



**HEWLETT-PACKARD 9830A CALCULATOR**  
**SIMPLIFIED OPERATING INSTRUCTIONS**

## **A WORD ABOUT ...**

In this book, we discuss many topics in relatively few pages. Our purpose is to enable you to:

- Do a wide variety of calculations in just a few minutes;
- Begin using the editing capabilities of the calculator;
- Use your tape cassette advantageously;
- Begin programming.

It is our intent to 'get you on board' fast! For this reason, some of the more difficult concepts are avoided entirely while others are greatly simplified. But depending on your particular needs, you may have to go no further than this book.

However, if it's just facts you want, go to the Operating and Programming Manual. To become really proficient using the Model 30, you will need the information in that book (especially the tape cassette and programming information).

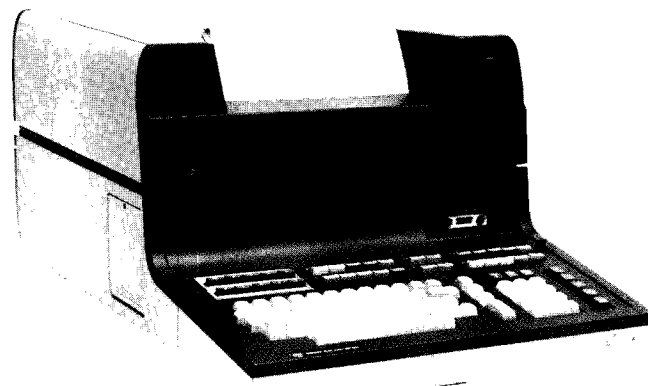
With the information presented in this book, you will have enough of a background to try things on your own, which is actually the best way to attain a good working knowledge of the calculator (and don't worry, you can't damage the calculator with any keyboard operation)!

**HP Computer Museum**  
**[www.hpmuseum.net](http://www.hpmuseum.net)**

**For research and education purposes only.**

# SIMPLIFIED OPERATING INSTRUCTIONS

## MODEL 9830A CALCULATOR



9830A CALCULATOR SHOWN WITH 9866A PRINTER

HEWLETT-PACKARD CALCULATOR PRODUCTS DIVISION  
P.O. Box 301, Loveland, Colorado 80537, Tel. (303) 667-5000  
(For World-wide Sales and Service Offices see rear of manual.)

Copyright Hewlett-Packard Company 1972

# TABLE OF CONTENTS

## CHAPTER 1: MEETING THE CALCULATOR

	1
Introduction	1
Scanning the Keyboard	2
Clearing the Display	5
Evaluating Mistakes	6
Keyboard Calculations	6
Arithmetic Operations	7
Arithmetic Hierarchy	11
Variables	12
Editing	14
Odds and Ends	17
Functions	26
Programming Preview	28
Summary	30

## CHAPTER 2: TAPE CASSETTES AT A GLANCE

	33
Preparing your Cassette	34
Mark	34
Store	37
Load	38
Summary	39

## CHAPTER 3: PROGRAMMING

An Anecdote	41
Time Savers	42
A Short Example (fold-out)	47
Editing Hints	49
Usable Memory	54
A Shorter Example (fold-out)	57
Determining Memory	57
Additional Statements	61
Loose Ends	63
Summary	64
9830A Keyboard (fold-out)	67

## **CAUTION**

The Model 30 can be severely damaged if it has not been set to the correct voltage; if in doubt, please refer to Appendix A of the Operating and Programming Manual.

# **Chapter 1**

## **MEETING THE CALCULATOR**

### **INTRODUCTION**

Congratulations! You now have at your disposal an extremely powerful device — the HP 9830A Calculator; and it's easy to use! But before you start playing with it, you should make a couple of quick checks.

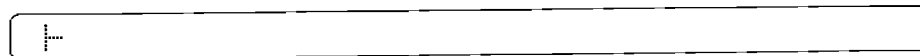
If you have just received your calculator, please be sure that it's immediately inspected (refer to Appendix A of the Operating and Programming Manual for the correct procedure). But if the calculator has already been running in your area, just do the following:

1. If the calculator is not plugged in: Plug one end of the power cord into the lowest socket on the rear panel of the calculator; plug the other end of the cord into a suitable power outlet, such as a wall socket.
2. If the calculator is switched off: Set the OFF/ON switch, which is located on the front of the calculator, to the ON position (see Figure 1).



3. If the calculator is already on: Check with other users to be certain that nothing important is in the calculator 'memory' (there is a certain amount of space allocated to the calculator user, which can be used for programming, etc. — and we call this the memory). After this check, turn the calculator off, and a few seconds later turn it on again. (This procedure erases everything that was previously in the memory area.)

When the following display appears, the calculator is ready to operate:



So much for necessities.

## SCANNING THE KEYBOARD

Let's take a quick look at the Model 30 keyboard. (If you don't have the calculator there with you, you can refer to the fold-out of the calculator keyboard provided in the back of this book.)

The first thing you probably notice is the variety of colors on the keyboard. Keys with the same color coding have some similar characteristics. As you become more familiar with the capabilities of the Model 30, these similarities will become fairly obvious. There's no real benefit in mentioning them here.

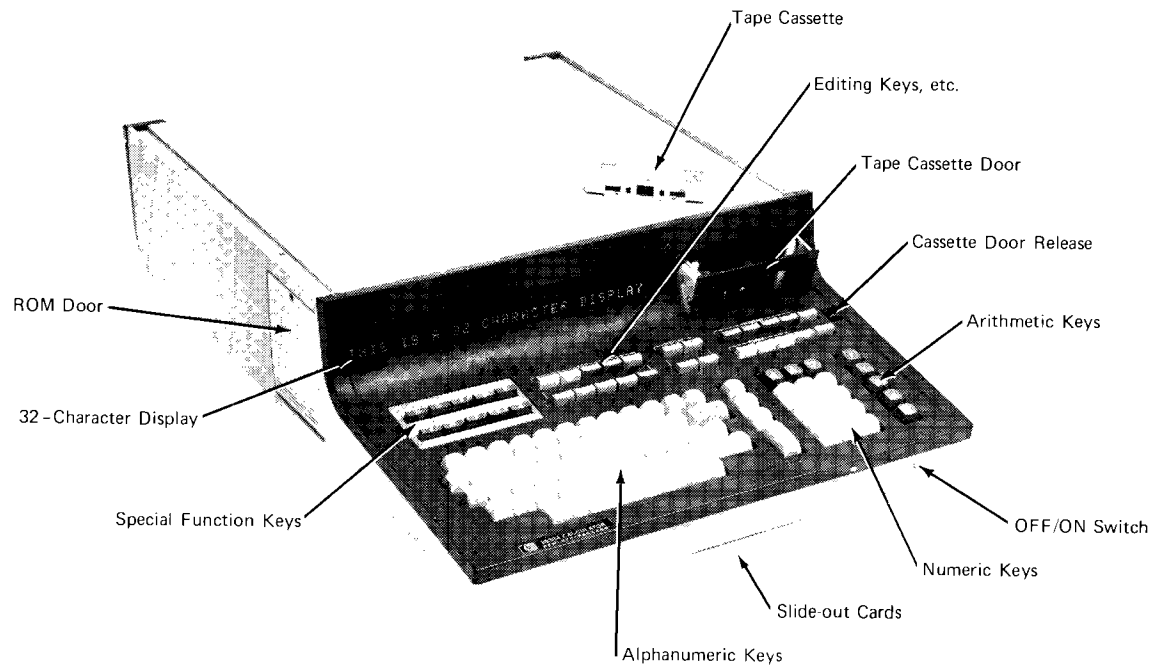


Figure 1. The Model 30 Calculator

Then you see that the keyboard is divided into specified areas (see Figure 1). They are briefly described below:

- **Alphanumeric Keys** — This area acts very much like a standard typewriter keyboard. If, for instance, you want to display the dollar sign, \$, you must have the SHIFT key held down when you press  $\text{\$}$ . It differs from a standard typewriter in that letters always appear in upper case on the display.
- **Numeric Keys** — This area conveniently locates a set of numbers to the left of the five arithmetic keys. If, however, you feel more comfortable using the other set of numbers (in the alphanumeric region), go right ahead. It makes no difference to the calculator.
- **Special Function Keys** — The keys in the upper left-hand region of the keyboard  $f_0$  through  $f_9$ , add considerable flexibility to the calculator. However, since this flexibility is not needed initially, we won't discuss the keys in this book. The Operating and Programming Manual discusses them in detail.
- The rest of the keys in the upper half of the keyboard are helpful in a variety of ways. Some are especially useful as editing tools, while others have more specific uses. The keys are described, in appropriate places, throughout this book.

Let's discuss two topics briefly — Clearing the Display and Evaluating Mistakes. Then we'll be ready to do some calculations.

## CLEARING THE DISPLAY

There are several situations when information that is on the display does not have to be cleared before new information is keyed in; for instance, after either an error message or the result of a calculation is displayed. In both cases, the previous display is automatically cleared when additional characters are keyed in.

But to clear the display of a line you are currently keying in, you should press:



When this key is pressed, the symbol,  $\text{⌂}$ , appears on the display; a new entry can then be made.

There may come a time when you inadvertently press some combination of keys which 'locks out' the keyboard from further inputting. In this case, keeping the STOP key held down until STOP appears on the display will generally return control of the keyboard to you. If this does not help, as a last resort you can turn the calculator off and then on again. You can then input as before; unfortunately, everything previously in memory is erased if you have to turn the calculator off.



## EVALUATING MISTAKES

Although the Model 30 is easy to use, mistakes still occur. Fortunately, the calculator lets us know when we make one by making a soft beeping sound and by referencing an error message; e.g.,

A rectangular display box with a dashed border containing the text "ERROR 12".

ERROR 12


A list of the error messages is located both on a slide-out card on the bottom of the calculator and in the Operating and Programming Manual. This list should help you determine the cause of any error.

## KEYBOARD CALCULATIONS

Arithmetic calculations are easily performed on the Model 30. You key in your problem and press the EXECUTE key; the result appears on the display.

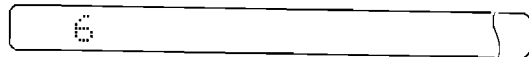
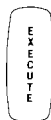
The appearance of the result can vary, however, as discussed beginning on page 22; for now, merely knowing that it can vary is sufficient.

An arithmetic expression is entered into the display by pressing keys in the same order as they would be written on paper, one key per character or symbol; for example,

A row of three circular buttons labeled "2", "+", and "4" followed by a rectangular display box with a dashed border containing the text "2+4".

2 + 4

To get the result, press:



By the way, if you need to repeat a key several times, hold the key down for about two seconds and it will rapidly repeat itself.

## ARITHMETIC OPERATIONS

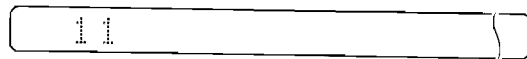
In the following examples, we show commonly used arithmetic operations.

### NOTE

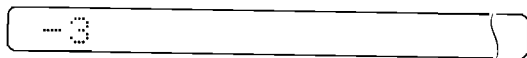
After each expression is keyed in, the EXECUTE key should be pressed to obtain the desired results. If you make a mistake, press CLEAR, then key in the expression again.



Addition:



1. Subtraction:



—

2. Using negative numbers:

( — ) ( 7 )

—7

( 4 ) ( + ) ( ( ) ( — ) ( 7 ) ( ) )

—3

or

( 4 ) ( + ) ( — ) ( 7 )

—3

\*

Multiplication:

( 4 ) ( \* ) ( 7 )

28

( 4 ) ( · ) ( 3 ) ( \* ) ( ( ) ( 3 ) ( — ) ( 8 ) ( ) )

—21.5

/

Division:

( 4 ) ( / ) ( 7 )

0.571428571

( 2 ) ( \* ) ( 3 ) ( / ) ( ( ) ( 3 ) ( — ) ( 1 ) ( 5 ) ( ) )

—0.5



Exponentiation (raising to a power):

3

↑

2

9

(

4

+

1

)

↑

3

125

but

4

+

1

↑

3

5

S

Q

R

Square Root:

S

Q

R

2

5

5

S

Q

R

(

4

\*

7

)

5.291502622

S

Q

R

4

\*

7

14



## ( ) Parentheses:

As shown in the previous examples, quantities within parentheses are treated as one quantity. Thus  $\text{SQR}(4 \times 7)$  is equivalent to  $\sqrt{28}$  whereas  $\text{SQR } 4 \times 7$  is equivalent to  $\sqrt{4}$  multiplied by 7.

Implied multiplication, e.g.,  $4(3+2)$ , is not allowed; enter  $4 * (3+2)$  instead.

Parentheses can be nested; that is, parentheses within parentheses are allowed. Be certain that your expression contains the same number of left parentheses as right parentheses. For instance, press:

S Q R ( 3 ↑ ( 2 \* ( 5 - 3 ) ) )

The display is:

SQR(3↑(2\*(5-3)))

To get the result, press:

EXECUTE

9

## THE ARITHMETIC HIERARCHY

The arithmetic hierarchy, presented below, gives the order of execution in the calculator. It's the same as the order commonly used in standard arithmetic.

- |   |                    |
|---|--------------------|
| 1. Mathematical Functions (e.g., square root) | highest precedence |
| 2. Exponentiation                             |                    |
| 3. Multiplication, Division                   |                    |
| 4. Addition, Subtraction                      | lowest precedence  |

The order of execution is from highest precedence to lowest precedence.

When an expression contains two or more operations at the same level in the hierarchy, the order of execution is from left to right.

The use of parentheses enables the order of execution to be changed; e.g., in the SQR ( $4*7$ ), the '\*' is executed before the 'SQR' even though multiplication occupies a lower level in the hierarchy. Consequently, adding parentheses raises the hierarchy of the enclosed operations.

If you are uncertain of the order of execution in a particular expression, you can add parentheses for the sake of clarity, whether or not they are required.

## VARIABLES

A variable is an alphanumeric representation of a number. Several uses of variables will be shown throughout this book. There are two types of variables — ‘simple’ variables and ‘array’ variables. In this book we are concerned only with simple variables.

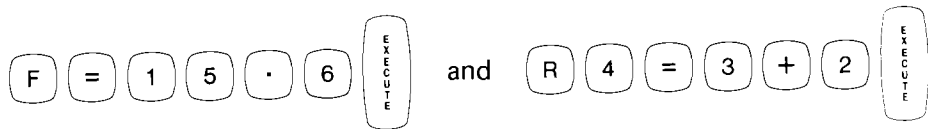
A simple variable can be:

- A letter (from A through Z), or
- A letter followed by a digit (from 0 through 9).†

Some examples of simple variables are: B0, C, F, F9, Q, Z6.

A variable does not exist in the calculator until it is given a value. The following two examples assign values to the simple variables F and R4.

Let's press:



Notice that only the variable is allowed to the left of the equals sign.

† Sometimes we slash zeros, 0, where they might be confused with the letter, O.

Now that these variables have values, they can be used in arithmetic expressions. For example, we can 'execute' the following expressions:

F		15.6
R	4	5
F	/ 3 + R 4	10.2
S	Q R ( R 4 * F )	8.831760066
R	4 = R 4 + 1	6
R	4 * 3	18

The variable, R4, was initially assigned the value of 5. It retained that value until the expression,  $R4 = R4 + 1$ , was executed. This technique of assigning new values to variables is commonly used in programming, as you will see later.

The other variables, array variables, although quite useful in programming, demand a greater explanation than is warranted in this book; hence, they are discussed in the Operating and Programming Manual.

## EDITING

Now let's see how to edit. Supposing we wish to evaluate:

$$\frac{1+2+3+4+5+6+7+8}{9}$$

On a bad day, we might begin by keying in something like:

1+2+3+4+5+6+7\*8

**BACK**

By now we have realized that we pressed 7\*8 instead of 7+8. We can easily correct the display if we begin by pressing **BACK** twice; this positions us at the '\*' as confirmed by a blinking indicator there. Wherever this indicator is located, editing can always be performed; so we can immediately press '+' thus replacing the '\*'.

The display will now be:

1+2+3+4+5+6+7+8

Now the indicator is located at the 8.

**FORWARD**

To complete the expression to include the division by 9, we can first press **FORWARD**. When we press

this key, the blinking indicator shifts one character to the right; however, since (in this case) there are no characters to the right of the 8, the blinking disappears. We can now key in the division by 9; then the display will be:

1+2+3+4+5+6+7+8/9

Unfortunately, this is not the correct expression since the order of execution in the arithmetic hierarchy was not taken into account. As it now stands, 8 will be divided by 9 and the result will be added to the summation of 1 through 7. To account for the hierarchy and obtain the desired results, we must group the numbers 1 through 8 within parentheses.


#### INSERT

We need a right parenthesis ')' immediately after the 8 and a left parenthesis '(' in front of the 1. This can be accomplished with the help of the INSERT key. If initially we press **BACK** twice, the indicator is positioned at the '/'. A space can now be opened up to the left of the blinking indicator if we press **INSERT**. The indicator then repositions itself over the blank space. So now we can key in the right parenthesis.

1+2+3+4+5+6+7+8) /9



We can then use the same procedure to key in the left parenthesis. But instead of having to press **BACK** 16 times to position the indicator at the 1, we can just keep the BACK key held down; after about two seconds, the key will repeat its operation in rapid succession. By releasing the


key, we can stop the backward movement at its current location. Once the blinking indicator is located at the 1, we can press  which will leave a space to the left of the 1. Now by keying in the left parenthesis, we have the desired expression:

(1+2+3+4+5+6+7+8)/9

After all that we might as well execute the expression. The result is 4 .

There is one other important application of the INSERT key. Suppose we have the following display:

19673

Now supposing we wish to delete the 6, we would first press the BACK key until the indicator is positioned at the 6. To delete the 6 and close the spacing between the 9 and 7, we can hold down the SHIFT key and press  . The display will then be:

1973


The 6 could also have been deleted by pressing the space bar when the indicator was appropriately positioned; however, this method would have left a blank space between the 9 and 7.

## ODDS AND ENDS

Here are a couple of additional features that will come in handy.

### RECALL

The RECALL key is extremely handy! When pressed, it returns to the display the last thing that was executed. It is most useful in two particular situations — when you execute an expression and either an error message or a totally unexpected answer appears. In both situations, being able to take another look at the original expression can be quite helpful.

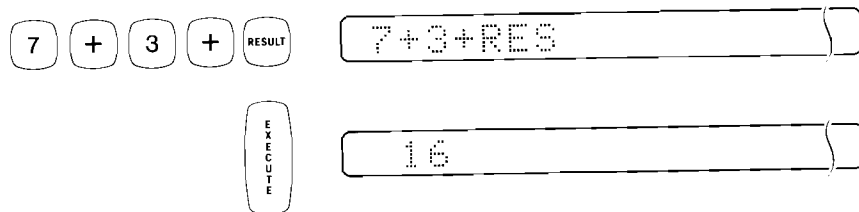
Just press  and the original expression is returned to the display. It can then be changed and executed if editing is required.

### RESULT

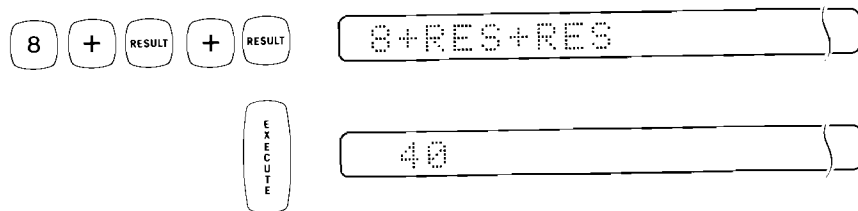
Another useful tool is the RESULT key. Pressing the keys — CLEAR RESULT EXECUTE — displays the numerical value of the last arithmetic statement that was executed. More importantly, the RESULT key can function as an 'accumulator' during arithmetic operations.



For example, if we execute  $2+4$ , 6 is displayed. Then if we press:





The previous result of 6 was added to  $7+3$ . Now if we key in:



The previous result (this time, 16) was added twice to the number 8.



When over 32 characters are being keyed in (80 characters maximum), the characters to the left of the display are pushed out of the display region to make room for the additional characters.


To view the beginning of the input, press  (the right arrow); this operation moves the characters in the display to the right. Pressing  (the left arrow) performs the reverse operation. Either arrow key, when held down for about two seconds, repeats its operation in rapid succession.

For example, let's add the numbers 1 through 20 inclusive; after the first 32 keystrokes, the display is:

1+2+3+4+5+6+7+8+9+10+11+12+13+14

Now let's input the rest of the expression; notice that after the 32<sup>nd</sup> keystroke, each additional keystroke causes the display to be shifted by one to the left, until finally the display is:

10+11+12+13+14+15+16+17+18+19+20

Now we can review the beginning of the line by holding down  for a few seconds. Whenever we are confident that the expression was keyed in correctly, we can execute it.

210

By the way, if you are keying in an extremely long line, the calculator makes a soft beeping sound when 72 of the allowable 80 characters are input.

## SIMULTANEOUS CALCULATIONS

More than one expression can be executed at the same time. If you'd like to see the results of  $370.37 * 6$  and  $370.37 * 9$  together on the display, just execute the following:

(3) (7) (0) (.) (3) (7) (\*) (6) (,) (3) (7) (0) (.) (3) (7) (\*) (9)

The results will be:

2222.22

3333.33

By separating the two expressions by a comma, they are both executed. Because of this feature the use of commas within a number (such as in 19,642.73) is not allowed.

### **PRT ALL**

The PRT ALL key is used to obtain a printed record of your operations.

With the print-all mode in effect:

- If you execute an expression, both the expression and the result will be printed.
- All error messages will be printed.
- Program lines being input to memory will be printed.

Pressing **PRT ALL** either establishes or cancels the print-all mode:

- If ON appears in the display, the print-all mode is established.
- If OFF appears, the print-all mode is cancelled.

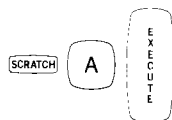
By pressing **PRT ALL** a second time, the ON or OFF designation is reversed.

To obtain printed records, you will have to be sure that your printer is set up properly. If it has been operating previously, just be sure it's turned on. If it's turned on and you cannot get a printed record, please refer to Appendix A of the Operating and Programming Manual for printer plug-in instructions.

If for some reason the printer isn't operating, be sure the print-all mode is OFF; otherwise, the display will blank out when you press the EXECUTE key (if this occurs, hold the STOP key down until STOP appears on the display to regain control of the calculator).

#### **SCRATCH**

There will be occasions when you want to erase all the information being stored in memory. (If you do programming, this will be especially true.) To erase memory, just press:



'Scratch All' erases everything from the user's memory area. Turning the calculator off, then on again, performs the identical function.

Other variations of the SCRATCH command are described in the Operating and Programming

Manual. It should be noted, however, that SCRATCH is a very powerful command, and as such, it should be used cautiously.

STD   FIXED N   FLOAT N

We mentioned earlier that the results of calculations can appear in different forms; the three possibilities are standard, fixed-point, and floating-point.

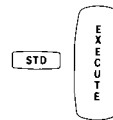
When the Model 30 is first turned on, the results of calculations appear in standard form. This form is so flexible that, in general, you aren't even aware of it. The results of calculations appear just as you would expect. For this reason, you will generally not even be concerned about changing this form.

If, however, you wish to have a specified number of digits to the right of the decimal point, you might prefer to use fixed-point form where the number of digits to the right of the decimal point is determined by the particular fixed-point form. For instance, in fixed-3 form, the number 123.45 would appear as 123.450 .

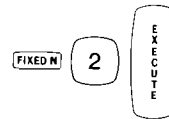
The third form is floating-point (often called scientific notation). If your applications are not in the scientific field, you may rarely use it. But occasionally, for very large and small numbers, the calculator reverts to floating-point regardless of the specified form; for this reason, you should at least be aware of it.

Any of these three forms can be specified:

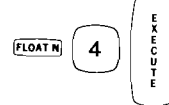
- Standard form can be obtained by pressing:



- Fixed-point form can be set up to have as many as 11 digits following the decimal point; if, for instance, we want our calculations to have two digits following the decimal point, we can press:



- Floating-point form also can have up to 11 digits following the decimal point. If, for instance, we want four digits following the decimal point, we can press:



Examples a through f, below, show how the given numbers would appear in each form — and for convenience, we will assume fixed '2' and float '4' as previously illustrated:

		Standard	Fixed '2'	Float '4'
a)	23	23	23.00	2.3000E+01
b)	-173.0	-173	-173.00	-1.7300E+02
c)	.068	0.068	0.07	6.8000E-02
d)	12345.6	12345.6	12345.60	1.2346E+04
e)	.003	3.00000E-03	0.00	3.0000E-03
f)	7.314	7.314	7.31	7.3140E+00

As you probably noticed, floating-point form does nothing more than relocate the decimal point. If the exponent, E, is followed by a minus sign (as in examples c and e), the decimal point has been shifted to the right by the number of places indicated by the two digits following the 'E'. If the sign is plus, the decimal point has been shifted to the left.

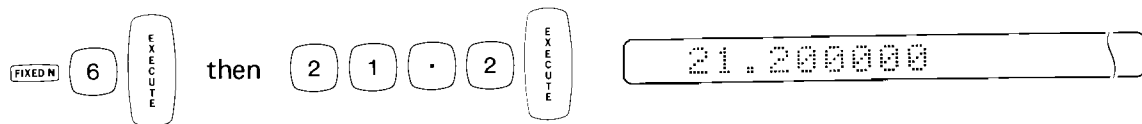
Notice too, that in example e, the standard form reverts to floating-point.

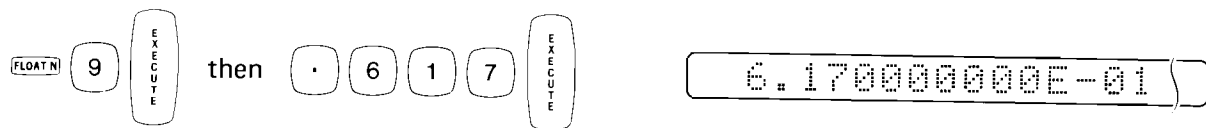
Occasionally, results are rounded depending on the specified form; for instance, in the fixed '2' form, where only two digits are allowed to the right of the decimal point:

- In example c, .068 was rounded 'up' to .07, whereas
- In example e, .003 was rounded 'off' to .00.

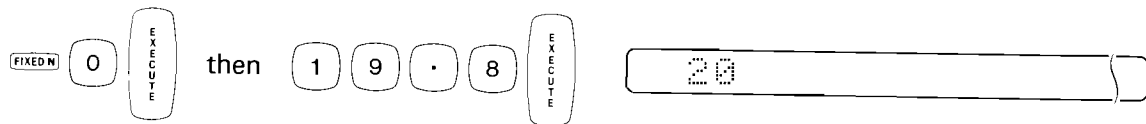
If the first excess digit is 5 or greater (as in example c), the last allowable digit is rounded 'up'. If the first excess digit is less than 5 (as in example e), the last allowable digit remains the same.

Here are some examples using fixed and floating-point form:

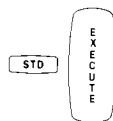




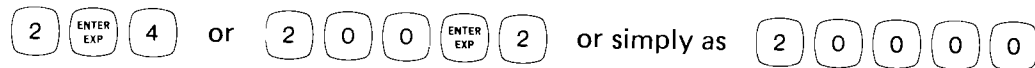
One other thing. In both fixed and float '0', the decimal point is suppressed; for example:



Then, of course, to revert to standard form, press:



We can key in numbers in floating-point form by using the ENTER EXP (Enter Exponent) key; for example, the number  $2 \times 10^4$  can be keyed in many different ways; for instance, as:





## ACCURACY

The Model 30 calculates to 12-digit accuracy. But for you to view the results of all calculations to the full 12-digit accuracy, it is necessary to perform calculations in float '11' form; for example, in fixed '4' form, 3.14159265360 would be rounded to 3.1416 on the display, even though the calculator itself retains the result as 3.14159265360 .

If you plan to use trigonometric functions, you will find the following information valuable. Otherwise, go to the Programming Preview on page 28.

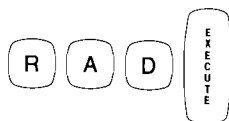
## FUNCTIONS

The trigonometric functions and their corresponding mnemonics are given below:

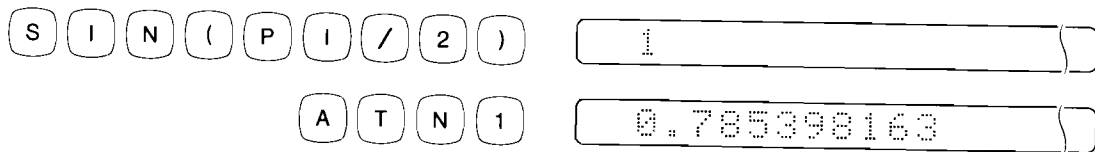
Sine	SIN	Tangent	TAN
Cosine	COS	Arctangent	ATN

The angle is always assumed to be in radians (RAD) unless otherwise stated. To express an angle in either degrees or grads, clear the display, then key in and execute either D E G or G R A D, respectively; then input the desired expressions.

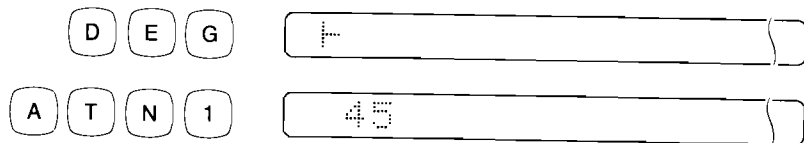
To revert to radians, clear the display, then press:



In addition, the value of  $\pi$  can be expressed by the keys P I. For example, with the calculator in radians, let's execute the following expressions:



Then reverting to degrees by executing:



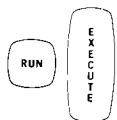
## PROGRAMMING PREVIEW

Even if you plan only to run existing programs, you should find the information in this section useful.

Here's a short program for you to key in. Just press the indicated keys, row by row. If you press an incorrect key, you can use the editing keys discussed earlier to help make the corrections; or else, press the CLEAR key, and re-enter the line. Notice that each new line is preceded by a unique line number and that each line is entered into memory when the END OF LINE key is pressed.

1	A	=	1	END OF LINE		
2	D	I	S	P	A	END OF LINE
3	A	=	A	+	1	END OF LINE
4	G	O	T	O	2	END OF LINE
5	E	N	D	END OF LINE		

The program can be run by pressing:



The number on the display just keeps increasing by 1.

Whenever you want to stop the program, press:



If the program didn't run for you, you can correct any error by keying the appropriate line over again.

The two key sequences just presented (RUN EXECUTE and STOP) are the general purpose commands for running and stopping programs. But, since each program has its own specific requirements, it will be necessary for you to obtain special instructions for each program that's unfamiliar to you.

We'll be using this program again in Chapter 2.



## SUMMARY

Certain items have not been discussed in great detail; others have been avoided entirely; pitfalls have not been emphasized. The best way to learn what you can and cannot do on the Model 30 is — try it; if you get an error message, press **RECALL**; you should be able to determine the error by comparing your expression with the error-message explanation.

If you feel comfortable using the Model 30 for arithmetic operations and you would now like to learn how to do a few handy things with the tape cassette, you should read Chapter 2.

Chapter 3 is a quick and easy introduction to HP BASIC programming. If you are already familiar with the BASIC language, the Operating and Programming Manual will be more useful to you since it points out, in greater detail, additions to BASIC as well as changes.

Besides the trigonometric functions, other mathematical functions are available on the Model 30. If you use these functions, the brief explanations offered below should be sufficient; in all cases, the mnemonic, followed by an appropriate expression in parentheses will give the indicated results:

### Available Mathematical Functions

ABS (expression)	determines the absolute value of the expression;
EXP (expression)	raises the constant, $e$ , to the power of the computed expression;

INT (expression)	gives the expression an integer value $\leq$ the value of the expression;
LGT (expression)	determines the logarithm of a positively valued expression to the base 10;
LOG (expression)	determines the logarithm of a positively valued expression to the base 'e';
RND (expression)	gives a random number between 1 and 0; the expression is a dummy argument;
SGN (expression)	returns a 1 if the expression is greater than zero, returns a 0 if the expression equals zero, or returns a -1 if the expression is less than zero;
SQR (expression)	computes the square root of a positively valued expression.

## NOTES

## **Chapter 2**

# **TAPE CASSETTES AT A GLANCE**

Tape cassettes offer a convenient method of retaining information. It's easy to put information on cassettes and once there, the information is readily accessible.

If you plan to do your own programming, then with the material presented in this chapter and the next, you should be able to write some programs and record them on a cassette. If possible, it will be to your advantage to obtain a cassette for your use only; then you won't have to worry about other people accidentally wiping out valuable information.

The Operating and Programming Manual explains many additional tape cassette and programming capabilities, such as ways to increase the effective memory of the calculator, that are not discussed in this book.

In this brief chapter, we shall consider only the most basic uses of the cassette. If you plan to use cassettes exclusively for running existing programs, then the material presented here should suffice.



## PREPARING YOUR CASSETTE

To open the cassette door, flick the switch on the upper right-hand corner of the calculator keyboard. Slide in the cassette through the guide-posts on the cassette door; be sure the cassette is right-side up with the FRONT label facing you. Push the door closed and press the key **REWIND**. Now you're ready to mark some files.

But just what are cassette files? Well, they are the physical means for separating one group of information (such as a program) from another. The first file on a tape is FILE 0; subsequent files are identified as FILE 1, FILE 2, etc. You can designate both the number of files and the lengths of the files you want by using the MARK command (discussed next).

But if you plan only to run existing programs, the only command you will be concerned with is LOAD, discussed on page 38.

## MARK

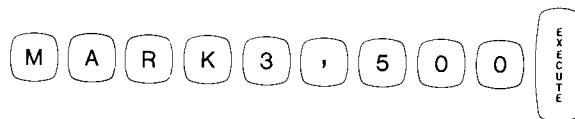
Before you mark any files, you should make a couple of quick checks:

- First, check with other users to make sure nothing important is on the cassette.
- Then open the cassette door. If the left tab on top of the cassette has been removed, the cassette is protected and you can't mark on it (see Figure 2). In this case, you'll need another cassette. Protected cassettes are discussed in the Operating and Programming Manual.

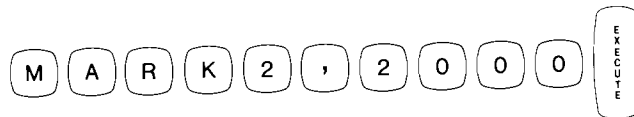


Figure 2. Protected Cassette

Assuming that your cassette is available for use, close the cassette door. You can now mark some files. Let's begin by marking three files with 500 word lengths (relatively small) and two files with 2000 word lengths (for considerably larger programs). Clear the display, then press:



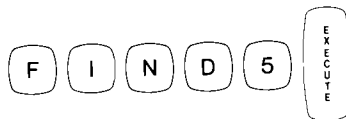
After these three files are marked (it takes about 30 seconds), the cassette tape halts, and 'H' appears on the display. Then press:



After about 80 seconds, these two files also are marked. So five files have been marked in all:

- FILES 0, 1, and 2 have 500 word lengths;
- FILES 3 and 4 have 2000 word lengths.

If at some time in the future you would like to mark some more files, you will first have to position the tape at the end of the previously marked files. Since FILE 4 was the last one marked, you can position the tape at the beginning of the next file, FILE 5, by pressing:



Additional files can then be marked starting at FILE 5.

Now rewind the tape by pressing: **REWIND** .

It's not always necessary to rewind the tape, but it is a good habit to get into, especially before taking a cassette out of the calculator. (The usable portion of the tape stays cleaner that way.)

Once the cassette is marked, programs can be recorded on individual files by using the STORE command.

## **STORE**

When you have a program in the calculator, you can immediately store it on a cassette:

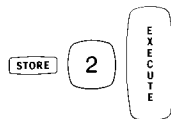
- If you have an available cassette file that is large enough to handle the size of your program, and
- If your cassette is not protected (as mentioned earlier).

To be available, a file does not have to be empty; if the information in a file is no longer of use, that file is available.

In Chapter 3 we show you how to determine the size of your program. If you try to store a program into a file that's not as large as the program size, an error message will be displayed. Then just store it into a larger file.

Now let's see just how to go about storing a program. If you keyed in the sample program in Chapter 1, page 28, and if you haven't inadvertently erased the calculator memory since then, you can store that program. (Try to run the program; if it doesn't run, you can key it over again — if it runs, you'll have to press the STOP key before you can store it.)

This program is very short (about 21 words) so it can be stored in any of the five files we marked. Let's store it in FILE 2, which can accept programs up to 500 words in length. Press:



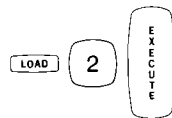
In about 10 seconds, all the program lines currently in memory will be recorded on FILE 2 (and the program is still in the calculator memory, too).

This operation can be reversed by using the LOAD command.

## LOAD

With this command, the program lines in a particular cassette file can be reproduced into the calculator memory. Any program lines previously in memory will be erased.

If the program that's being stored on FILE 2 of your cassette is ever erased from memory and you then decide to run that program, just be sure the correct cassette is in the calculator, and press:



In about 10 seconds, the program in FILE 2 will be loaded into memory; and the contents of FILE 2 remain intact.

You can now run the program as before.

## SUMMARY

Most of the tape cassette commands have not even been mentioned here; and the commands that were mentioned can also be greatly expanded. For example:

- Program lines can be selectively stored on the cassette.
- Program lines in a file can be merged into memory between program lines currently there.
- Programs can be stacked in memory.
- Data, as well as programs, can be stored on the cassette.
- A listing of the information on a cassette, file-by-file, can be obtained.
- Many other useful techniques are also available.

If there is something you would like to do using the cassette, but you don't know how to go about it, chances are that it's possible; if so, the explanation can be found in the Operating and Programming Manual.

However, even with only a brief explanation of the MARK, STORE, and LOAD commands given here, you should initially be able to use these commands without the need of further explanation. In fact, depending on your program requirements, you may never have the need to reference the 9830A Operating and Programming Manual.

## NOTES

## Chapter 3

# PROGRAMMING

Intimidated? If you have never programmed, you may possibly be intimidated by the title of this chapter. Don't be! 'Programming' doesn't have to be a scare word. Perhaps in the past. Not now. Several programming languages have been developed that are really easy to use. One of these languages, called BASIC, is used on the Model 30.

BASIC is just a simplified version of English. A BASIC program is a set of directions organized to accomplish certain tasks. The program is made up of statements, where each statement is generally self-evident and contains a minimum of punctuation.

Each statement must be preceded by a line number. Line numbers must appear in the program in ascending order for 'bookkeeping' purposes. However, you can type your program lines in any order since the calculator sorts the lines into numerical order as they are entered.

After a program line is written (80 characters maximum), it is entered into the calculator memory by pressing:



END OF LINE

This key is used only to enter program lines into memory. Don't confuse it with the EXECUTE key.



The following example is meant to show you the general techniques involved in writing a program. It does not teach you individual program statements. The rest of the chapter is devoted to that task.

## AN ANECDOTE

*According to unreliable sources, many years ago there was a prosperous kingdom where a tired and grumpy king ruled. One day, looking for new amusement, the king sent out the following message throughout his kingdom: "Whosoever finds a game of suitable amusement for me, shall be granted any wish he desireth."*

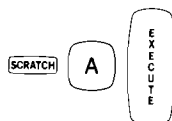
*Lo and behold, a young gentleman presented the king with the game of chess. The king was ecstatic! "What is your wish?" asked the delighted king. Replied the gentleman, "O wise and noble king, all I ask is that you put down one stalk of wheat for the first square on the chessboard, merely double this amount for the second square, then double the new amount for the third square, and so on for the remaining squares. All I wish to be given is the amount of wheat put down for the final chess square."*

*To this the king replied, "But, generous gentleman, this is a prosperous kingdom; surely I can do more for you than that!" But the gentleman was equally insistent.*

*P.S. The kingdom produced about one billion (1,000,000,000) stalks of wheat (W) annually and the chessboard had 64 squares (S) to be filled.*

*Was the gentleman really being generous?*

Before keying in the following short program to find out, let's erase memory by pressing:



Now key in the eight lines — don't forget to press the END OF LINE key at the end of each line to put it into memory. (The program steps are discussed later; don't worry about them for now.)

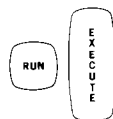
```
10 W=S=1
20 DISP W,S
30 IF W>1000000000 THEN 80
40 IF S=64 THEN 80
50 W=W*2
60 S=S+1
70 GOTO 20
80 END
```

When we execute the program, the display will flash, for each square in succession, both the number of wheat stalks and the number of the associated square.

We have the program set up to halt in either of two cases:

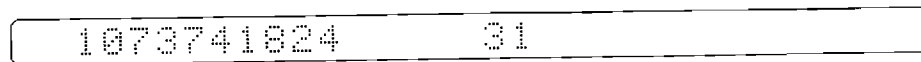
- If more than one billion stalks of wheat have to be supplied for a given square; or
- If the 64 chess squares can be filled up without depleting the kingdom's supply.

Let's execute the program by pressing:



This is the normal sequence followed when running a program.

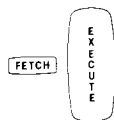
When this program halts, the final display is:



Over a billion stalks of wheat have to be placed on the 31st square!

Needless to say, the generous gentleman was executed!



Did your program run okay? If not, you can compare it line by line with ours. Press:



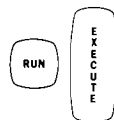
Then the first line of your program appears on the display.

If this line doesn't compare with our line 10, it can be corrected by using the editing keys discussed in Chapter 1. (By the way, don't worry about the spacing between characters; as soon as a line is input to memory, the calculator automatically puts the appropriate spacing between characters.) If you edit a line, be sure to press the END OF LINE key after the appropriate

corrections have been made; otherwise, the uncorrected version of the line will remain in memory.

When this line looks like ours, the other lines of the program can then be checked and edited if necessary. Just press  and the next higher-numbered line appears on the display. (Pressing  displays the next lower-numbered line.) All the lines can be checked this way.

When your program is corrected, you can execute it by pressing:



Let's look at the program steps briefly. If after reading the following explanation, you feel we've rushed through the example, don't worry; we'll discuss particular statements more thoroughly later.

We use only two variables in this program: W (wheat stalks) and S (squares on the chessboard).

The program 'loops' (repeats part of itself) several times.

### LOOP 1

First, the two variables, W and S, are set equal to 1 (line 10). These values are then displayed on the calculator (line 20).

Then the program checks to see:

- If either the number of wheat stalks on any one square is greater than 1 billion (line 30), or
- If the 64th chess square has been filled with wheat (line 40).

When either of these conditions is satisfied, the program will branch to line 80 and halt. But since neither condition is initially satisfied, the program continues with line 50.

In this line the amount of wheat is doubled; hence the new amount of wheat equals the old amount multiplied by 2. In line 60, the number of the chess square is increased by 1.

Then the statement in line 70 causes the program to loop back to line 20.

## **LOOP 2**

The updated values of W and S are displayed ( $W = 2$  and  $S = 2$ ). Neither one of the two IF conditions is true so the program again increases the value of W and S;  $W = 2 * 2$  and  $S = 2 + 1$ .

## **LOOP 3**

Loop 3 is performed, and so on. The program continues executing in the manner described above until one of the IF conditions (line 30 or 40) is satisfied. When this happens, the program branches to line 80 and ends. The final values for W and S remain on the display.

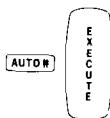
As we already mentioned, particular types of statements will be discussed in a few pages. For now, let's talk about some of the techniques you can use while keying in a program.

## TIME SAVERS

The following keys can simplify program entry. A few of them have already been mentioned briefly.

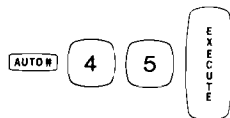
### AUTO #

We mentioned that each program statement must be preceded by a line number. If you are planning on keying in the lines in ascending order, you can have the line numbers automatically displayed for you at the beginning of each new line. For example, by pressing:



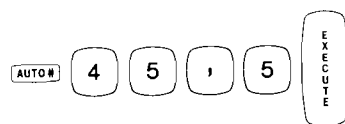
The first line number, 10, immediately appears on the display. After this line is keyed in, successive line numbers will be: 20, 30, 40 ... The spacing of 10 between consecutive line numbers allows you to insert additional lines later.

If you wish to specify a beginning line number of, say, 45 rather than 10, press:



The first line number, 45 immediately appears on the display. Successive line numbers will be: 55, 65, 75 ... The spacing of 10 between consecutive line numbers is retained in this case.

But for a spacing between lines of, say, 5 rather than 10, press:



The first line number is 45, followed by 50, 55, 60 ... The comma must be keyed in to separate the beginning line number from the spacing.

By using automatic line numbering, you can begin with any line number and have any spacing you want. The only stipulation is that no program line number can exceed 9999.

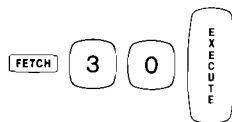


The uses of this key are twofold:

- If you are displaying a line previously entered into memory, you can press the DELETE LINE key to erase it from memory.
- If you are currently keying in a program (using automatic line numbering) and you make a major mistake while keying in a line, you will probably be tempted to press the CLEAR key and start over; don't! If you do, automatic line numbering will cease — of course, it can be re-initiated, but that's extra work. Instead, press the DELETE LINE key; the statement is erased but the line number remains intact.







Any program line currently in memory can be brought to the display. If line 30 is in memory, you can view it by pressing:




Line 30 is immediately displayed. If no specific line number is keyed in, the lowest line number in memory will be displayed.



An entire program can be quickly scanned by using either  or . Each time  is pressed, a higher-numbered program line is displayed (if there is a higher-numbered line).  performs in the opposite manner.

#### RECALL

The RECALL key can be used in programming in the same manner that it is used in calculating. If you attempt to enter a program line into memory and an error appears on the display, just press  and the original program line is returned to the display. Then compare the line to the specified error message and edit your line accordingly. When the line is corrected, press the END OF LINE key to enter it into memory.

#### LIST

Program lines in memory can be printed out for review at any time (assuming your printer is properly hooked up to the calculator). Just press:



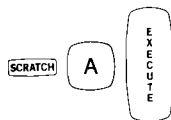
All the program lines currently in memory will be listed in ascending order on your printer. If you have not erased the chess program, it will now be listed. (The number



## A SHORT EXAMPLE

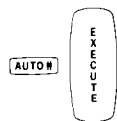
The program on the following fold-out averages a group of numbers. (To run this program, be sure the printer is turned on.)

First press:



Although it's not always necessary to erase memory before keying in a new program, in general, it's a good habit to get into.

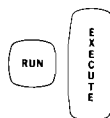
Now key in the program, and to make the inputting easier, press:



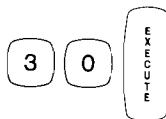
Now the line numbering will be automatically done for you. Each time you press the END OF LINE key, one line is entered into memory and the next line number appears on the display.

## RUNNING THE PROGRAM

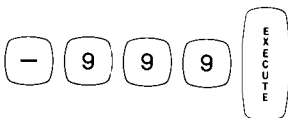
Now let's run the program to see if it was entered properly. We'll use the following data for our inputs: 30, 55, 22.8, 5.67, 11.53. Press:



Program execution begins at the lowest-numbered line. If a '?' does not soon appear on the display, you'd better compare your program listing with ours (see page 49). But if the '?' does appear, press:



This inputs the first number; use the same routine with the other four numbers each time the '?' appears. When all the data is finally input, press:



This causes the results to be printed and the program to be terminated. The output should look like this:

THE AVERAGE OF THE 5       NUMBERS IS 25



If your program did not execute properly, there is a good chance that an error message will now be in the display pinpointing the type of error and the line number in which it occurred.

Whether or not you made an error in this program, the following information will come in handy sooner or later.

## EDITING HINTS

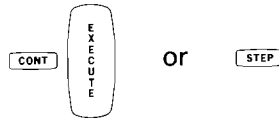
STOP

Generally a program that is running can be immediately halted by pressing the STOP key; but if the program was already waiting at an input statement (a '?' appears on the display), press the keys STOP EXECUTE instead.

If no program lines are then edited, either of the following two keys can be used to continue program execution.

CONT STEP

The program will continue from where it was previously halted if you press either:





With CONT EXECUTE the program will run normally, whereas with STEP it will execute only the next statement and then stop again. Each time STEP is pressed, the line number of the next statement to be executed will appear on the display.

By using the STEP key, you can step through the program line-by-line; this enables you to check the program immediately after each line is executed to see if it's performing as required.




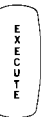
## PROGRAMMING CHECKS

While the program is stopped, various things can be done. A few of the more useful capabilities are as follows:









- Values of the variables can be checked or changed; for example, pressing:

  will display the current value of N,

and pressing:

    will change the value of N, setting it equal to 7.

- The calculator can be set to continue program execution at any line you want; if, for example, the program was stopped at line 60 and you press:

        When program execution is continued, line 110 will be executed first.

- Many other possibilities are described in the Operating and Programming Manual.

## ADDITIONAL CAPABILITIES

Many other editing commands can be used on the Model 30. Included among these are:

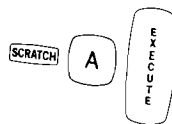
- TRACE — When a program is being executed, this command causes the order of program execution to be printed (by line number).
- NORMAL — With this command, you can get out of the TRACE mode.
- REN (Renumber) — The spacing between program line numbers can be either widened or reduced with this command.
- DEL (Delete) — Any amount of consecutive line numbers can be deleted using this command.

The mechanics of these commands are discussed in the Operating and Programming Manual. Until you begin writing extremely long programs, you will probably find these commands of limited use.

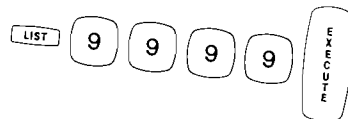
By using the various keys and commands discussed, you can cope with most editing situations that will arise; and by experimenting with the various techniques, you should be able to develop a good intuitive feeling as to when a particular technique is required.

## USABLE MEMORY

Let's determine the total calculator memory before we input another example program. Press:



Now press:



This specific LIST command causes a number to appear on the display. This number indicates the available memory in the calculator, expressed in number of words. (For the computer oriented, 16 bits makes up one word.) Since the calculator was erased just prior to the LIST command, the display is showing the total user memory in the calculator.

After we input the following example, we'll check to see how much user memory is still available.

## A SHORTER EXAMPLE

Another example program is provided on the following fold-out to show you some additional statement types and to reinforce some of the programming concepts you learned earlier.

This program is similar to the previous program in that it averages a group of numbers. But it uses different methods to achieve the same results. Since both examples do the same thing in a slightly different manner, you may want to compare the two programs by leaving the previous fold-out extended.

Let's key it in now.

at here:

a team;

act as a team;

els the PRINT statement.

of the numbers is represented by X1 and the number of  
r, it is necessary to set only  $X1 = 0$  (in line 10) initially,  
value.

ust be accompanied by one or more DATA statements, in  
in the READ statement (N) is assigned the first DATA  
er (an internal mechanism used by the calculator) is then  
ent value (30), which will be the next value assigned in a

a loop with line 60, the NEXT statement. Since  $N = 5$  in  
is equivalent to the actual program statement:

30 FOR I = 1 TO 5

(N) is read, it is possible to have a READ statement with several variables,

The statements in the FOR ... NEXT loop (lines 30 through 60, inclusive) are executed five times each as I is incremented by 1 from 1 to 5. After the I = 5 loop is completed, the line following the NEXT statement (line 70) is executed.

Line 40, another READ statement, assigns a new value to Z each time the FOR ... NEXT loop is executed. The values of Z are obtained from the DATA statement, beginning with the value 30. After each data element is taken, the data pointer moves one more element to the right.

Line 50 keeps a running total of the summation of the inputs each time a new value for Z is read in the FOR ... NEXT loop.

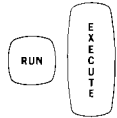
When the FOR ... NEXT loop is completed, line 70, the DISP (Display) statement is executed. Occasionally it is more convenient to put the results of calculations on the display instead of on the printer. Whichever statement seems more appropriate for your needs is the one you should employ.

The same rules that govern the PRINT statement also govern the DISP statement. There are, however, two considerations: if you want a permanent record of the results you should use the PRINT statement; also, if your output takes up more than 32 characters of space, you should use the PRINT statement since the DISP statement can display only 32 characters at one time.

Line 80, the DATA statement, is accessed only when a READ statement is executed. So after line 70 is executed, the DATA statement is ignored and line 90, the END statement, is executed and the program halts.



Let's execute the program by pressing:



The following output should appear on the display:

THE AVERAGE IS 25

When this program is run, there is no interaction between the user and the program as there is in the first example program. This program runs through till completion and displays the results.

If you want to run the program with different data, just change line 80 by putting in your own data. The number of items that you want to have averaged is up to you. However, the first element of the DATA statement must accurately specify the number of items to be averaged (N); otherwise, the program will not correctly average the numbers.

We mentioned that data can appear in one or more DATA statements. Thus, if we had 6 items to be averaged (say, 83, 84, 92, 76, 95, 80), the following representations would all be equivalent:

80 DATA 6, 83, 84, 92, 76, 95, 80

or

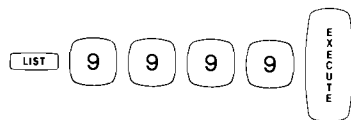
80 DATA 6  
81 DATA 83, 84, 92, 76, 95, 80

or

80 DATA 6  
81 DATA 83, 84, 92  
82 DATA 76, 95, 80

## DETERMINING MEMORY

If you'd like to know how much memory it took to input and run this program, press:

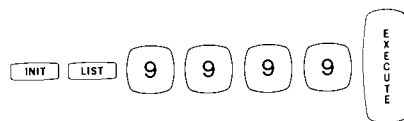


The available memory appears on the display.

To determine how many words of calculator memory this program used, subtract the available memory (shown on the display) from the total memory (determined on page 57). It took up only about 82 words of the total memory.

Although this is good to know, it's not the information you need to store the program on a cassette file. The information needed is the size of the program and the 82 words not only include this, but also include the amount of memory used to execute the program.

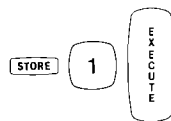
To determine just the program size, press:



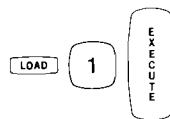
The maximum available memory minus the number now on the display determines the actual program size (about 58 words).

The INIT (Initialize) key resets the calculator to the state it was in immediately prior to program execution. (This is always true unless your program has a COM or DIM statement in it — these statements and the INIT key are discussed thoroughly in the Operating and Programming Manual.)

Just for practice, you might like to store this program on a cassette file. Any file that was marked with a length large enough to contain this program can be accessed. We previously marked FILES 0 through 2 with 500 word lengths. Since these files are larger than the program length, any of them can be used to store the program. For instance, press:



This puts a copy of the program into FILE 1 of the cassette. If at some later date, after the program is erased from memory, we wish to load it back into memory, we can press:



This will put a copy of the program back into memory.

### NOTE

Be careful not to confuse the roles of STORE and LOAD. Pressing the incorrect sequence can result in an erased program!

## ADDITIONAL STATEMENTS

There are several additional statements in BASIC, not mentioned in this book because you won't need many of them initially. We will, however, briefly mention three more statement types that you may find of some use as the complexity of your programs increases:

- REM (Remark) — This statement is merely a note to the programmer and as such, it can be located wherever you want; it is not executed by the program. For example:

```
10 REM THIS PROGRAM IS ABOUT ...
```

- STOP — This statement can be used instead of END; it's particularly useful when stacking programs (that is, having more than one program in memory at a given time). Whereas encountering the END statement causes the program line counter to revert to the beginning line in memory, the STOP statement will cause the program line counter to retain its position; so if there is another program after the STOP, it can be accessed by pressing the CONTINUE and EXECUTE keys.

● GO SUB (Go To Subroutine) and RETURN — Lines needed over and over can be accessed by the GO SUB statement. If the subroutine begins at line 2000, you would access it with a statement like:

```
80 GO SUB 2000
```

The last line in the subroutine is a RETURN statement; when it is encountered, the line following the GO SUB statement is accessed (in this case, the one after line 80).

## LOOSE ENDS

The FOR ... NEXT statements discussed earlier do not have to be incremented by 1 after each loop; for example,

```
10 FOR D = 0 TO 360 STEP 5  
  .  
  .  
  .  
  .  
  .  
90 NEXT D
```

This loop will be executed 73 times: when  $D = 0, 5, 10, 15, \dots$ . By adding STEP to the FOR statement, you can increment the variable by anything you want.

The IF ... THEN statement discussed earlier uses relational and/or logical operators. For example, in:

```
20 IF Z = 8 THEN 60
```

The equals sign is a relational operator; all the following relational operators can be used in BASIC:

equality	=
inequality	#
greater than	>
less than	<
greater than or equal to	> =
less than or equal to	< =

The following logical operators can also be used:

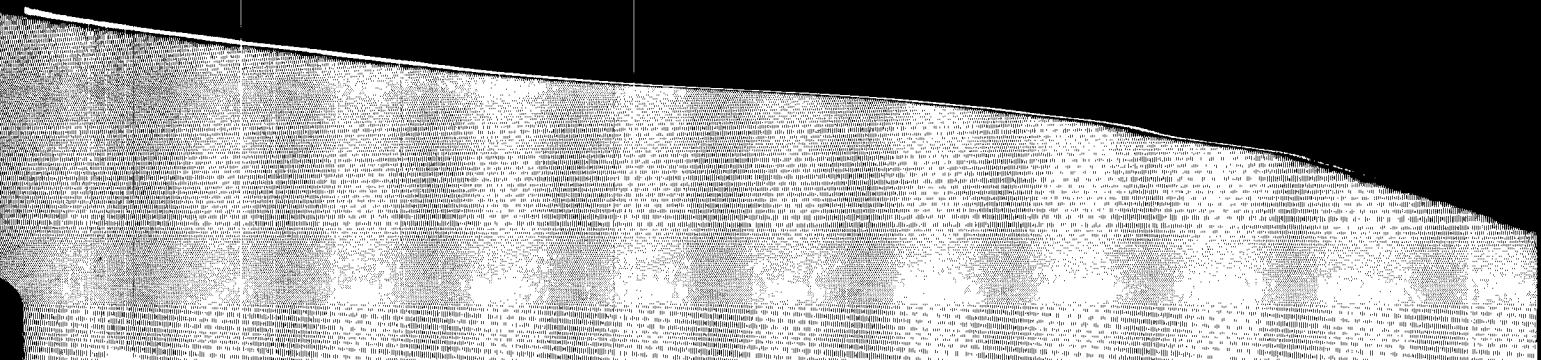
```
AND  
OR  
NOT
```

Logical evaluation can be quite complex; for this reason, we will avoid the topic here by referring you to the Operating and Programming Manual, where it is thoroughly discussed.

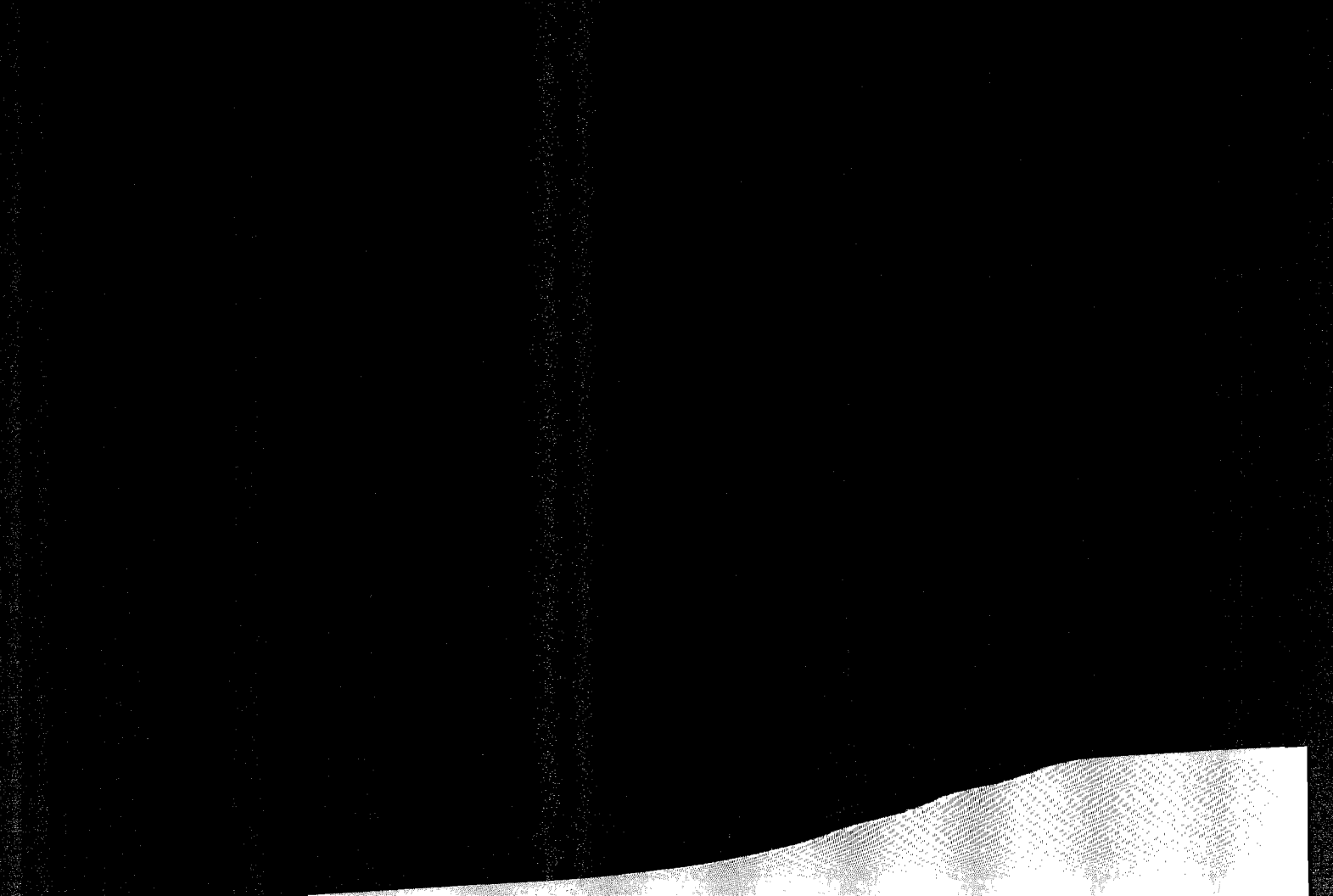
If you are familiar with the BASIC language, you may have been surprised not to see the LET statement. You can use the LET statement on the Model 30; however, we find it more convenient to use the 'implied' LET statement. Thus:

LET X1 = X1 + X is equivalent to X1 = X1 + X

If you are familiar with FORTRAN programming, you will be pleased to know that we also have FORMAT and WRITE statements, which parallel those in FORTRAN. The Operating and Programming Manual shows how to obtain various printouts using these statements.







## **HEWLETT-PACKARD SALES AND SERVICE OFFICES**

For a complete list of world-wide, Hewlett-Packard Sales and Service Offices, please refer to your Calculator Operating Manual.

### **IN THE UNITED STATES**

#### **CALIFORNIA**

3939 Lankershim Blvd.  
North Hollywood 91604  
Tel: (213) 877-1282

#### **GEORGIA**

P.O. Box 28234  
450 Interstate North  
Atlanta 30328  
Tel: (404) 436-6181

#### **ILLINOIS**

5500 Howard Street  
Skokie 60076  
Tel: (312) 677-0400

#### **NEW JERSEY**

W. 120 Century Road  
Paramus 07652  
Tel: (201) 265-5000

### **IN CANADA**

#### **QUEBEC**

Hewlett-Packard (Canada) Ltd.  
275 Hymus Blvd.  
Pointe Claire  
Tel: (514) 697-4232

### **IN EUROPE**

#### **SWITZERLAND**

Hewlett-Packard S.A.  
7 rue du Bois-du-Lan 7  
CH-1217 Meyrin 2, GE  
Tel: (022) 41 54 00

### **REST OF THE WORLD**

Hewlett-Packard  
INTERCONTINENTAL  
3200 Hillview Avenue  
Palo Alto, California 94304 U.S.A.  
Tel: (415) 326-7000



PART NO. 09830-90000  
MICROFICHE NO. 09830-99000

PRINTED IN U.S.A.  
FEB 1973