MANUAL CUSTOMIZATION PACKAGE

For the Portable PLUS by Hewlett-Packard

Your Series 100/BASIC manual originally was designed for use with the HP 150 and The Portable (HP 110). To use this manual with your Portable PLUS, replace the pages noted in the table below with the corresponding pages in this package.

Remove Page:	Add Page:
Table of Contents, ii	Table of Contents, ii (Portable PLUS)
1-1, 1-2	1-1 (Portable PLUS), 1-2 (Portable PLUS)
1-13, 1-14	1-13 (Portable PLUS), 1-14 (Portable PLUS)
1-21, 1-22	1-21 (Portable PLUS), 1-22 (Portable PLUS)
5-7, 5-8	5-7 (Portable PLUS), 5-8 (Portable PLUS)
B-1, B-2	B-1 (Portable PLUS), B-2 (Portable PLUS)

82862-90001

Printed in U.S.A. 6/85

Table of Contents

Preface x
Manual Organization x
Notation Conventions xi

Chapter 1: Getting Started

The Series 100/BASIC User 1-1
Before You Begin 1-1
Starting BASIC 1-2
Modes of Operation 1-2
  Direct Mode 1-2
    Quick Computation 1-3
  Indirect Mode 1-3
    Line Format 1-4
Character Set 1-5
Entering a Program 1-8
Modifying a Program 1-9
  Edit Mode 1-9
    Edit Mode Subcommands 1-9
    Entering Edit Mode from a Syntax Error 1-14
  Modify Mode 1-14
    Using Modify Mode 1-15
    Start of Text Pointer 1-16
Error Messages 1-18

Chapter 1: Getting Started

Documenting Your Program 1-18
Printing Operations 1-19
L Commands and Statements 1-19
Writing a Simple Program 1-20

Chapter 2: Data, Variables, and Operators

Introduction 2-1
Constants 2-2
Single and Double Precision Form for Numeric Constants 2-3
Variables 2-3
Variable Names and Declaration Characters 2-4
Special Type Declaration Characters 2-4
Reserved Words 2-4
String Variables 2-5
Numeric Variables 2-5
Array Variables 2-6
Type Conversion 2-7
Expressions and Operators 2-10
Arithmetic Operators 2-10
Integer Division and Modulus Arithmetic 2-11
Overflow and Division by Zero 2-12
Relational Operators 2-12
Logical Operators 2-13
Functional Operators 2-16
String Operations 2-17
Concatenation 2-17
Comparisons 2-17

Chapter 3: The BASIC Environment

Introduction 3-1
BASIC 3-2

Chapter 1

GETTING STARTED

The Series 100/BASIC User

To use Series 100/BASIC successfully, you only need to be familiar with general programming concepts and the BASIC language. If you are not familiar with BASIC, we recommend that you either read one of the introductory texts on programming in BASIC or take a beginning-level course on this language.

Before You Begin

If you have not yet installed the Series 100/BASIC software module into the software drawer in your computer, you need to do that now. Here's the procedure:

Step 1. If your computer's Edisc has any files that you don't want to lose, copy those files to an external disc. (To learn how to copy files from Edisc, refer to the section on "Copying Discs" in Chapter 5 of *Using the Portable PLUS*.)

Step 2. Install the BASIC software module in your HP 82982A Software Drawer. Just follow the *Software Module Installation Instructions* that came with your BASIC software package.

Step 3. Install the software drawer in a drawer receptacle on the bottom of your computer. Follow the *Drawer Installation Instructions* that came with your software drawer.

Starting BASIC

You start Series 100/BASIC just as you would any other application program on your Portable PLUS. First, go to the main P.A.M. screen on your computer. Move the display pointer over to the box labeled BASIC and then press Start Applic (f1).

Modes of Operation

Once you start BASIC, it shows you a symbol like this: Ok. This symbol, called a **prompt**, means that the BASIC interpreter is waiting for you to tell it what to do. This condition, where BASIC shows you a prompt and you respond, is called the **command level**. BASIC will remain at the command level until you enter a RUN command.

At the command level, you may converse with the BASIC interpreter in one of two modes: Direct Mode or Indirect Mode.

Direct Mode

In Direct Mode, you do not precede BASIC statements or functions with line numbers. Rather, you "talk" interactively with the BASIC interpreter, and it executes each instruction as you enter it.

For example,

```
Ok
PRINT "HELLO MOM" Return
HELLO MOM
Ok
```

Direct Mode is useful for debugging programs and for quick computations. You may use Direct Mode to display the results of mathematical and logical operations (using PRINT statements, as in the preceding example) or to store the results for later use (using the LET statement). However, the instructions that produce these results are lost after the interpreter executes the instruction.

Ending and Restarting Edit Mode

Return — Pressing the Return key prints the remainder of the line, saves any changes you have made, and exits Edit mode.

E — The E subcommand saves any changes you have made and exits Edit mode.

Q — The Q subcommand exits Edit mode without saving any changes that you made to the line during Edit mode.

L — The L subcommand lists the remainder of the line, saves any changes that you made, and repositions the cursor at the beginning of the line. Edit mode remains active. (You usually would use this subcommand to list a line when you first enter Edit mode.)

A — The A subcommand restores the line to its original state (cancels any changes) and repositions the cursor at the beginning of the line so you can start again.

Ctrl A — Simultaneously pressing the Ctrl and A keys takes you into Edit mode on the line that you are currently typing. BASIC executes a carriage return, prints an exclamation point (!) and a space, and positions the cursor at the first character in the line. You may now enter any Edit mode subcommand.

NOTE

If you have just entered a line and decide you want to edit it, just type `EDIT.`. BASIC takes you into Edit mode at that line. In this context, the period (.) is a special symbol that refers to the line you just entered.

When BASIC receives an unrecognizable command or illegal character while in Edit mode, it ignores the command and sends a Control-G ("Bell" character) to ring your computer's bell.

Entering Edit Mode From A Syntax Error

When BASIC encounters a syntax error while executing a program, it automatically takes you into Edit mode at the line that caused the error. For example:

```
10 K=2(4) Return
RUN Return
Syntax error in 10
Ok
10
```

When this happens, modify the line to correct the error and then either press Return or use the E subcommand to exit Edit mode. However, modifying the line this way destroys all variable values. If you want to preserve the variable values, first exit Edit mode with the Q subcommand. BASIC will go back to the command level where you can examine the variable values.

Modify Mode

You cannot use Modify mode with your Portable PLUS. This section applies only to the HP 150.

With the HP 150, you may use Modify mode to edit program lines with a minimum of typing:

- LIST the lines of the program you want to edit.
- Enter Modify mode (as described in the next subsection).
- Move the cursor to the first line you want to modify.
- Use the keyboard's character editing keys to modify the line.
- Press Return to store the edited line into memory.

NOTE

When a BASIC statement takes up more than one screen line (that is, you pressed Ctrl J to insert a line feed character while entering the line), you cannot use Modify mode to edit that statement. You must use Edit mode instead.

The following steps lead you through a simple exercise where you use each of these commands.

Step 1. Go to the main P.A.M. screen on your computer.

Step 2. Move the display pointer to the box labeled: `BASIC`

Step 3. Press `Start Applic` (f1) and wait for the command level prompt: `Ok`.

Step 4. To start programming, type:

```
AUTO Return
```

This command tells BASIC to automatically prompt you with the next line number after you finish each line in your program. Notice how BASIC starts you with the first line number, `10`.

Step 5. Now type the following short program:

```
10 FOR I = 1 TO 10 Return
20     PRINT I Return
30 NEXT I Return
40 PRINT "LOOP DONE, I ="; I Return
50 END Return
60
```

Step 6. Simultaneously press the Ctrl and C keys to stop the automatic line number prompt.

Step 7. Type:

```
RUN Return
```

The program prints the output from the program to your screen:

```
1
2
3
4
5
6
7
8
9
10
LOOP DONE, I=11
Ok
```

Step 8. To list your program on the screen, type:

```
LIST Return
```

BASIC shows you the listing:

```
10 FOR I=1 TO 10
20     PRINT I
30 NEXT I
40 PRINT "LOOP DONE, I="; I
50 END
Ok
```

Step 9. BASIC provides a variety of ways to modify an existing program. In this step, you will use the Edit mode subcommands to change the first line of the program so the loop counts back from 10 to 1.

- Type:

```
EDIT 10 Return
```

BASIC takes you into Edit mode on line 10.

Computer Control

Several BASIC statements let you control your computer from the program. These statements are:

DATE$	Sets the current date.
INP	Returns a byte, which is read from a microprocessor port.
OUT	Sends a byte to the microprocessor port.
POKE	Writes a byte into a memory location.
TIME$	Sets the current time.
WAIT	Suspends program execution while monitoring the status of a microprocessor input port.

To control your computer from the program, you can also use escape sequences. For example, the sequence ESC H "homes" the cursor, and the sequence ESC J clears the screen from the cursor to the end. Therefore, you could clear the entire display by executing this statement:

```
PRINT CHR$(27) + "H" + CHR$(27) + "J"
```

(CHR$(27) is the ASCII code for the escape character.)

For details on using escape sequences, refer to Appendix B. For a complete list of escape sequences that you can use with your Portable PLUS, refer to the *Portable PLUS Technical Reference Manual* (HP 45559K), which is available from your HP sales representative.

Program Control, Branching, and Subroutines

Several BASIC statements let you control the flow of your program through branching to other lines, subroutines, and programs. These statements are:

CALL	Calls an assembly-language subroutine.
CALLS	Calls a subroutine with segmented addresses.
CHAIN	Calls a program and passes variable values to it from the current program.
DEF FN	Names and defines a user-written function.
DEF SEG	Assigns the current segment address. Subsequent CALL, CALLS, POKE, PEEK, or USR instructions refer to this address.
DEF USR	Assigns the starting address of an assembly-language subroutine.
END	Ends program execution, closes all files, and returns control to the command level.
FOR...NEXT	Loops through a series of instructions a given number of times.
GOSUB...RETURN	Branches to and returns from a subroutine.
GOTO	Branches unconditionally to the specified line number.
IF	Determines program flow based on the result returned by a logical expression.
ON ERROR GOTO	Enables error trapping and specifies the first line number of the error-handling subroutine.
ON...GOSUB	Branches to a subroutine, or subroutines, depending on the value returned by the governing expression.
ON...GOTO	Branches to one of several specified line numbers, depending on the value returned by the governing expression.
RESUME	Continues the program after an error recovery procedure.

Appendix B

USING TERMINAL FEATURES IN BASIC

Introduction

You can program the terminal portion of your computer to perform many of the functions of an intelligent terminal. By using these features, you can tell the computer to perform tasks that would otherwise be done within each application program.

Most tasks that you do at the keyboard can also be done under program control with escape sequences. An escape sequence is simply a series of ASCII characters preceded by the escape character, ESC (ASCII code 27). Each escape sequence tells the computer to do a certain task. For example, the escape sequence `ESC h` "homes" the cursor to the upper left-hand corner of the screen.

This appendix shows some examples of how you might use escape sequences. For a list of all the escape sequences that you can use on your Portable PLUS, refer to the *Portable PLUS Technical Reference Manual* (HP 45559K), which is available from your HP sales representative.

NOTE

For clarity, this appendix shows a space between each character in an escape sequence. When you type in the sequence, do NOT insert spaces.

Escape sequences fall into two categories: two-character sequences and multiple-character sequences. For two-character sequences, you must press the keys in order and use the correct case (upper- or lower-case). For example, `ESC B` (Esc then the shifted B key) moves the cursor down one row whereas `ESC b` (ESC then the unshifted B key) unlocks the keyboard. The difference appears subtle but is quite important to your computer.

Multiple-character escape sequences have one or more groups of characters. Each group, consisting of a number or other character followed by a letter, specifies one parameter of the sequence. Generally, you can arrange these groups in any order or even leave some out entirely, depending on the task you want your computer to do. In this type of escape sequence, a capital letter defines the end of the escape sequence, so you would capitalize only the last letter and type the rest in lower case. An example is the cursor-positioning escape sequence. Here is an escape sequence that positions the cursor at row zero, column zero ("home"):

```
ESC & a 0 r 0 C
```

The uppercase `C` ends the sequence. But since you can interchange the order of groups in a multiple-character escape sequence, you could also use:

```
ESC & a 0 c 0 R
```

This escape sequence does the same task as the other one. Only the order of groups of characters is different. Again, the capital letter ended the sequence. The order of the groups of characters is not critical as long as the last character is an uppercase letter.