond the right margin setting, the Personal Editor moves the entire word down to the next line after it inserts a line.

If a single word is too long to fit between the left and right margins, the Personal Editor splits it at the right margin.

The Personal Editor does not automatically right-justify a paragraph.

The Personal Editor provides the [find blank line] function specifically to allow paragraphs to be located, marked, and reflowed. These functions are combined on the Alt-P keys.

See also the SET MARGINS command.

REPLACE MODE Function

Purpose: The [replace mode] function unconditionally sets the status line to Replace.

Default: none

Remarks: Regardless of its previous setting, [replace mode] puts the Personal Editor into Replace, changes the cursor to a blinking dash, and puts the word Replace on the status line.

The [replace mode] function is primarily intended for assigning functions to keys to do a single task.

See also [insert toggle] and [insert mode].

RIGHT Function

Purpose: The [right] function moves the cursor one position to the right.

Default: Cursor Right key

Remarks: If moving right causes the cursor to move off the screen, the Personal Editor displays a new screen with the cursor position near the middle column of the screen.

If the cursor is already in column 255 of the current line, the [right] function does nothing.

See also [left], [right40], and [right8].

RIGHT8 Function

Purpose: The [right8] function moves the cursor 8 positions to the right.

Default: none

Remarks: This function allows faster horizontal scrolling than the [right] function.

If moving the cursor right 8 spaces causes the cursor to move off the screen, the Personal Editor displays a new screen with the cursor approximately in the middle column of the screen.

If the cursor is already in the last 8 columns of the current line, the [right8] function moves the cursor out toward the 255 column limit.

See also [left8], [right40], and [right].

RIGHT40 Function

Purpose: The [right40] function moves the cursor 40 positions to the right.

Default: Ctrl-Right arrow keys

Remarks: This function allows faster horizontal scrolling than the [right] and [right8] function.

If moving the cursor right 40 spaces causes the cursor to move off the screen, the Personal Editor displays a new screen with the cursor in that screen.

If the cursor is already in the last 40 columns of the current line, the [right40] function moves the cursor out toward the 255 column limit.

See also [left40], [right8], and [right].

RIGHT EDGE
Function

Purpose: The [right edge] function moves the cursor to the rightmost column of the line on the current screen.

Default: none

Remarks: The cursor moves straight to the right and the Personal Editor does not display a new screen.

Since the command line scrolls independent of the text area, the right edge column number of the command line may not be the right edge column number of the text area.

Under no circumstances does the [right edge] function move the cursor to a screen position with a column number greater than 255.

See also [left edge].

RUBOUT Function

Purpose: The [rubout] function deletes the character to the left of the cursor and shifts the remaining characters on the current line one position to the left to close up the gap.

Default: Backspace key

Remarks: If the cursor is in column one of the current line, the [rubout] function deletes the first character on the line and shifts the remaining characters left by one position.

The [rubout] function is useful for correcting mistakes immediately after they have been typed. You should use the [delete char] function for correcting existing text.

See also [delete char].

SHIFT LEFT Function

Purpose: The [shift left] function shifts the text in a marked area one position to the left.

Default: Shift-F7 keys

Remarks: When you use the [shift left] function, the Personal Editor deletes the first character of each line of the marked area.

This function is most useful for shifting program text with line marks or moving a column of a table with block marks.

For large marked areas, the [shift left] function can require a noticeable amount of time.

In some cases, the Personal Editor does not redisplay the screen immediately after each shift, or it may not redisplay the screen at all until you release the key(s) assigned to the [shift left] function.

See also [shift right].

SHIFT RIGHT Function

Purpose: The [shift right] function shifts the text in a marked area one position to the right.

Default: Shift-F8 keys

Remarks: When you use the [shift right] function, the Personal Editor inserts a blank before the first marked character on each line of the marked area.

This function is most useful for shifting program text with line marks or moving a column of a table with block marks.

For large marked areas, the [shift right] function can require a noticeable amount of time.

In some cases, the Personal Editor does not redisplay the screen immediately after each shift, or it may not redisplay the screen at all until you release the key(s) assigned to the [shift left] function.

See also [shift left].

SPLIT Function

Purpose: The [split] function divides the current line at the cursor.

Default: Alt-S keys

Remarks: The Personal Editor moves text at and to the right of the cursor on the current line to the beginning of a new line. The Personal Editor inserts a new line below the current to place the remainder of the divided line. The cursor does not move.

This function does nothing if the current line is the command line.

One way to insert a line before the beginning of the file is to use [split] when the cursor is positioned on the first character of the file.

See also [join] function.

TAB Function

Purpose: The [tab] function moves the cursor to the right to the next tab stop.

Default: Tab key

Remarks: Tab stops are originally set at every 8 columns beginning after column 1 at column 9, 17, 25, 33, 41, 49, etc.

You may also redefine the tab settings with the SET TABS command.

When the cursor moves right with the [tab] function, it finds the first tab stop and continues to move to each tab stop.

For example, if the cursor is at column 30 and you press the Tab key, the cursor stops at column 33, or 3 columns right. After that the cursor moves 8 columns right or to column 41, 49, 57, and so on.

If you press the Tab key to move the cursor off the right side of the screen (when possible), the Personal Editor scrolls approximately an additional half screen of columns.

To see what the tab settings are, use the QUESTION MARK command for tabs on the command line.

See also [backtab] and [tab word].

TAB WORD Function

Purpose: The [tab word] function moves the cursor to the right, stopping at the first character in the next word.

Default: none

Remarks: A word is defined to be a group of non-blank characters separated by blanks.

If the cursor is at or beyond the end of the current line, pressing the [tab word] function does not do anything.

If the cursor moves to the right off the screen with the [tab word] function, the Personal Editor displays 20 or 40 new columns (depending on the monitor size).

See also [backtab word] and [tab].

TOP Function

Purpose: The [top] function moves the cursor to the first line of the file.

Default: Ctrl-Home keys

Remarks: When you press the Ctrl-Home keys, the Personal Editor shifts the file so that the Top of file marker is on the top line of the screen.

If the first line of the file is already on the first line, nothing happens.

See also [bottom].

TOP EDGE
Function

Purpose: The [top edge] function moves the cursor to the top line of the current screen.

Default: Ctrl-PgUp keys

Remarks: The cursor moves up, and the contents of the current screen do not change.

See also [bottom edge].

UNDO
Function

Purpose: The [undo] function restores what you had on a line before you began to change it.

Default: Shift-F4 keys

Remarks: The [undo] function recovers an erased line or words as long as the cursor does not leave the current line. Once the cursor leaves the line or joins two lines, you cannot restore that line.

To restore a lost line, see the .UNNAMED internal file in Chapter 4 Special Topics.

UNMARK Function

Purpose: The [unmark] function removes any existing marks.

Default: Alt-U keys

Remarks: The [unmark] function removes all marks that exist in any active file. No change occurs to the text inside of the marked area.

If a marked area is on the screen, the Personal Editor removes the reverse video highlighting of the area. You may use the [unmark] function no matter where the cursor is located.

See also [mark block], [mark char], and [mark line].

UP Function

Purpose: The [up] function moves the cursor one position towards the top of the screen.

Default: Cursor Up key

Remarks: If the cursor is at the top of the file when you use the [up] function, nothing happens.

If the cursor is at the top of the screen but not at the top of the file, the cursor remains stationary and the text on the screen shifts down by one line, bringing a new line into view.

If you use the [up] function on the command line, the cursor moves to the bottom line of the text area. You must use the [cursor toggle] or [cursor command] functions to return the cursor to the command line.

See also [down] and [up4].

UPPERCASE Function

Purpose: The [uppercase] function changes all lowercase letters in a marked area to uppercase letters.

Default: none

Remarks: The Personal Editor does not alter any characters within the marked area that are not lowercase letters.

See also [lowercase].

UP4 Function

Purpose: The [up4] function moves the cursor four positions toward the top of the file.

Default: none

Remarks: If you use the [up4] function when the cursor is located in the top three rows of the current screen, the function shifts the appropriate number of lines down into view. The cursor appears on the top line.

If the cursor is within three top lines of the file, the Personal Editor moves the cursor to the top line of the file, shifting the text down if necessary.

See also [down4] and [up].

Notes

Appendix A. Messages

The messages the Personal Editor produces and their meanings are summarized below:

Are you sure? Type y or n

The Personal Editor displays this message when you attempt to use the QUIT command for a recently changed file. Typing "y" throws away all unsaved changes. Typing "n" cancels the QUIT command.

Block mark required

The Personal Editor will not allow you to use the [overlay block] function unless you use the [mark block] function to mark the area.

Unmark the current file and use the Alt-b [mark block] function.

Cannot open spill file

You requested an operation which required more memory than was currently available. The Personal Editor tried to open a spill file on the default drive but was unsuccessful, possibly because the drive was not ready.

Cannot name ".unnamed"

You are not allowed to change the name of the internal file, ".unnamed."

If you want to save the text in .UNNAMED, use the [mark line] and [copy mark] functions to copy text from .UNNAMED to some other active file, which you may then save.

Cannot save internal file

You are not allowed to use the SAVE or FILE command on the three internal files, .DIR, .UNNAMED, or .KEYDEFS. You can rename .KEYDEFS and .DIR to legal DOS filenames, however, then save them.

The renaming may be explicit, using the NAME command, or implicit, by specifying a new filename as part of the SAVE or FILE command.

Cannot use while memory is full

You tried to use a key that requires memory. When the memory is full, the Personal Editor only allows you to use keys that do not require more memory. Use the QUIT command to remove the current file, or save and quit another active file.

Use the QUESTION MARK command for memory to occasionally check the amount of memory you have.

Color/graphics adapter not installed

You entered a SET DISPLAY command specifying color or b/w, but there was no color/graphics display adapter attached or it is not operational.

Command file not found

This message can occur in two contexts. When the Personal Editor first loads, it searches for a command file named PE.PRO on the default drive. If PE.PRO file does not exist on that drive, the Personal Editor produces this message.

The Personal Editor also displays this message if it could not find the file named in a MACRO command.

Command files nested too deep

A command file is a list of commands that are executed with a MACRO command like the Personal Editor PE.PRO file.

A command file may not contain MACRO commands.

Confirm change

The individual string you were looking for with the CHANGE command was found.

You replace the target string with the replacement string with the [confirm change] function. If the next key you type does not perform [confirm change], you will lose the opportunity to substitute the replacement string for this occurrence of the target string.

Conflict with existing mark

The second of a pair of marks conflicts with the first. Both members of the pair of marks must be in the same file and be the same type (both character marks, line marks, or block marks).

Disk error in directory

You left the drive door open or the disk was not found or formatted. Close the door or get a formatted disk.

Disk is full

When the Personal Editor tried to write a file onto a diskette, it discovered that there was not enough space remaining for the file to fit. The Personal Editor leaves the partially written file on the diskette.

You can replace the diskette that is full with one having enough space remaining to contain your file, then try the SAVE or FILE command again. You can also use the ERASE command to permanently delete files from your disk to make room. Then save the file again.

Error closing file

A file was not closed properly, either because of disk errors or because of a full disk. The changed/ unchanged status of the file will not be altered.

You should try writing the file to a different disk or on a different drive.

Error in display type

The format of the SET DISPLAY command is "set display (type)," where "type" is either "mono," "color 40," "color 80," "b/w 40," or "b/w 80."

Error in margin settings

The format of the SET MARGINS command is "set margins n1 n2 n3," where n1, n2, and n3 are numbers from 1 to 254. The third number is optional. The second number must be larger than either of the other two numbers.

Error in notabs specification

You did not type the full word "notabs," misspelled it when using the SAVE, EDIT, or the FILE command, or you left a space between the drive name and the filename. Type the correct word.

Error in replacement string

In a CHANGE command, the Personal Editor could not determine what you intended as the replacement string. You must type all three characters that indicate the beginning and end of the target and replacement strings.

If you typed the name of the CHANGE command as anything other than "c" or "change," the Personal Editor uses the second character you typed (probably "h") as the separator character, with unpredictable results.

Error in search pattern

For a CHANGE or LOCATE command, the Personal Editor could not determine what to use as a search pattern. See the comments for the previous message, as well as the examples of the CHANGE and LOCATE commands in the chapter on commands.

Error in tab settings

You typed an unknown setting like a letter, or a number bigger than 255. Also, the second tab setting must be larger than the first, the third larger than the second, and so on.

Error opening file

A file could not be opened because the filename was not a legal DOS filename, the drive door was open, the diskette was damaged, or the diskette was unformatted.

Error reading command file

The Personal Editor could not read the file named in a MACRO command or the file PE.PRO, probably due to some problems with the disk.

Error reading file

An error occurred while reading a file or you opened the drive door when the light was on. The file was not read past the point of the error. You may retry the operation on this or some other drive.

Error writing file

A file was not written properly because of disk errors, the disk was full, or you opened the drive door when you saved a file. The changed/unchanged status of the file is not altered.

You should try writing the file to a different disk or on a different drive.

File not found

A file specified in an ERASE or RENAME command could not be found on the diskette.

Invalid function

An invalid function name was typed as a key definition, either as part of a DEFINE command or in a command file read by the MACRO command. Remember that function names must be typed exactly as they appear in this book, in brackets and in lower-case.

Invalid key

An invalid key name was typed in a key definition either as part of a DEFINE command or in a command file. Also, you may have not typed a key after the "? key" command. The valid key names are listed in Appendix B.

Invalid name

The name typed in a NAME or EDIT command is invalid. Files may not be given names beginning with a period (.), since they could be confused with internal files.

Key not defined

A key was pressed that has no function or macro associated with it.

Line mark required

The [reflow] function operates only on areas marked with line marks, and no such area exists. If you have marked the area to be reformatted with character or block marks, [unmark] the area, then remark it using the [mark line] function before retrying the [reflow] operation.

Line(s) truncated

The Personal Editor erased characters past column 254 when you tried to join two lines, shift, or change text. Go to the .UNNAMED internal file to recover the lines that were erased.

Macro truncated

A list of functions in a DEFINE command or in a command file is too long. Macros are restricted to a length of 120 tokens, where each function counts as one token and each literal string character counts as one token.

Marked block already exists

Only one marked area may exist at a time, among all active files. Use the [unmark] function to remove the existing mark.

Max spill file size reached

The maximum spill file size is 128KB. This limit has been reached. Remove files from memory.

Memory full condition ended

You removed a file from memory during the memory full condition and now have memory space. Check the available memory with the QUESTION MARK command. You may only have a small amount if you removed a small file. The same memory shortage may occur.

Memory is full — remove files

The Personal Editor does not have any more memory to put information. The character "ϕ" appears on the status line until you remove files from memory.

Use the QUIT or FILE command on a file other than the current file to create memory space. If loading the current file caused this message, it may not all appear as an active file because of the lack of memory. Saving the current file would then erase what did not appear.

Missing file name

You tried to use the SAVE or FILE command on a file that doesn't have a filename, or you didn't type a name after the NAME command.

The NAME command requires a filename which must be separated from the command name by at least one blank.

Missing quote or]

You did not type a quote in a literal string or an end bracket with a function. Insert the correct character then press the Enter key.

Monochrome display not installed

You entered a SET DISPLAY command specifying the monochrome display, but there is no monochrome display adapter attached or it is not operational.

Name already in use

You may not use the NAME command to assign the name of another active file to the current file.

New file

The file you named with the EDIT command is not the name of an internal file, nor is it the name of an active file, and it cannot be found on the requested diskette. The Personal Editor therefore assumes it to be a new file. The file will not be created on the diskette until it is written using the SAVE or FILE commands.

No = in definition

Key definitions must contain an equal sign to separate the key name from the function or literal strings.

No change pending

You tried to perform the [confirm change] function, but the previous keystroke did not execute the CHANGE command. You may not use any commands or functions between the individual change and [confirm change] or you will get this message. Move the cursor above the last occurrence and try it again.

No macro definition

The DEFINE command expects to find a key name, an equal sign, and either a function or a literal string after the command name.

No marked area

You asked for a function that requires a marked area, but no marked area currently exists.

No printer; type r or c (resume/cancel)

You used the PRINT command and either the printer was not turned on, not on-line, or not connected properly.

Only "r" or "c" keys work. Get the printer ready then press the "r" key, or press "c" to cancel the PRINT command.

Not enough memory — press any key

You tried to load the Personal Editor but there wasn't enough room for it to operate. Pressing any key takes you back out of the Personal Editor and returns you to DOS.

Not found

The Personal Editor could not find the search string you asked for in LOCATE or CHANGE command, or it could not find a blank line for the [find blank line] function.

You used the Alt-p keys to reflow a paragraph and there was no blank line between the last line of the paragraph and the Bottom of file marker.

Set option not recognized

The Personal Editor interpreted the command you entered as a SET command, but did not find a valid set option. The only valid set options are "DISPLAY" "TABS," and "MARGINS."

The Personal Editor also produces this message if there are not enough arguments to a SET command. If you meant to enter the SAVE command but forgot to spell out the entire name, the Personal Editor thinks you meant to type one of the SET commands and displays this message.

Source and target conflict

You attempted a mark function that would have copied the marked area at least partially on top of itself. Since the meaning of this is unclear for certain types of marks, the Personal Editor will not allow you to overlap marked areas.

Spill disk is full

The diskette which contains the spill file is full. Although this file can be up to 128KB, there was not enough room on the diskette. Remove a file from memory with the QUIT or FILE command. Removing files from memory frees space in the spill file.

If loading the current file caused this message, it may not all have been loaded into memory as an active file because of the lack of spill file and memory space. Saving the current file would then erase what did not get loaded into memory.

Spill error; type r or c (retry/cancel)

The Personal Editor tried to create a spill file on the disk drive and the door wasn't closed, there was no disk to spill to, or the diskette was write-protected. Also, you may have replaced the diskette with the spill file PE.TMP with another diskette.

Type "r" if the situation is corrected, or "c" will cancel the spill and replace the data with blanks.

Spill file created

You ran out of memory so the Personal Editor created a new file, called PE.TMP, on the diskette to hold any new information for active files. When the spill file is full (maximum of 128KB), you get another message.

Spilled lines lost — blanks substituted

The Personal Editor was unable to recover lines previously written to the spill file, possibly because of a diskette error or because you were unable to successfully respond to the SPILL ERROR message. As a result, blank lines were substituted in your active file for the unrecoverable text. The last saved version of the file, however, remains unchanged.

Too many files

Not counting the .UNNAMED file, there may only be a maximum of 20 active files at once. Use the FILE or QUIT commands to reduce the number of active files, then retry the EDIT or DIR command that caused this message.

Too many tab stops

You typed more than 20 tab settings for the SET TABS command. Erase tab settings until there are 20 or less.

Type a character

The Personal Editor waits for you to type a character or a number after the [fill mark] function or the [escape] function.

Unknown ? option

You did not type one of the four possible options correctly. The options are “KEY,” “MARGINS,” “TABS,” or “MEMORY.”

Unknown command

An unrecognized command was entered on the command line. Remember that most commands can be abbreviated to their first letter, but the SAVE, DIR, ERASE, RENAME, and FILE commands must be spelled out in full.

Write protect error

The Personal Editor attempted to write text to a diskette but could not because the diskette was write-protected.

Appendix B. Personal Editor Profile*

def up = [up]	
def down = [down]	
def left = [left]	
def right = [right]	
def pgup = [page up]	
def pgdn = [page down]	
def home = [begin line]	
def end = [end line]	
def ins = [insert toggle]	
def del = [delete char]	
def enter = [begin line] [down] [indent]	Moves cursor to paragraph setting of next line
def backspace = [rubout]	
def esc = [command toggle]	
def tab = [tab]	
def f1 = [cursor command] [begin line] [erase end line] 'e pe.hlp' [execute]	Gets you the Help Menu

def f2 = [cursor command] [begin line] [erase end line] ’save’	Saves the current file
def f3 = [cursor command] [begin line] [erase end line] ’file’	Saves and quits file
def f4 = [cursor command] [begin line] [erase end line] ’quit’ [execute]	Quits the current file
def f5 = [begin line] [erase end line]	Clears the current line
def f6 = [erase end line]	
def f7 = [cursor command] [begin line] [erase end line] ’print’	Prints the current file
def f8 = [cursor command] [begin line] [erase end line] ’e’ [execute]	Switches active files
def f9 = [insert line]	
def f10 = [insert line] [up] [first nonblank] [down]	Inserts a line and indents cursor
def c-left = [left40]	
def c-right = [right40]	
def c-pgup = [top edge]	
def c-pgdn = [bottom edge]	
def c-home = [top]	
def c-end = [bottom]	

def c-enter = [execute]	
def c-backspace = [delete line]	
def c-f1 =	
def c-f2 =	
def c-f3 =	
def c-f4 =	
def c-f5 =	
def c-f6 =	
def c-f7 =	
def c-f8 =	
def c-f9 =	
def c-f10 =	
def s-tab = [backtab]	
def s-f1 = [page down] [bottom edge] [down] [down] [cursor command]	Scrolls down Help Menu
def s-f2 = [page up] [top edge] [up] [up] [cursor command]	Scrolls up Help Menu
def s-f3 = [reflow]	
def s-f4 = [undo]	
def s-f5 = [confirm change]	
def s-f6 =	

def s-f7 = [shift left]	
def s-f8 = [shift right]	
def s-f9 = [cursor command] [begin line] [erase end line] ’dir a:’ [execute]	Directory of drive A
def s-f10 = [cursor command] [begin line] [erase end line] ’dir b:’ [execute]	Directory of drive B
def a-a =	
def a-b = [mark block]	
def a-c = [mark char]	
def a-d = [delete mark]	
def a-e = [end mark]	
def a-f = [fill mark]	
def a-g =	
def a-h =	
def a-i =	
def a-j = [join]	
def a-k =	
def a-l = [mark line]	
def a-m = [move mark]	
def a-n =	
def a-o = [overlay block]	

def a-p = [cursor data] [unmark] [mark line] [find blank line] [up] [mark line] [reflow] [end mark] [down] [down] [unmark]	Reformats paragraph within set margins
def a-q =	
def a-r = [redraw]	
def a-s = [split]	
def a-t =	
def a-u = [unmark]	
def a-v =	
def a-w =	
def a-x = [escape]	
def a-y = [begin mark]	
def a-z = [copy mark]	
def a-f1 =	
def a-f2 =	
def a-f3 =	
def a-f4 =	
def a-f5 =	
def a-f6 =	
def a-f7 =	
def a-f8 =	

def a-f9 =	
def a-f10 =	
def a-0 =	
def a-1 =	
def a-2 =	
def a-3 =	
def a-4 =	
def a-5 =	
def a-6 =	
def a-7 =	
def a-8 =	
def a-9 =	

*All functions assigned to a key must be typed on the same line.

Appendix C. Word Processing Profile*

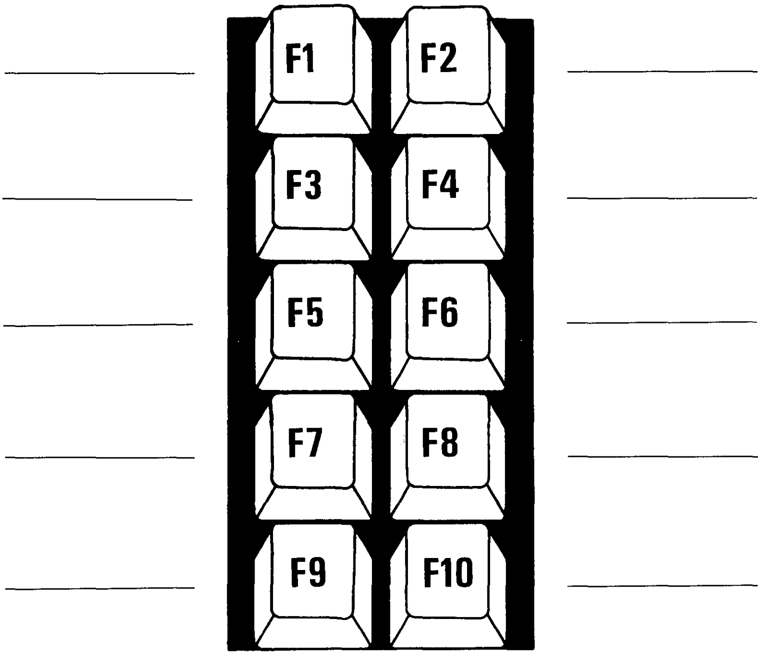
def home = [begin line] [first nonblank]	Moves to first character
def enter = [insert line] [indent]	
def tab = [tab word]	
def f1 = [cursor command] [begin line] [erase end line] 'save'	Saves current file
def f2 = [up4]	
def f3 = [undo]	
def f4 = [down4]	
def f5 = [left8]	
def f6 = [right8]	
def f7 = [unmark] [mark char] [tab word] [left] [mark char] [begin mark] [delete mark]	Deletes a word
def f8 = [cursor command] [begin line] [erase end line] 'e' [execute]	Switches active files
def f9 = [erase begin line]	
def f10 = [erase end line]	
def s-tab = [backtab word]	
def a-a = [begin mark]	
def a-f = [cursor command] [begin line] [erase end line] 'file'	Quits and saves file
def a-q = [cursor command] [begin line] [erase end line] 'q' [execute]	Quits current line
def a-y = [copy mark]	
def a-z = [end mark]	

*All functions assigned to a key must be typed on one line.

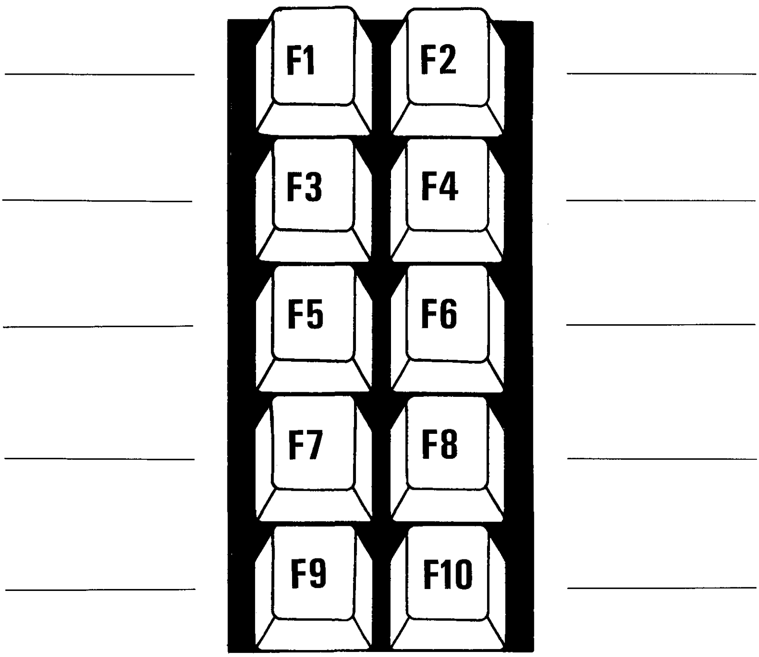
Notes

Appendix D. Function Key Fill-In Sheets

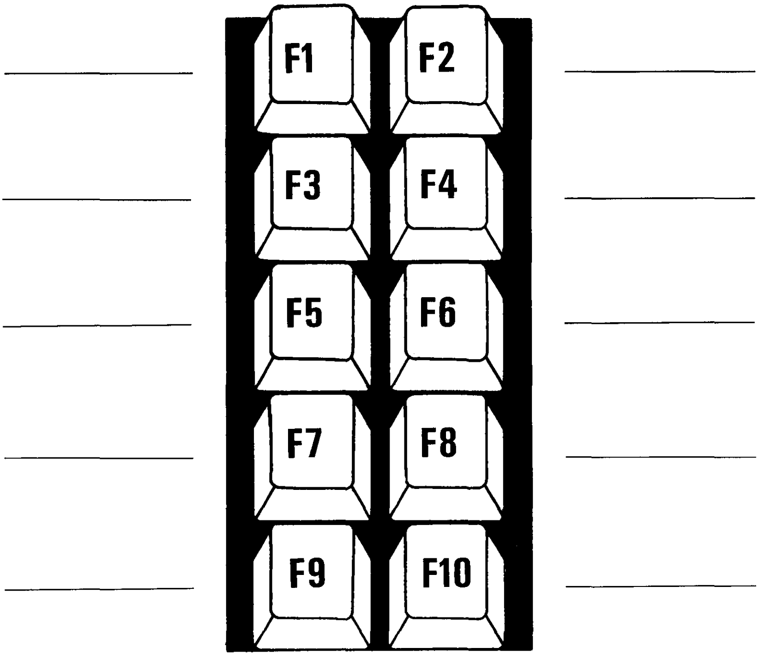
Normal



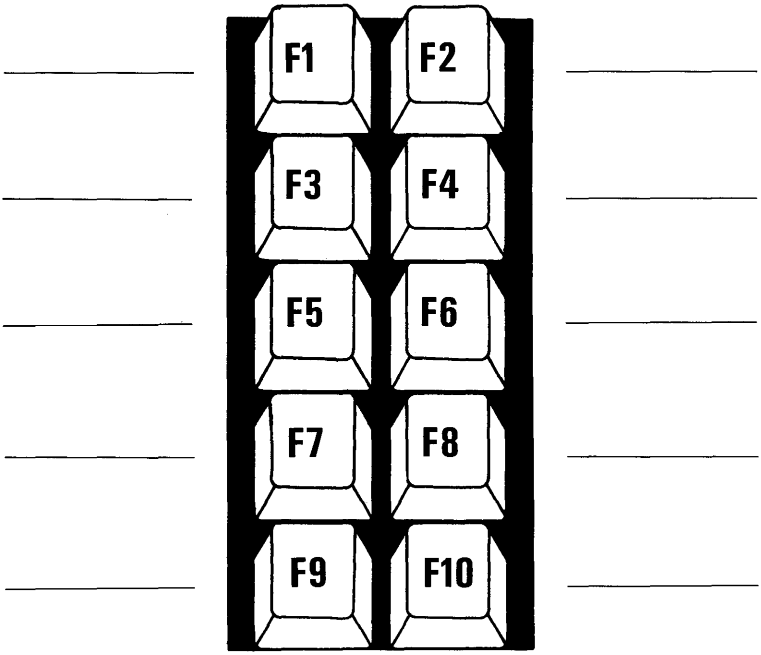
Ctrl



Shift



Alt



Appendix E. Making Backups and Copying DOS

It's a good idea to make a backup copy of the Personal Editor program diskette just in case the diskette is ever damaged or destroyed. This procedure is discussed in the Making Backups section below.

You can also put the Disk Operating System (DOS) on your Personal Editor diskette. Doing this eliminates using the DOS diskette when you start the IBM Personal Computer. This is covered in the Copying DOS section.

Making Backups

To make a backup copy of the Personal Editor diskette, follow the directions carefully:

For a two-drive system

1) Have the DOS diskette in drive A with the DOS prompt for drive A on the screen.

2) After the DOS prompt, type

```
diskcopy a: b:
```

and press the Enter key. The DOS message appears on the screen.

```
Insert source diskette in drive A:

Insert target diskette in drive B:

Strike any key when ready
```

3) Remove the DOS diskette.

4) Put the Personal Editor diskette in drive A, and a blank DOS formatted diskette in drive B.

5) Now strike any key. In a moment, a message tells you the diskette copy is complete.

6) Put your original Personal Editor diskette away in a safe place where it cannot be damaged. Go to the next section on copying DOS onto your diskette.

For a one-drive system (read carefully)

1) Same as above.

2) After the DOS prompt, type

diskcopy

and press the Enter key. The DOS message appears on the screen. Do not touch any key yet. We will in a moment.

Insert source diskette in drive A:

Strike any key when ready

3) Remove the DOS diskette and put in the Personal Editor diskette. Strike any key. The IBM Personal Computer reads the first half of the diskette. A new message appears:

Insert target diskette in drive A:

Strike any key when ready

4) Remove the Personal Editor diskette and put in the blank DOS formatted diskette. Strike any key.

5) In a moment, the first message returns. Repeat steps 3 and 4 following the instructions until you get the message "Copy Complete."

6) Put your original Personal Editor diskette away in a safe place where it cannot be damaged. Go to the next section on copying DOS onto your diskette.

How To Copy DOS

Here is how you copy DOS onto the backup Personal Editor program diskette you just made.

For a two-drive system

1) Have your DOS diskette in drive A with the DOS prompt for drive A on the screen.

2) Remove the write-protect tab from the left edge of the Personal Editor diskette. Don't throw it away. We'll put it back on in a moment.

3) Put the Personal Editor diskette in drive B.

4) After the DOS prompt, type

B:SETUP

and press the Enter key. The setup procedure takes a few moments.

5) When the setup procedure is complete, you should see the following on your screen:

```
A>b: setup
A>sys b:
System transferred
A>copy a: command.com b:
        1 File(s) copied

A>
```

6) Put the write-protect tab back on the diskette.

For a one-drive system

1) Have the DOS prompt on the screen.

2) Same as above

3) Put the Personal Editor diskette in drive A.

4) After the DOS prompt, type

B:SETUP

and press the Enter key. Even though you do not have a drive B, you must make the operating system think it does.

Note: In the following procedure, when the message asks you to insert the diskette for drive A, insert the DOS diskette. When it asks you to insert the diskette for drive B, insert the Personal Editor diskette.

5) The first message reads

```
Insert diskette for drive A: and strike any key
when ready
```

Insert the DOS diskette and press the Enter key.

6) The second message reads

```
Insert diskette for drive B: and strike any key
when ready
```

Insert the Personal Editor diskette and press the Enter key.

7) The third message reads

```
System transferred
A>copy a: command.com b:

Insert diskette for drive A: and strike any key
when ready
```

Insert the DOS diskette and press the Enter key.

8) The fourth message reads

```
Insert diskette for drive B: and strike any key
when ready
```

Insert the Personal Editor diskette and press the Enter key.

9) The last message says

```
        1 File(s) copied

A>
A>
```

DOS is now loaded onto your backup diskette. Use this diskette for everyday operation. Remember to keep the original in a safe place in case anything happens to the backup copy.

Index

Reader's Comment Form

Personal Editor 6936761

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your Authorized IBM Personal Computer Dealer

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Reader's Comment Form

Personal Editor 6936761

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your Authorized IBM Personal Computer Dealer

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Reader's Comment Form

Personal Editor 6936761

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your Authorized IBM Personal Computer Dealer

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Continued from inside front cover

SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

IBM does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free.

However, IBM warrants the diskette(s) or cassette(s) on which the program is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your receipt.

LIMITATIONS OF REMEDIES

IBM's entire liability and your exclusive remedy shall be:

1. the replacement of any diskette(s) or cassette(s) not meeting IBM's "Limited Warranty" and which is returned to IBM or an authorized IBM PERSONAL COMPUTER dealer with a copy of your receipt, or
2. if IBM or the dealer is unable to deliver a replacement diskette(s) or cassette(s) which is free of defects in materials or workmanship, you may terminate this Agreement by returning the program and your money will be refunded.

IN NO EVENT WILL IBM BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAM EVEN IF IBM OR AN AUTHORIZED IBM PERSONAL COMPUTER DEALER HAS BEEN ADVISED OF THE POSSIBLITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

GENERAL

You may not sublicense, assign or transfer the license or the program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be governed by the laws of the State of Florida.

Should you have any questions concerning this Agreement, you may contact IBM by writing to IBM Personal Computer, Sales and Service, P.O. Box 1328–W, Boca Raton, Florida 33432.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

International Business Machines Corporation

P.O. Box 1328-C
Boca Raton, Florida 33432

6936761
Printed in United States of America

IBM

Personal Computer
Professional Series

Personal Editor

6936761