MODEL 20

HEWLETT-PACKARD 9820A CALCULATOR

SIMPLIFIED OPERATING INSTRUCTIONS

WHY THIS BOOK?

The Operating and Programming Manual supplied with your Calculator describes the Calculator, very completely, in a formal way. Being the 'formal' manual, it includes essentially all operating and programming procedures for the Calculator. When the manuals for earlier HP calculators were written they were organized to serve the dual purpose of teaching the user how to operate and program the calculator, and of acting as quick-reference manuals. That type of organization required that, in each manual, much of the material be repeated in several places. Obviously, if the same technique were to be applied to the manual for a calculator as sophisticated as the 9820, the size of the manual would become ridiculous.

The formal manual is, therefore, organized as a text-book; which means that you may well have to read many pages before you learn to actually do anything. That's why this book came into being — to enable you to start using your calculator almost immediately.

This book is arranged into two parts: The first part explains how to make keyboard calculations and how to run pre-written programs. The second part is an introduction to writing programs for the 9820, based on the assumption that the reader has never before written a program.

This book (the term 'manual' has been deliberately avoided) contains only selected topics, and does not contain complete operating and programming information for the 9820: for that you must refer to the formal manual. However, even if you are an experienced programmer, you may well find that an hour or two spent with this book will pay large dividends when you start to work with the formal manual.

Should you care to comment about this book, there is a prepaid reply card in the operating and programming manual.

HEWLETT *hp* PACKARD

SIMPLIFIED OPERATING INSTRUCTIONS

MODEL 9820A CALCULATOR

TABLE OF CONTENTS

Part 1

RUNNING THE CALCULATOR

INTRODUCTION TO THE CALCULATOR

The -hp- 9820A Calculator can be used equally well as a simple office machine or as a sophisticated programmable calculator. It can be tailored, by means of various plug-in ROM's (see Page 3) and peripheral devices, to suit any particular requirements. Even without any such additions, the basic machine* is still a powerful calculator and is fully programmable.

The basic machine, shown in Figure 1, contains a magnetic card reader and an alphameric printer as standard equipment. Its memory consists of 179 registers (see Page 22); calculators which have Option 001 installed have 256 more registers, giving a total of 435. Lift the lid on the top of the calculator - if an option has been installed, there will be an identifying decal located in the indent behind the printer (see Figure 1). The Option 001 decal shows 429 registers, not 435; the reason for this is explained with the description of the memory, later in this book.

The basic keyboard is shown on the foldout at the back of this manual.

*This book describes only the basic calculator; information for each plug-in and peripheral device is contained in its own manual.
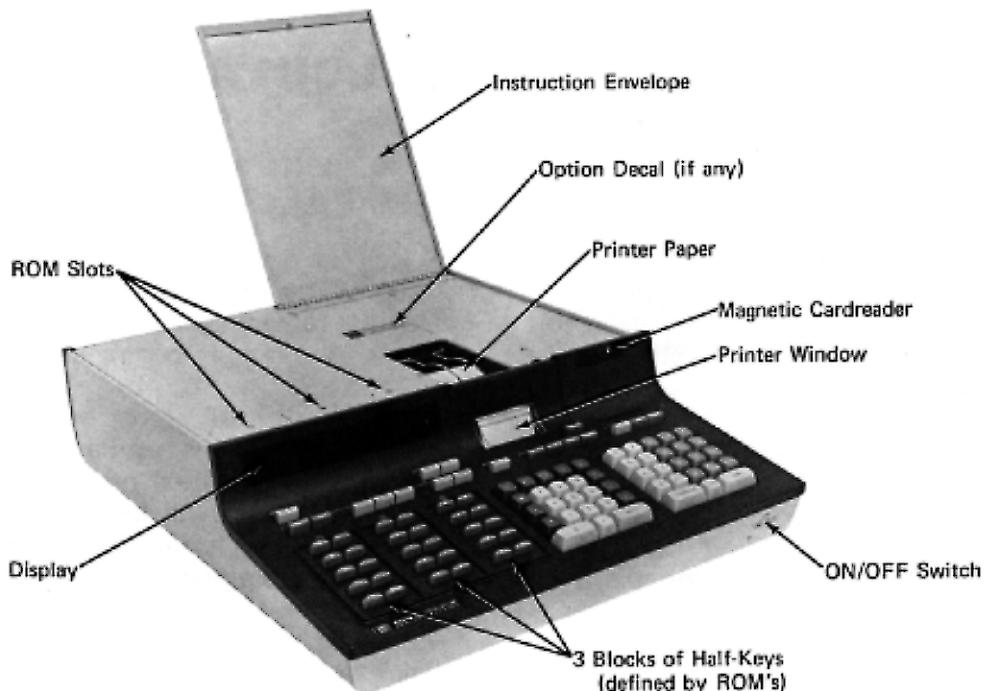
Figure 1. Identifying the 9820A Features

ROM'S AND THE HALF-KEY BLOCKS

The three blocks of half-keys (not including those along the top of the keyboard) are used with the optional 'Read-only-memories' (ROM's). The ROM's plug into the slots on top of the calculator (Figure 1). In general, each ROM uniquely determines the meaning of the half-key block located immediately in front of it; for example, the Mathematics ROM assigns mathematical functions such as sine, cosine and tangent and a Peripheral Control ROM enables the calculator to control other devices, such as a plotter and a digitizer (some ROM's may affect more than one of the key blocks).

If there is no plug-in ROM installed, the half-keys are used to print or display the characters which appear on them. Any attempt to use these keys for other purposes results in a diagnostic note (NOTE 11) being displayed; see Page 9 for an explanation of the diagnostic notes.
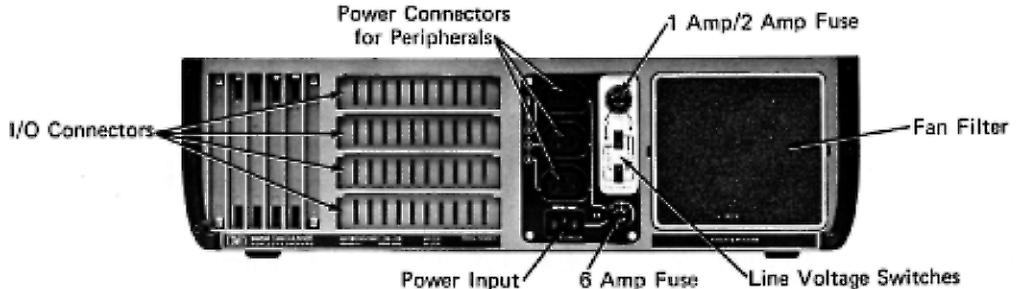
Figure 2. The 9820A Rear Panel

TURN-ON PROCEDURE

> **NOTE**
>
> Refer to the Operating and Programming manual for complete information regarding power and grounding requirements, fuses, and so on. Instructions to run the calculator's exerciser program are included in the Model 20 System Electrical Inspection booklet.

The calculator operates with power-line voltages of (nominally) 100, 120, 220 or 240 volts. Table 1 indicates the operating range, and the fuse required, for each nominal voltage. Before turning on the calculator, ensure that the two slide-switches on the rear panel are correctly set to the voltage whose operating range covers the line-voltage in your area (also ensure that the correct fuse is installed). Figure 2 shows the location of the switches, Figure 3 shows the setting of the switches for each nominal voltage.

Table 1. Power-Line Voltages

NOMINAL VOLTAGE	OPERATING RANGE (−10%, +5% of nominal)	FUSE
100 volts	90 to 105 volts	2 amp
120 volts	108 to 126 volts	2 amp
220 volts	198 to 231 volts	1 amp
240 volts	216 to 252 volts	1 amp

| 100 volts | 120 volts | 220 volts | 240 volts |
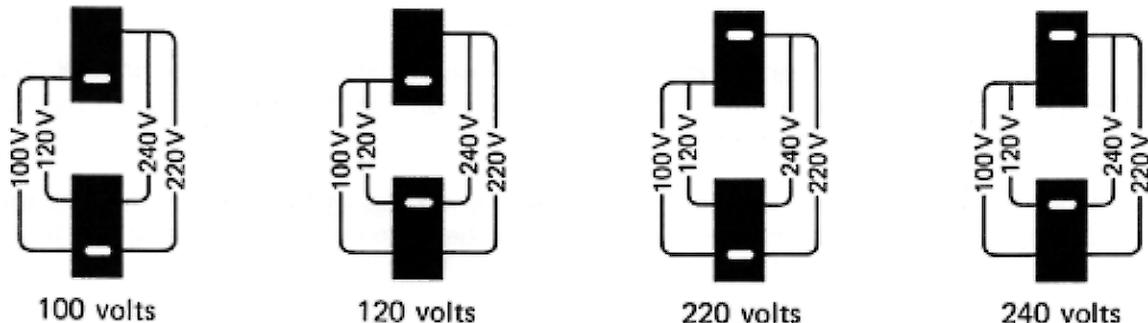
Figure 3. Switch Settings for Nominal Power-Line Voltages

1. If the calculator is not plugged in: Plug one end of the power cord into the lowest of the four sockets at the back of the calculator (Figure 2); plug the other end of the cord into a suitable power outlet, such as a wall-socket. Plugs and connectors are keyed, they cannot be connected improperly.

2. If the calculator is switched off: The OFF/ON switch is located on the front of the calculator, below the keyboard and to the right (Figure 1). Set the switch to the ON position; when the following display appears, the calculator is ready to operate.

```
0:  END ⊢
```

LOADING PRINTER PAPER

`PAPER`   The paper key advances the printer paper as long as the key is being pressed. Do not press PAPER if the printer is printing or if a program is running. PAPER is purely a mechanical key; it cannot be operated from the program.

Printer paper is loaded into the well underneath the flap on top of the calculator. Three rolls of paper are supplied with the calculator; extra rolls can be ordered from HP using the following part numbers:

        Pack of six rolls — -hp- Part No. 9281–0401–006

        Case of sixty rolls — -hp- Part No. 9281–0401–060

To load a new roll:

1.   Lift the bail (see Figure 4); remove and discard the paper core from the previous roll (if any).

2.   Remove the first layer of paper from the new roll and insert the roll, with the free end positioned as shown in Figure 4, into the printer. Lower the bail.

3.   Hold the PAPER key pressed until the paper emerges from behind the printer window. The paper is now loaded and the printer ready to use.

To remove an unfinished roll:

1. Unroll the paper until the roll can be lifted out of the printer.

2. Hold the roll firmly and pull it up and forward; the paper guide will tear the paper off cleanly.

3. Press the PAPER key to push out the remaining paper.
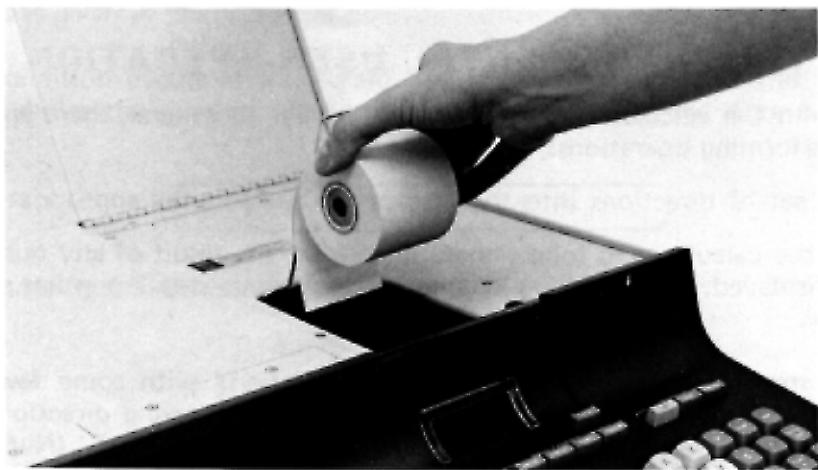
Figure 4. Loading Printer Paper

INITIALIZING THE CALCULATOR

ERASE The ERASE key has the same effect as switching the calculator off and then on again; it erases all stored data and programs from memory, and clears the results of any previous calculation or operation.

THE FUNDAMENTAL USER-OPERATION

Communication with the calculator is through the display. In general, there are two basic steps to follow when performing operations:

1. You 'write' a set of directions into the display (by pressing the appropriate keys) and then

2. You instruct the calculator to follow these directions; the result of any numerical operation is automatically displayed. When making keyboard calculations, step 2 consists solely of pressing the EXECUTE key.

These two basic steps form the 'fundamental user-operation;' with some few exceptions, all operations — making calculations, loading or running programs, giving directions to the printer, etc. — consist of some variation of the 'fundamental user-operation'. (Numerous examples appear in this book; almost all of them can be cited as examples of the fundamental user-operation.)

DIAGNOSTIC NOTES

In addition to displaying numbers, directions, and the results of operations, the calculator also displays diagnostic notes to inform you of operational errors or of special situations. The basic notes are numbered from 01 to 16 (higher numbered notes are associated with the various plug-in ROM's); the note number indicates the type of error or situation. For example, NOTE 01 indicates that you gave the calculator a direction which it could not understand; NOTE 16 indicates that the printer has run out of paper. A list of the basic notes and a brief description of their meanings is given in the appendix.

When a 'note' condition occurs in a program, the program stops; as well as showing the note, the display also shows the number of the program line in which the note condition occurred; e.g.,

```
NOTE 02 IN 4
```

indicates that a note 02 condition occurred during line 4.

KEYING DIRECTIONS AND NUMBERS

Directions are written into the display by pressing the appropriate keys. Suppose, for example, that you want to add 2 to 4 and print out the result; you press keys PRINT 2 + 4. The calculator does not, however, follow your directions until it is instructed to do so, by your pressing EXECUTE. It then prints (and displays) the result, 6.

(As you have not yet been shown how to initialize the display — see Positioning the Decimal Point, on Page 13 — the printout from the above keying sequence will probably consist of 6 followed by the decimal point and some zeros; also, it may end with 'E 00'.)

Numbers are keyed into the display, as on any standard office-machine, by pressing the number keys (0 through 9) and the decimal point key in the required order. If a number is negative the 'minus sign' should be keyed first before the number is keyed. Use of commas (such as in 32,341.6) is not allowed. (See 'Making Arithmetic Calculations' for examples of typical keying sequences.)

As is the case with a direction, even though the keyed number is displayed, it will not be 'executed' by the calculator until the EXECUTE key is pressed (step 2 of the Fundamental User Operation). An executed number may appear in a different form from the original number, although it will still have the same value; the reason for this is explained under 'Fixed- and Floating-Point Numbers' on Page 30. You would not, of course, normally want to execute just a single number; the number would be included in some set of directions and then the directions would be executed.

USE OF CLEAR

The CLEAR key clears the display; it operates immediately and does not have to be followed by EXECUTE. An 'end of line' symbol (⊢) appears in the display when CLEAR is pressed; this indicates that the calculator is 'idle'.

It is not necessary to clear the display before keying the next direction as long as the previous direction has been executed; in this case use of CLEAR is optional. If no subsequent execution has taken place since the last direction was keyed, then CLEAR must be used.

NOTE

As indicated on the inside front-cover, there is a great deal of information about the 9820A Calculator which is not included in this book. It is, therefore, conceivable that you might inadvertently press some combination of keys which sets off some internal activity that 'locks-out' the keyboard — the display may blank and even the CLEAR key may have no effect!

If this should occur, keyboard operation can be returned by pressing ERASE; or if you do not wish to erase data and programs stored in the memory, by first pressing STOP and then pressing CLEAR.

PRINTING KEYBOARD-OPERATIONS

To make a printed record of your keyboard-operations, press the following key sequence:

TRACE EXECUTE

This establishes the 'trace' mode so that, as you perform subsequent operations, the printer prints the operations performed, and the result, each time the EXECUTE key is pressed. (A few keys, such as CLEAR, are not printed.)

To discontinue tracing, press the key sequence:

NORMAL EXECUTE
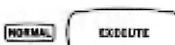
that operation will be printed and then tracing will cease.

POSITIONING THE DECIMAL POINT

The FIXED N key, followed by any one of number keys 0 through 9, establishes the number of digits to be seen to the right of the decimal point when the result of a calculation is displayed or printed. For example, if you are working with whole numbers you will not want to see any digits to the right of the decimal point; in this case press

FIXED N (0) (EXECUTE)

If you are working with dollars and cents, two digits are sufficient:

FIXED N (2) (EXECUTE)
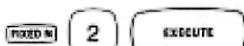
or, if hundredths of a cent are to be seen,

FIXED N (4) (EXECUTE)
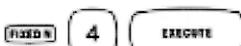
Often you will see a number which contains the letter 'E'; this type of number is known as a 'floating-point' number, discussed on Page 30.

MAKING ARITHMETIC CALCULATIONS

For arithmetic, the fundamental user-operation consists of writing an arithmetic expression into the display and then pressing the EXECUTE key, to instruct the calculator to evaluate that expression.

```
                              NOTE
    Keying examples are included in the text for you to perform if
    you wish to do so; the EXECUTE key is not shown in every case.
    When a complete expression has been written into the display,
    press the EXECUTE key if you want the calculator to evaluate
    that expression. If you do not execute an expression, remember to
    press CLEAR before keying the next one.

    If you make an error when keying an expression either press the
    CLEAR key or refer to 'Making Corrections' on Page 20.

    Before starting the examples, initialize the display by pressing the
    following keys:

              CLEAR  FIXED N  ( 2 )  ( EXECUTE )
```

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

Keys	Display
(4) (+) (8)	`4+8`
(EXECUTE)	`12.00`
(() (4) (+) (8) ()) (/) (() (5) (−) (2) ())	`(4+8)/(5-2)`
(EXECUTE)	`4.00`

THE ARITHMETIC OPERATORS

Apart from the number keys the following operational keys are available for writing arithmetic expressions (press EXECUTE after each expression).

[+] Addition:

$$(3)(+)(6) \qquad \boxed{9.00}$$

[−] 1. Subtraction:

$$(9)(\cdot)(3)(-)(6) \qquad \boxed{3.30}$$

2. Unary minus (negative of a quantity):

$$(-)(7) \qquad \boxed{-7.00}$$

$$(6)(*)(()(-)(7)()) \qquad \boxed{-42.00}$$

✳ Multiplication:

⑥ · ② ⑤ ✳ ④   33.00

(⑦ − ②) ✳ (④ + ⑨)   65.00

/ Division:

③ / ⑤   .60

⑥ ✳ ③ / (① ① − ②)   2.00

√ Square root:

√ ③   1.73

√ ④ + ⑤   7.00

√ (④ + ⑤)   3.00

17

THE ARITHMETIC OPERATORS (Continued)

( )

1. Grouping: as in the above examples, quantities in parentheses are treated as one quantity. Thus $\sqrt{(4+5)}$ is equivalent to $\sqrt{9}$, whereas, $\sqrt{4}+5$ adds 5 to the square root of 4.

2. Implied Multiplication: 4(3+2) is the equivalent of 4*(3+2); in this case use of the 'multiply' sign is optional.

Parentheses can be nested (i.e., parentheses inside parentheses, inside parentheses, etc.) but they must always be balanced, that is, there must be the same number of left-handed parentheses as there are right-handed.

$\sqrt{}$ ((8 + 4) / (1 . 5 * 2) + 5) + 7

$$10.00$$

Notice that the preceding example contains too many keystrokes to be displayed all at one time and that the display shifts left to accomodate the 'extra' keystrokes.

THE ARITHMETIC HIERARCHY

When an arithmetic expression contains more than one operator, as do several of the preceding examples, there is a prescribed order of execution; an expression must be properly written or the answer will be wrong. The order of execution, known as the 'hierarchy', is exactly the same as the order commonly used in standard arithmetic;

1. Mathematical Functions (only the square root on the basic calculator);

2. Implied multiplication;

3. Multiplication and division;

4. Addition and subtraction.

Where an expression contains two or more operators at the same level in the hierarchy, they will be executed in order from left to right.

The use of parentheses enables the order of execution to be changed; thus in the $\sqrt{(4 + 5)}$ the '+' is executed before the '$\sqrt{}$' even though the '+' occupies a lower level in the hierarchy.

EXCEEDING THE LENGTH OF THE DISPLAY

The length of an expression is not limited to the length of the display. When you keyed the last example under The Arithmetic Operators (Page 18), you saw that the expression contained too many symbols to be displayed at one time; as each 'excess' symbol was keyed, the display shifted left to make room. The maximum allowable length for an expression varies between 35 and 69 keystrokes, depending upon the nature of the expression. If too many keys are pressed, the display shows 'NOTE 09' (see the Appendix); depending upon the nature of the expression, the note may appear either before or after the EXECUTE key is pressed. In either case, press CLEAR and write a shorter expression.

MAKING CORRECTIONS

[BACK] [FORWARD]

The BACK and FORWARD keys enable a displayed expression to be altered or corrected without re-keying the entire sequence.

1. If a wrong key is pressed when writing an expression, it can be corrected immediately by pressing the BACK key and then pressing the correct key.

(2) (+) (BACK) (*) (4)   2÷4

2.   A displayed expression can be blanked, key by key in reverse order, by pressing BACK once for each displayed key. The blanked keys can then be returned to the display one at a time by pressing FORWARD. If an expression contains a wrong key, press BACK until that key is blanked, press the correct key and then press FORWARD to return each subsequent key (or, if extra keystrokes are required, key in the remainder of the expression). For example, suppose the number 123456789 is keyed incorrectly into the display:

(1) (2) (3) (4) (4) (4) (7) (8) (9)   [123444789]

To correct, press:

[BACK] [BACK] [BACK] [BACK] [BACK]   [1234]

and then press:

(5) (6) [FORWARD] [FORWARD] [FORWARD]   [123456789]

3.   If the incorrect expression has been executed but no key has since been pressed, the expression can be returned to the display (by pressing BACK), corrected as before, and then again executed.

THE DATA MEMORY

The calculator's memory serves the dual purpose of storing data-numbers and of storing programs. The memory-map of Figure 5 shows that the memory is divided into rows, referred to as 'registers'. Six registers are given alphabetic names and have their own keys (A, B, C, X, Y, Z). The rest have alphameric names — R (for 'register') followed by numbers, starting with zero; the R-registers do not have their own keys but are selected by pressing the R( ) key and following it by the appropriate number keys.
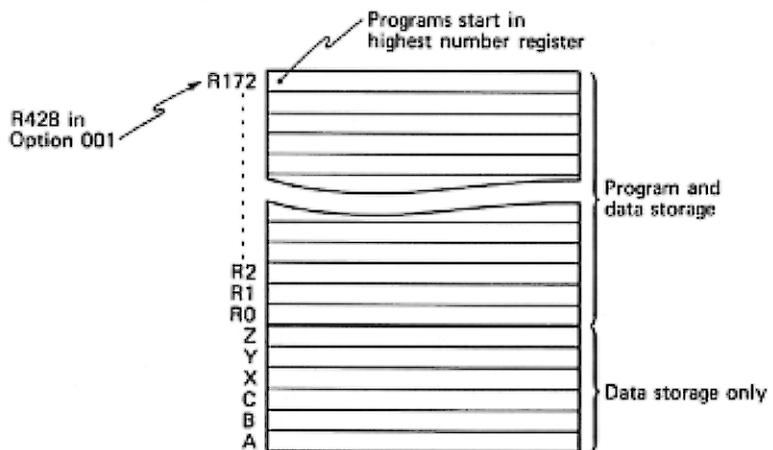

Figure 5. The Memory Map

The basic calculator contains 179 registers: six alphabetic and 173 R-registers (R0 through R172). Calculators with the Option 001 memory installed have an additional 256 R-registers (R173 through R428) giving a total of 435 registers (the decal shows 429 because it does not include the six alphabetic registers).

Some of the ROM blocks (see Page 3) require part of the memory for their own use. When one of these ROM's is installed, it automatically takes the required registers, starting at the highest numbered register and working downwards. Those registers are then temporarily not available for program or data storage, until the ROM is removed.

When programs are stored they start in the highest-numbered available R-register and sequentially fill the memory downwards; programs cannot be stored in the alphabetic registers. It is, therefore, most convenient to store data first in the alphabetic registers and then in the lower numbered R-registers. If the memory contains no program (i.e. at turn-on, or if ERASE has been pressed), then all registers (except those required by a ROM) will be available for data storage. If the memory does contain a program, then the higher-numbered registers will not be available for data; 'NOTE 06' will be displayed if you attempt to store data in a register which is not available.

The number of available R-registers can be determined at any time by pressing CLEAR LIST STOP. The printer will start to list the program (the STOP saves having to wait for the whole program to be listed); at the bottom of the list will be a number preceded by the letter R, indicating the number of R-registers available. (The lowest-numbered register is R0; subtract 1 from the number printed to obtain the name of the highest-numbered register available for data storage).

STORING DATA

One register can contain one data-number. It is not necessary to clear a register before storing a number in it because the number being stored automatically substitutes for the existing stored number. The entire memory is, however, cleared at turn-on or if ERASE is pressed.

→ Storing data requires use of the 'goes into' key:

(1)(2)(·)(6)(→)(A)(EXECUTE)
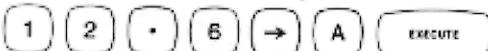
stores the value (12.6) in the A register. Similarly,

(6)(→)(X)(EXECUTE)
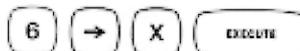
stores 6 into register X and

(1)(9)(→)(R( ))(1)(2)(EXECUTE)
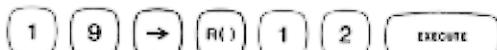
stores 19 into R12.

If you wish to see a stored number, use either the DISPLAY or the PRINT keys:

(DISPLAY)(A)(EXECUTE)

displays the number currently stored in A; the number remains stored in A.

(PRINT)(R( ))(1)(2)(EXECUTE)
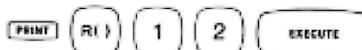
prints the contents of R12; the number remains stored in R12.

IMPLIED Z

In general, if a stored number is to be kept for any length of time it should not be stored into the Z register because of the 'implied Z' feature. The result of any arithmetic expression is automatically stored in Z if no other place is specified; thus

$$1 \quad 4 \quad \cdot \quad 2 \quad \boxed{\text{EXECUTE}}$$

is equivalent to

$$1 \quad 4 \quad \cdot \quad 2 \quad \rightarrow \quad Z \quad \boxed{\text{EXECUTE}}$$
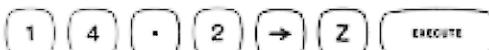
both expressions display 14.2 and also store it in Z. Similarly,

$$3 \quad * \quad 4 \quad + \quad 1 \quad 6 \quad / \quad 3 \quad \boxed{\text{EXECUTE}}$$

is equivalent to

$$3 \quad * \quad 4 \quad + \quad 1 \quad 6 \quad / \quad 3 \quad \rightarrow \quad Z \quad \boxed{\text{EXECUTE}}$$
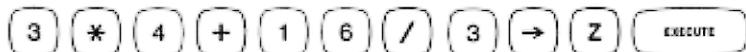
ARITHMETIC WITH REGISTERS

Arithmetic expressions can be written using register names instead of actual numbers. When the expression is executed, the values currently stored in those registers will be automatically substituted for the register names in order to evaluate the expression. For example, if you have just performed the examples under 'Storing Data' (Page 24), the following numbers should be currently stored in the memory:

12.6	in	A
6	in	X
19	in	R12

With the above values stored, keying

$$\boxed{A}\ \boxed{+}\ \boxed{R(\ )}\ \boxed{1}\ \boxed{2}\ \boxed{-}\ \boxed{X}\ \boxed{\text{EXECUTE}}$$
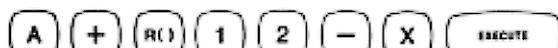
would be equivalent to keying

$$\boxed{1}\ \boxed{2}\ \boxed{\cdot}\ \boxed{6}\ \boxed{+}\ \boxed{1}\ \boxed{9}\ \boxed{-}\ \boxed{6}\ \boxed{\text{EXECUTE}}$$

Other values stored in these registers would, of course, give a different result for the same expression.

Numbers and register-names can be mixed in an expression:

$$\boxed{3}\;\boxed{A}\;\boxed{+}\;\boxed{4}\;\boxed{-}\;\boxed{X}\;\boxed{\text{EXECUTE}}$$
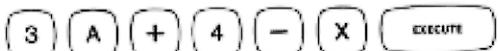
would be the equivalent of keying

$$\boxed{3}\;\boxed{*}\;\boxed{1}\;\boxed{2}\;\boxed{.}\;\boxed{6}\;\boxed{+}\;\boxed{4}\;\boxed{-}\;\boxed{6}\;\boxed{\text{EXECUTE}}$$

(again assuming that the values previously stored in A and X have not been changed).

TYPICAL USES OF THE DATA MEMORY

Following are two examples illustrating the type of calculation which you can make using the memory.

Example 1. Storing a value which is to be used many times.

Suppose the unit-price of some item is $132.57 and you are to calculate the cost of buying various quantities. To save keying in the price for each calculation, you can store it and then refer to it by register-name.

Assume you have the following list of quantities: 47, 29, 36, etc.

First store the unit-price into the memory (say in register A):

$$\boxed{1}\;\boxed{3}\;\boxed{2}\;\boxed{.}\;\boxed{5}\;\boxed{7}\;\boxed{\rightarrow}\;\boxed{A}\;\boxed{\text{EXECUTE}}\qquad\boxed{132.57}$$

TYPICAL USES OF THE DATA MEMORY (Continued)

Now make the calculations for each quantity in turn using the price stored in A:

(4) (7) (A) (EXECUTE)   [6230.79]

(2) (9) (A) (EXECUTE)   [3844.53]

(3) (6) (A) (EXECUTE)   [4772.52]

and so on for each quantity.

Example 2. Accumulating a total.

Suppose you have a (long) list of numbers to be added: 9, 36, 25, ..., etc.

The total can be accumulated, one number at a time, in a storage register. Be sure that the register is first cleared (store zero) or the final total may be wrong.

(0) (→) (B) (EXECUTE)   [0.00]

(B) (+) (9) (→) (B) (EXECUTE)   [9.00]

28

$\boxed{B}\ \boxed{+}\ \boxed{3}\ \boxed{6}\ \boxed{\rightarrow}\ \boxed{B}\ \boxed{\text{EXECUTE}}$ $\boxed{45.00}$

$\boxed{B}\ \boxed{+}\ \boxed{2}\ \boxed{5}\ \boxed{\rightarrow}\ \boxed{B}\ \boxed{\text{EXECUTE}}$ $\boxed{70.00}$

and so on for each number on the list. The accumulative total will be displayed and stored in register-B at each step.

Because of the 'implied Z' the above sum can also be made as follows:

$\boxed{0}\ \boxed{\text{EXECUTE}}$

$\boxed{Z}\ \boxed{+}\ \boxed{9}\ \boxed{\text{EXECUTE}}$

$\boxed{Z}\ \boxed{+}\ \boxed{3}\ \boxed{6}\ \boxed{\text{EXECUTE}}$

$\boxed{Z}\ \boxed{+}\ \boxed{2}\ \boxed{5}\ \boxed{\text{EXECUTE}}$

the total will be displayed and stored in Z. (Page 40 contains a program which shows how to even further simplify totalling a list.)

FIXED- AND FLOATING-POINT NUMBERS

Numbers can be keyed into the display and displayed in either of two forms: 'fixed-point' or 'floating-point'.

'Fixed-point' means that the number appears as commonly written, with the decimal point correctly located (e.g. 123.45).

'Floating-point' numbers are written with the decimal point immediately following the first digit (discounting leading zeros) and with an exponent. The exponent, which represents a positive or negative power of ten, indicates the direction, and the number of places, that the decimal point should be moved, to express the number as a fixed-point number. In the calculator the exponent can be any integer within the range −99 to +99.

Examples:

	FIXED	FLOATING
a.	1234.5	$= 1.2345 \times 10^3$
b.	0.0012345	$= 1.2345 \times 10^{-3}$
c.	1.2345	$= 1.2345 \times 10^0$

(exponent)

[FIXED N] selects fixed-point display of (executed) numbers. The letter N indicates that the key must be followed by one of the number keys (0 through 9) to select the number of digits to be displayed to the right of the decimal point.

For example:

FIXED N [4] (EXECUTE)

(1)(2)(3)(.)(4)(5)(6)(7)(8)(9) | 123.456789 |

(EXECUTE) | 123.4568 |
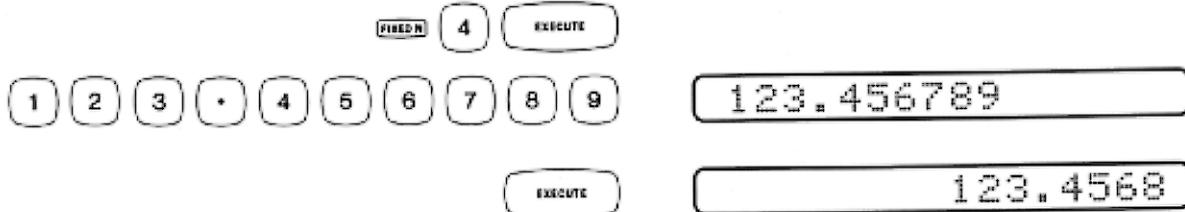
notice that the last displayed digit is automatically rounded up from 7 to 8 because the next (non-displayed digit) is greater than 5. Even though it was rounded in the display, the number was not changed:

FIXED N [6] (EXECUTE) | 123.456789 |

FLOAT N   FLOAT N (and its associated numeric key) operate in the same way as FIXED N (and its numeric key) except that floating-point display is selected. (When the calculator is first turned on, FLOAT 9 is automatically assumed.)

For example: (with the number from the previous example in the display)

FLOAT N [9] (EXECUTE) | 1.234567890E 02 |

FIXED— AND FLOATING—POINT NUMBERS (Continued)

The 'E' indicates that the next two digits constitute the exponent. If the exponent is negative the minus sign follows the E.

(·) (0) (0) (1) (2) (3) (4) (EXECUTE)   `1.234000000E-03`

NOTE

No more than ten significant digits can be displayed; therefore, if a number becomes too large to be properly displayed as a fixed-point number, it will be automatically displayed as a floating-point number. If the number becomes too small, only zeros are displayed but the number can still be seen if 'floating' is then selected.

ENTER EXP — 'Enter Exponent' is used to designate the E (exponent) when numbers are being keyed in floating-point form.
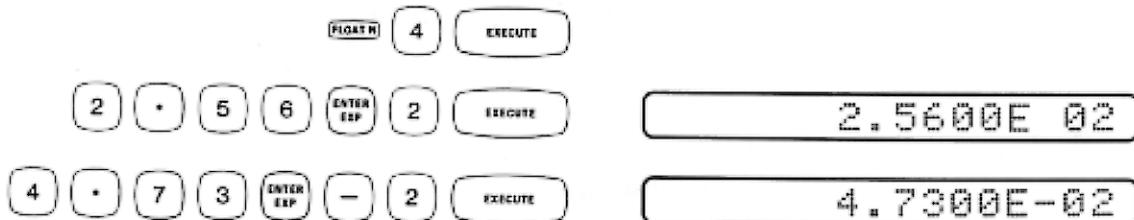
(FLOAT N) (4) (EXECUTE)

(2) (·) (5) (6) (ENTER EXP) (2) (EXECUTE)   `2.5600E 02`

(4) (·) (7) (3) (ENTER EXP) (—) (2) (EXECUTE)   `4.7300E-02`

When keying numbers in floating-point, the decimal point need not always be keyed immediately after the most significant digit; for example, 12345 could be keyed in many different ways, here are a few of them:

$$\boxed{1}\;\boxed{2}\;\boxed{3}\;\boxed{4}\;\boxed{5}$$

$$\boxed{1}\;\boxed{\cdot}\;\boxed{2}\;\boxed{3}\;\boxed{4}\;\boxed{5}\;\boxed{\text{ENTER EXP}}\;\boxed{4}$$

$$\boxed{1}\;\boxed{2}\;\boxed{3}\;\boxed{\cdot}\;\boxed{4}\;\boxed{5}\;\boxed{\text{ENTER EXP}}\;\boxed{2}$$
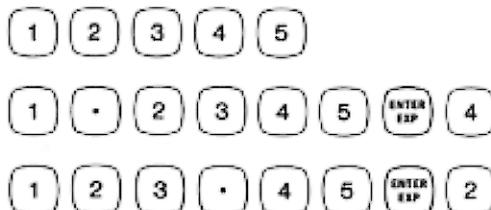
RANGE OF CALCULATION

The range of the calculator is from $\pm10^{-99}$ to $\pm9.999999999 \times 10^{99}$; when this range is exceeded during a calculation, 'NOTE 10' is displayed. Calculations which normally result in zero, such as subtracting a number from a number equal to itself, do not exceed the range.

OPERATING THE PRINTER

PAPER Refer to 'Loading Printer Paper' on Page 6.

NORMAL **TRACE** Refer to 'Printing Keyboard Operations' on Page 12.

LIST Lists programs stored in the memory; see 'An Example Program' on Page 40.

PRINT The print key is used to print both numerical values and messages. The form of a numerical printout is changed by the FIXED N and FLOAT N keys in the same way as the display is changed (see 'Positioning the Decimal Point', on Page 13).

To print numbers:

FIXED N (2) EXECUTE

PRINT (1)(2)(3) EXECUTE

123.00
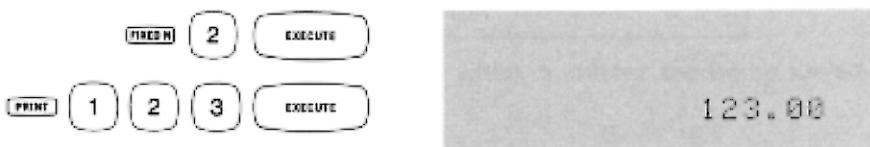
To print the result of a calculation:

PRINT (6)(+)(8)(/)(2) EXECUTE

10.00

To print the contents of a storage register:

$$\boxed{4}\;\boxed{5}\;\boxed{6}\;\boxed{\rightarrow}\;\boxed{A}\;\boxed{\text{EXECUTE}}$$

$$\boxed{\text{PRINT}}\;\boxed{A}\;\boxed{\text{EXECUTE}}$$

456.00

Similarly $\boxed{\text{PRINT}}\;\boxed{X}$ or $\boxed{\text{PRINT}}\;\boxed{R(\,)}\;\boxed{1}\;\boxed{5}$

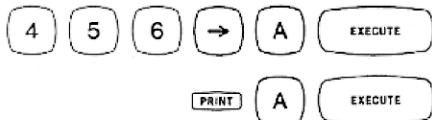To print an alphanumeric message: This requires the use of the 'quote' key (") to both start and end the message (the 'quote' symbol is not printed):

$$\boxed{\text{PRINT}}\;\boxed{"}\;\boxed{M}\;\boxed{E}\;\boxed{S}\;\boxed{S}\;\boxed{A}\;\boxed{G}\;\boxed{E}\;\boxed{\text{SPACE}}\;\boxed{N}\;\boxed{O}\;\boxed{.}\;\boxed{2}\;\boxed{"}\;\boxed{\text{EXECUTE}}$$

MESSAGE NO. 2
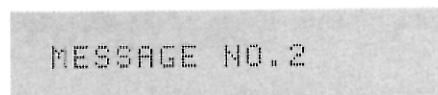
No more than sixteen characters (including spaces) can be printed on one line of a message; each line must be enclosed in quotes. When following the same PRINT instruction, lines must be separated by commas; for example:

$$\boxed{\text{PRINT}}\;\boxed{"}\;\text{-------}\;\boxed{"}\;\boxed{,}\;\boxed{"}\;\text{-------}\;\boxed{"}\;\boxed{\text{EXECUTE}}$$

prints two lines.

OPERATING THE PRINTER (Continued)

If messages and values are to be mixed, they must be separated by a comma:

$$\boxed{4}\ \boxed{5}\ \boxed{6}\ \boxed{\rightarrow}\ \boxed{A}\ \boxed{\text{EXECUTE}}$$
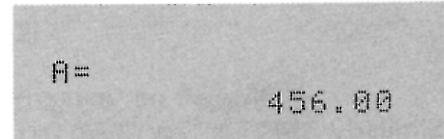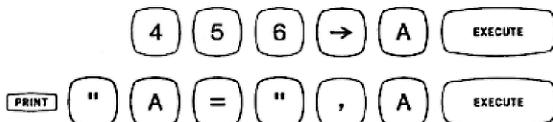
$$\boxed{\text{PRINT}}\ \boxed{"}\ \boxed{A}\ \boxed{=}\ \boxed{"}\ \boxed{,}\ \boxed{A}\ \boxed{\text{EXECUTE}}$$

```
A=
                    456.00
```

SPACE N Followed by number keys (any one of the set 0 through 15) causes the printer to space vertically (line-feed). The number key specifies the number of lines spaced.

$$\boxed{\text{PRINT}}\ \boxed{"}\ \boxed{D}\ \boxed{A}\ \boxed{Y}\ \boxed{S}\ \boxed{"}\ \boxed{\text{EXECUTE}}$$

$$\boxed{\text{SPACE N}}\ \boxed{2}\ \boxed{\text{EXECUTE}}$$

$$\boxed{\text{PRINT}}\ \boxed{4}\ \boxed{\text{EXECUTE}}$$

```
DAYS


                      4.00
```
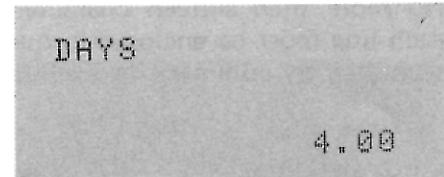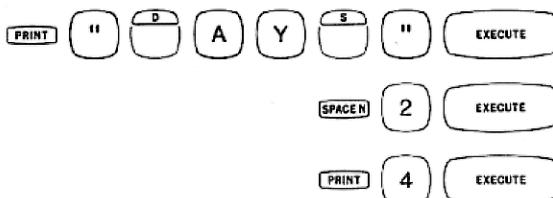
When used in a message, most keys result in the character printed being the same as the character on the key; following are the exceptions:

(SPACE) prints one blank character-space.

(GO TO) prints a̅

(R( )) prints ≡

(STOP) prints !

(ENTER EXP) prints †

The following keys either cannot be used in a message or they result in some meaningless character being printed:

1. All of the half-keys at the top of the keyboard and the four blank keys in the left-hand keyblock.

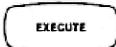2. These three large keys: (EXECUTE) (RUN PROGRAM) (STORE)

3. These keys in the right-hand keyblock: (JUMP) (END) (IF) (GO TO SUB) (FLAG N) (RETURN) (SET CLEAR FLAG N)

PROGRAMS

A program enables the calculator to automatically execute the keys necessary to solve a particular problem. First the program must be loaded into the calculator's memory; this 'teaches' the calculator which key sequences are required and the order in which they are to be executed. Once loaded, the calculator can 'remember' that program until a new one is loaded over it or until the calculator is switched off.

A program need not be keyed into the calculator more than once because a loaded program can be recorded on magnetic cards; recorded programs can then be loaded back into the memory any time in the future. Magnetic cards are a convenient way of permanently storing programs because they enable even long programs to be loaded back into the memory in seconds.

When the program has been loaded, you run it by initializing it in some way and then by pressing the RUN PROGRAM key to start it running. Most programs have one or more halts in them to enable you to key in any required data numbers.

PROGRAM USER-INSTRUCTIONS

The versatility of the calculator and the variety of programs makes it impractical to give here a precise set of instructions which will enable you to run all programs. User-instructions to load and run a particular program should be provided by whoever writes the program. Without

user-instructions, the user who knows nothing about programming would find it impossible to determine the steps necessary to run the program; even an experienced programmer might, in many cases, find it easier to write a new program rather than try to run an existing one without any user-instructions!

Following are some general guidelines to the type of information which a set of user-instructions should contain (these guidelines assume that the program has already been recorded on a magnetic card and that it is known to work properly):

1. What the program does.

2. How many magnetic program cards are to be loaded and how they are identified.

3. Where, and how, to start loading the program. . . usually this will be at the beginning of memory, but not always.

4. Where, and how, to start running the program. . . usually program execution will start at the beginning of the program, but, again, not always.

5. When to key in data numbers and when to press RUN PROGRAM.

6. How to interpret the display or printout and where to look for results.

7. Any other special information you may need (such as which ROM's should be installed).

8. A set of data numbers with known results which you can use to test-run the program and check that you are interpreting and following the instructions properly.

AN EXAMPLE PROGRAM

The purpose of this program is to show you the type of operation which you will have to perform to run programs. It is also used to demonstrate use of the magnetic cards.

Before the program can be demonstrated it must first be keyed into the calculator's memory; use the following key sequences to do this. As you will see, the program consists of lines, which are loaded one at a time; before pressing the STORE key at the end of each line, check the display to ensure that the line has been keyed correctly — if necessary the BACK, FORWARD and CLEAR keys can be used as before to correct a line before it is stored. If you need to correct a line after it has been stored, wait until you have stored all of the remaining lines. The incorrect line can then be recalled by pressing

CLEAR GO TO (Line Number) RECALL

Correct the line as before, and press STORE to re-store it (even though the corrected line may have been lengthened or shortened, it will not be necessary to move any subsequent stored lines to compensate for the correction; the calculator does this automatically).

To load the program; first initialize the calculator: ERASE

40

Load line 0:

(SPACE N) (2) (;) (0) (→) (B) (STORE)   `0: SPC 2;0→B⊢`
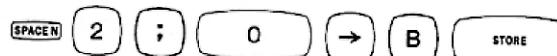
Notice that the line number (0) and the end-of-line symbol (⊢) were added when STORE was pressed.

Load line 1:

(ENTER) (") (N) (E) (X) (T) (SPACE) (N) (U) (M) (B) (E) (R) (") (,) (A) (STORE)

Load line 2:

(IF) (FLAG N) (1) (3) (=) (1) (;) (GO TO) (4) (STORE)

Load line 3:

(PRINT) (A) (;) (A) (+) (B) (→) (B) (;) (GO TO) (1) (STORE)
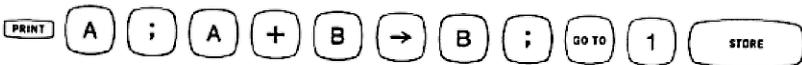
AN EXAMPLE PROGRAM (continued)

Load line 4:

⟨PRINT⟩ ⟨"⟩ ⟨T⟩ ⟨O⟩ ⟨T⟩ ⟨A⟩ ⟨L⟩ ⟨=⟩ ⟨"⟩ ⟨,⟩ ⟨B⟩ ⟨STORE⟩

Load line 5:

⟨END⟩ ⟨STORE⟩

The program is now loaded.

To check that the program has been loaded correctly print a 'listing' (shown opposite) and compare that to the keying sequence used:

⟨END⟩ ⟨LIST⟩

(the R-number at the end of the list will probably differ from that shown, because of different calculator configurations).

```
0:
SPC 2;0→B⊦
1:
ENT "NEXT NUMBER
",A⊦
2:
IF FLG 13=1;GTO
4⊦
3:
PRT A;A+B→B;GTO
1⊦
4:
PRT "TOTAL=",B⊦
5:
END ⊦
R414
```

The program just loaded enables you to total any list of numbers (as was done manually in Example 2, on Page 28); as each number is keyed, it is printed automatically; then, at the end of the program, the total is printed and identified. The program has been arranged so that the list of numbers can be any length.

Before running the program, position the decimal point so that you will obtain the desired printout. For example:

FIXED N (0) (EXECUTE)

Start the program:

(END) (RUN PROGRAM)   ⌐ NEXT NUMBER ⌐

(Using the same list of numbers as before, 9, 36, 25,) key in the first number from the list and press RUN PROGRAM. When that number has been printed, key in the next number and again press RUN PROGRAM; continue in the same way for all remaining numbers. When all numbers have been printed, press RUN PROGRAM, without pressing any other key, so that the program will print (and display) the final total. The program has now ended; it can be started again for a new list of numbers by pressing END RUN-PROGRAM and then keying the list as before. (If you cannot re-run the program, press STOP and CLEAR, and then press END RUN-PROGRAM.)

Here is the printout from the example program, using the list of numbers given above.

```
                    9
                   36
                   25
TOTAL=
                   70
```

MAGNETIC PROGRAM-CARDS

Figure 6 shows a magnetic card, which is used to permanently (or temporarily) store programs or data. The card has two sides, identified by the word 'SIDE', space being left for you to write any other identifying information. The sides are used independently, somewhat in the same way as the two sides of a tape are used in a standard tape-recorder.
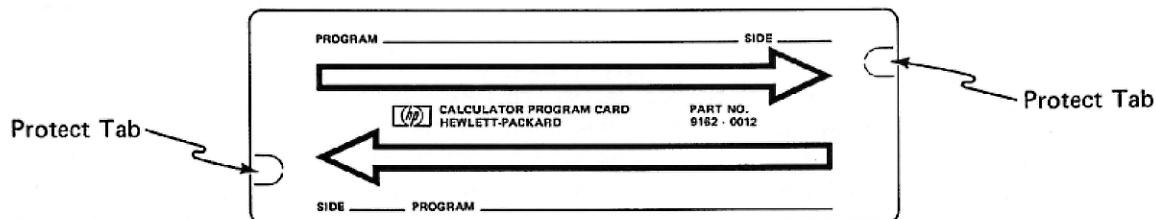
Figure 6. Magnetic Program Card

Once a recording has been made on a card-side, that card-side can be protected from erasure by tearing out the protect tab (see Figure 6). The recording on a protected card-side cannot be changed, so never protect a recording until you have ensured that the recording is correct — load the program back into memory and run it.
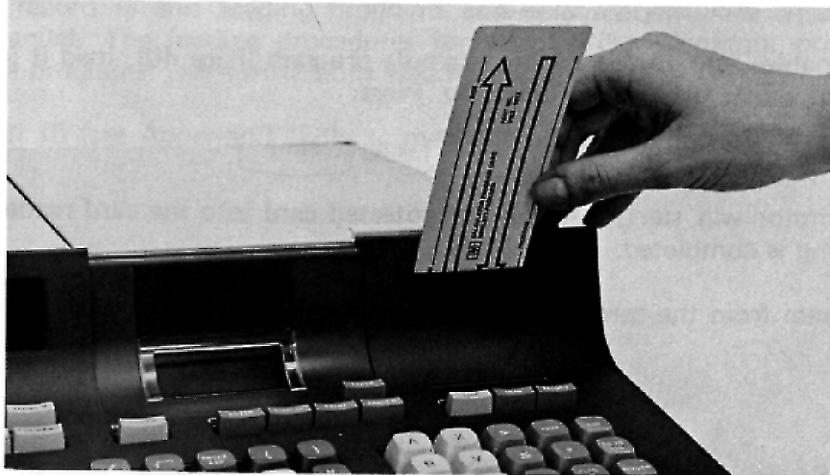
Figure 7. Inserting the Card

Figure 7 shows a card being inserted into the card reader, either for recording or for loading; the printed face faces the keyboard and the card-side to be used is the side with the arrow pointing down (notice that the card is leaned slightly towards the keyboard). Always start the card-reader motor (keying sequences given below) before inserting the card. When the card is about an inch into the slot it will be automatically pulled through the card reader and partially ejected from the lower slot. When the card stops moving, remove it from the slot.

MAGNETIC PROGRAM—CARDS (Continued)

This procedure enables you to record the example program (Page 40), load it into the memory, and run it, to ensure that it is loaded correctly. Press:

(END) (EXECUTE) (RECORD) (EXECUTE)

The card-reader motor will start; insert an unprotected card into the card reader; when the card stops, the recording is completed.

To load the program from the card into the memory, first clear the memory:

(ERASE)

Then press:

(END) (EXECUTE) (LOAD) (EXECUTE)
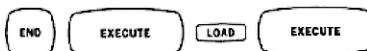
and insert the card (with the arrow on the card-side containing the program pointing down) into the card reader. When the card stops moving, the program is loaded and ready to run as before (press END RUN-PROGRAM).

The procedure given above is intended specifically for the example program. The same procedure will, however, apply to most other programs which start at line 0 and which do not require more than one card-side.

Other ways of recording and loading programs and data are included in the Operating and Programming Manual. The loading procedures for specific (pre-recorded) programs should be included in those programs' user-instructions (see Page 38).

Notes 12 through 15 (see Appendix) apply to magnetic card operations.

NOTES

Part 2

PROGRAMMING THE CALCULATOR

PROGRAM WRITING

Writing programs for the 9820A Calculator is, in general, easier than you might expect. It consists of writing 'statements' (such as the arithmetic expressions, instructions to the printer, data storage instructions, etc., used earlier in this book) and then combining them with other special (programming) statements, in some logical order, to form a program.

Program writing can be considered as having three main steps:

1.   Define the objectives of the program — the problem to be solved, how the results are to be presented, etc.

2.   Decide what operations are necessary, and the order in which they are to be performed, in order to achieve the objectives.

3.   Write precise instructions, in a manner which the calculator can understand, instructing it to perform the required operations.

PROGRAM WRITING (Continued)

To illustrate the three steps described above, consider the program from Page 40.

1. Define program objectives:
The program is to be used to total any list of numbers; each number in the list is to be printed and then the final total is to be printed and identified.

2. Decide necessary operations:
This step is best achieved by means of a 'flowchart' — a diagramatic representation of the operations to be performed. Figure 8 is the flowchart of the example program; the arrows indicate the order of operations, the ovals represent user operations, and the rectangles calculator operations. The diamond represents a question which the calculator must ask in order to decide which way to 'branch'. The calculator itself makes this decision, whether to branch back around the 'loop', or whether to branch out of the loop and finish the program; in this case the decision is based on whether or not a new number was keyed in. Notice, in particular, that there is considerable similarity between the operations shown on the flowchart and the operations which you might perform in order to total a list of numbers on paper.

NOTE

It is perhaps only fair to point out to the novice programmer that a final flowchart (such as that shown) usually results only after several attempts to produce it.
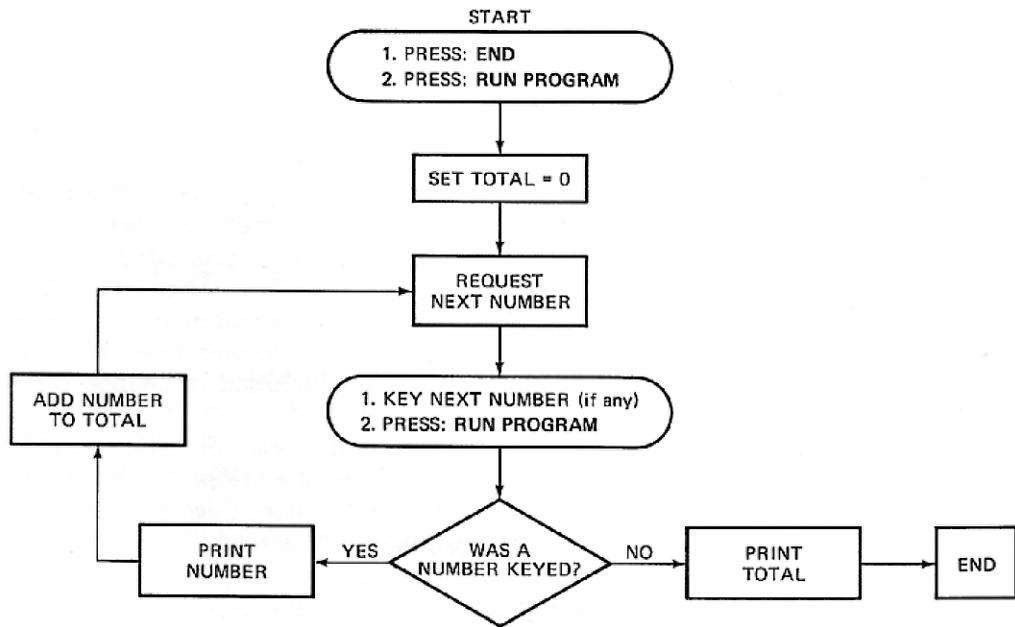
Figure 8. Example Flowchart

PROGRAM WRITING (Continued)

3. Write the program:

```
0:
SPC 2;0→B⊢
1:
ENT "NEXT NUMBER
";A⊢
2:
IF FLG 13=1;GTO
4⊢
3:
PRT A;A+B→B;GTO
1⊢
4:
PRT "TOTAL=";B⊢
5:
END ⊢
```

From the program listing it can be seen that a program consists of numbered lines (of varying length); in this program the lines are numbered 0 through 5. (One 'program line' may occupy more than one 'printed line' because the printer can print no more than sixteen characters on a line.) A line's number is assigned automatically, in numerical sequence, when the line is stored into the calculator's memory. When the loaded program is run, the lines will be executed in the same numerical order, unless special instructions (see 'The Go To Statement' on Page 59) are included to change that order.

Each line consists of one or more statements; the statements on any one line are separated from each other by means of the semi-colon. Several of the statements in the sample program will already be quite familiar to you while others will be new — the following discussion concentrates mostly on the types of statement which are new. While reading this material, correlate the operations shown in the flowchart with the statements of the program.

Line 0 contains two familiar statements:

SPC 2 (not shown in the flowchart) advances the printer paper two lines; this separates the printout for this program from any previous printout.

$0 \rightarrow B$ stores zero into register B so that the initial total for the list is zero.

Line 1 contains an 'enter' statement (a request for data); this halts the program to enable the user to key in a number and then press RUN PROGRAM to continue running the program. This particular 'enter' statement consists of two parts, separated, as is proper, by the comma: the characters in quotation marks (NEXT NUMBER) will be displayed when the program halts; the part following the comma is a register name (in this case A) — the keyed number will be automatically stored in that register as soon as program execution resumes.

Line 2 contains an 'if' statement, which enables the calculator to ask the question shown in the diamond in the flowchart. A more complete explanation of the 'if' is given later in this book; for the present, it is sufficient to know that if a new number has just been keyed (YES in the

PROGRAM WRITING (Continued)

flowchart) then the program will ignore the rest of line 2 and go on to the next line, line 3; if no new number was keyed (NO in the flowchart) then the calculator will execute the GTO 4 (go to line 4) so that program execution continues at line 4 (instead of the next sequential line, line 3).

Line 3 contains three statements; these (1) cause the value currently stored in A (the last keyed number) to be printed; (2) add the value in A to the current total of the list, stored in the B register; and (3) 'unconditionally' branch the program back to line 1. The branch is unconditional because there is no 'if' statement preceding the GTO 1; the calculator is not able to exercise any option in this case, it must make the branch and continue program execution at line 1. The program will continue to 'branch around the loop' (do line 1, ignore the 'go to' in line 2, do line 3, do line 1, and so on) until such time as the user reaches the end of the list and so presses RUN PROGRAM without first keying any number. When this occurs, then the 'conditional' branch (GTO preceded by an 'if' statement) in line 2 will be executed, so that program execution then resumes at line 4.

Line 4 is an ordinary 'print' statement, enabling the final total, in register B, to be identified (TOTAL =) and printed. Again notice the use of the comma to separate the two parts of a single statement.

Line 5 contains the 'end' statement to indicate that the program is finished and to stop program execution.

THE PROGRAM LINE

Even though the lines of a program are stored in the same memory as is data (see The Data Memory, Page 22), the length of individual lines bears no relationship to the length of a register; the calculator simply uses however many registers are necessary to accommodate a particular line.

The length of a line is determined by the programmer and depends upon the requirements of his program; however, the length is limited by machine requirements, in the same way that an individual expression is limited (see Exceeding the Length of the Display, Page 20). NOTE 09 appears, either before or after STORE is pressed, if the line is too long. When NOTE 09 occurs, press CLEAR and key in a completely new (shortened) line; do not attempt to shorten an existing line once it has been stored and NOTE 09 has appeared — it is easier to rewrite the line.

Line numbers are automatically assigned, by the calculator, in strict numerical sequence, beginning with line 0. You do not have to key the line number (in fact, you cannot) when you are keying in the line, but you must know what numbers will be assigned if there are any 'go to' statements (see Page 59) in your program.

The line numbers are not strictly a part of the program because they will automatically change if the program is moved to a different location in memory. For example, suppose a program (No. 1) is a ten-line program (lines 0 through 9) and is already stored in the memory. If a second program (No. 2) is now loaded below program No. 1, then No. 2's first line will be line

THE PROGRAM LINE (Continued)

10, whereas, if No. 2 had been the only program in the memory, then its first line would have been line 0. (Any 'go to' statements must be corrected, by the programmer, to reflect any such line number changes.)

A line can have one or more statements, separated by semi-colons. The actual number of statements on any one line is generally not significant, it being more important to have the statements in the correct order rather than on a particular line. Position of a statement does become significant where a line contains an 'if' statement (Page 60) or where a branch is to be made. In the former case, those statements which are to be conditionally executed must be on the same line as the 'if' statement and must come after the 'if'. In the latter case, a branch is always made to the beginning of a line; therefore, the first statement to be executed after a branch must be the first statement of the line to which the branch is made.

It is recommended that you do not put too many statements on one line because a short line is easier to change (once stored) than is a long line.

THE DATA ENTRY STATEMENT

`ENTER`

Enter statements are used to halt the program so that the user can key in data (the program shown on Page 52 contains a typical 'enter' statement). The simplest statement contains only a register name; the program then halts with that name displayed. The data keyed during the halt is stored, into the register designated, when RUN PROGRAM is subsequently pressed. For example:

$$ENT\ A;$$

results in the keyed data being stored in register A.

An enter statement may contain more than one register name (these must be separated by commas); the program will then halt for each register in turn. For example:

$$ENT\ A,R13,X;$$

is the equivalent of the three separate statements

$$ENT\ A;ENT\ R13;ENT\ X;$$

A 'label' (followed by a comma) may precede the register name; in this case the label will be displayed, instead of the name, when the halt occurs:

$$ENT\ "A=?",A;$$

displays A=? and stores the subsequent data entry into register A.

AN IMPORTANT NOTE

When the calculator halts for a data entry, it remains in the 'enter' mode until the requirements for the entry are properly satisfied. Failure to complete the entry will cause NOTE 01 to appear if any subsequent program activity is attempted, even though certain keyboard activities, such as executing an arithmetic expression, can be performed normally. To avoid confusion (which may not occur until several hours later!), always complete a data entry; any one of the following ways may be used:

1. Key a number and press RUN PROGRAM.

2. Press RUN PROGRAM.

3. If you do not wish to resume running the program, press STOP.

Remember: always satisfy the requirements of an 'enter' statement.

THE 'GO TO' STATEMENT

Program lines are normally executed in numerical sequence; however, some statements cause the sequence of execution to be changed. This is known as 'branching'; instead of the program going to the next sequential line, it branches to some other (specified) line and continues program execution there. The branching statement consists of the GO TO key followed by numerical keys to specify the line number, for example:

$$\text{GTO 4; or GTO 15;}$$

(Other types of branching statements are described in the Operating and Programming manual.) The example program on Page 52 contains two 'go to' statements.

In a program, the 'go to' statement causes program execution to continue with the line whose number is specified.

From the keyboard and followed by RUN PROGRAM, the 'go to' statement causes program execution to start at the line whose number is specified.

From the keyboard and followed by EXECUTE, the 'go to' statement causes the calculator to go to the line specified but not to start program execution. Any subsequent activity then depends upon the next key pressed.

A line number is valid only if a currently stored program has a line identified by that number, or if it is the next higher number after the number identifying the last stored line. All other numbers are non-valid and, if used in a 'go to' statement, will cause NOTE 08 to be displayed.

THE 'IF' STATEMENT

The 'if' statement enables the calculator to decide whether or not to execute the succeeding statement(s) on the same line as that 'if' statement. The general form of the 'if' statement is:

$$IF \text{ [condition]} \text{;}$$

For example:

$$IF \quad A=3 \text{;}$$

tells the calculator to check the truth of the condition "the value in the A register is equal to 3" and to act accordingly. If the condition is true (YES), the rest of the current line is executed; if false (NO), the rest of the line is ignored and the next line is executed.

The 'conditions' all use one of the following keys:

$\boxed{>}$ "—greater than—"

$\boxed{\leq}$ "—less than or equal to—"

$\boxed{=}$ "—equal to—"

$\boxed{\neq}$ "—not equal to—"

thus, the general form of the 'if' statement can be written as:

$$\mathrm{IF} \begin{bmatrix} \text{thing} \begin{Bmatrix} \text{one of} \\ > \\ < \\ = \\ \neq \end{Bmatrix} \text{thing} \end{bmatrix} ;$$

where the 'thing' parts of the statement can be values, register names, arithmetic expressions or flags (explained later). Typical 'if' statements might be:

```
IF A=3;
IF R14>(4A+2)/6;
```

When the calculator makes the check, it substitutes the appropriate value for any register name or for any arithmetic expression, just as it does in other types of statement.

The most common use of the 'if' is to make conditional branches (that is, to conditionally execute a 'go to' statement). Consider this line, taken from an imaginary program:

```
10: IF 4>Y;Y+1→Y;GTO 7⊢
```

As long as Y is less than 4, the condition is true; the program then adds 1 to Y and branches back to line 7. The program will continue to 'loop', executing lines 7, 8, 9 and 10 until the test is made when Y has a value of 4; now the condition is no longer true so the program goes on to line 11. In this example, line 10 also acts as a counter and counts the number of times the branch (GTO 7) will be made; if Y is initially equal to zero, then the branch will be made four times (so that lines 7, 8 and 9 will be executed five times).

THE STOP AND END STATEMENTS

STOP   The STOP key, used as a statement in a program or pressed while a program is running, stops program execution. STOP should be used only to 'abort' a program (in the sense that you no longer wish to run the program, or that you wish to stop it and start again at the beginning).

END   END serves the dual purpose of stopping program execution and of initializing the calculator ready to start program execution at line 0.

THE FLAGS

FLAG N   **SET CLEAR FLAG N**

The calculator has sixteen 'flags'; these are selected by the FLAG N key, followed by numeric keys to designate one of the set: 0 through 15. For example:

$$\boxed{\text{FLAG N}}\;\boxed{4}\quad \text{(mnemonic } \mathsf{FLG\ 4}\text{)}$$

selects flag 4. Flags are used mostly as part of an 'if' statement to enable the user to define some special condition.

The basic concept of 'flagging', although quite simple, is sometimes difficult to grasp; an analogy may be helpful: imagine, as the driver of a car, that you are given these instructions; "Half a

mile down the road is a flagman; if his flag is raised, turn right; if his flag is lowered, turn left."
The programmer's 'flag' is just that — a flag! Notice that the condition — a raised or lowered
flag — does not specify any particular reason for the state of the flag; in this case, the flag could
be raised for road repairs, for floods, or for stock on the road. The flags in the calculator are
somewhat like this; you select any reason you wish for raising or lowering them.

The calculator terminology used to describe flags is quite simple: If you raise a flag, you SET it;
a set flag is considered to have the value 1. If you lower a flag, you CLEAR it; a cleared flag is
considered to have the value 0.

Flags are set and cleared by means of the SET/CLEAR FLAG N key; press once to 'set', twice
to 'clear':

SET/CLEAR FLAG N (1) (2)   (mnemonic SFG 12)

sets flag 12;

SET/CLEAR FLAG N SET/CLEAR FLAG N (7)   (mnemonic CFG 7)

clears flag 7. Once set, a flag remains set until it is deliberately cleared; however, all flags are
automatically cleared at turn-on, or when MEMORY ERASE is pressed, or when an END
statement is executed.

The following program lines illustrate a typical use of the flag:

```
4: SFG 3⊢
8: IF FLG 3=1;CFG 3;GTO 5⊢
```

THE FLAGS (Continued)

In this case the purpose is to execute lines 5, 6 and 7 twice and then to go on to line 9. A flag is set in line 4 so that· the first time line 8 is reached the 'if' condition is true; therefore the branch back to line 5 is made. Before the branch is made, the flag is cleared so that, the second time line 8 is reached, the condition is not true and the program will go on to line 9.

As long as no program is being executed, the state of any flag can be determined by using this key sequence:

$$\boxed{\text{FLAG N}} \quad \boxed{\text{Number}} \quad \boxed{\text{EXECUTE}}$$
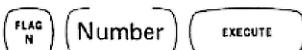
The state (value) of the flag will be displayed; a 1 for a set flag or a zero for a cleared flag. (The test will not change the state of any flag.)

In addition to their normal use, flags 0 and 13 also have a special purpose.

Flag 0 can be set from the keyboard while a program is actually running, by pressing the SET/CLEAR FLAG N key.

Flag 13 — set automatically if the program halts for an 'enter' statement and RUN PROGRAM is then pressed without any data being keyed (this feature is used to advantage in the example program shown on Page 40).

OPERATING PROGRAMS - A SUMMARY

1. To clear the memory, press:

    [ERASE]

2. Before loading, set the calculator to the beginning of memory, press:

    (END) (EXECUTE)

3. Load the program by keying one line at a time, in numerical sequence; at the end of each line press STORE. Statements can be corrected by means of the BACK and FORWARD keys.

4. To obtain a printout of the program, press:

    (END) [LIST]

5. To correct a line after it has been stored, press:

    (GO TO) (Line Number) [RECALL]

Then correct the program by means of the BACK and FORWARD keys and re-store it. Even though the line length may have been changed it will not be necessary to change any subsequent lines to compensate; the calculator automatically moves the subsequent lines up or down in the memory.

OPERATING PROGRAMS — A SUMMARY (continued)

6. To start the program, press:

(END) (RUN PROGRAM)

7. To check a program, set the printer to the 'trace' mode; press:

[TRACE] (EXECUTE)

and then run the program. As each line is executed, the printer will print the line number and the numerical result (if any) of that line. This enables you to see where a numerical error occurred, or where, for some reason, lines were not executed in the expected order.

Appendix

THE DIAGNOSTIC NOTES

The purpose of the diagnostic notes is described on Page 9.

In order to keep the following list reasonably short, it has been made 'one-way' only, in that it is intended to be used only when some note appears. The list contains guidelines to enable you to determine specifically what caused that note to be displayed at that time. The list does not contain sufficiently exhaustive information to enable you to predict in all cases which note will be displayed in any given set of circumstances.

The explanation of items marked with the symbol '†' is beyond the scope of this book; refer to the Operating and Programming manual for full details.

NOTE 01

In view of the preceding keys, the last key pressed does not make sense to the calculator; for example, a 'multiply' following the R( ) key. Note 01 is the most commonly seen note and generally occurs as soon as the 'wrong' key is pressed.

Appendix

NOTE 02

An attempt to execute an instruction which is followed by an improper value; for example, the FIX N key followed by a number larger than 9.

Taking a square root is a special case:

a. $\sqrt{-}$ causes NOTE 01 when 'minus' is pressed.

b. $\sqrt{(-4)}$ or $\sqrt{A}$ (where A contains a negative number) when executed cause NOTE 02 to appear.

NOTE 03

Statement has an extra left-hand parenthesis [(] or a missing right-hand parenthesis [)].

NOTE 04

Statement has an extra right-hand parenthesis [)] or a missing left-hand parenthesis [(].

NOTE 05

a. Attempt to use a non-existent, or a non-available, R-register as a value in an expression.

b. Attempt to designate a flag other than one of Flags 0 through 15.

Appendix

NOTE 06

a.   Attempt to store into a non-existent, or a non-available, R-register.

b.   Attempt to enter a number whose exponent has an absolute value greater than 99.

NOTE 07(†)

Attempt to execute a RET not preceded by a matching GSB.

NOTE 08

Attempt to execute a GTO followed by a non-valid line number or label(†). Also applies to GSB(†) and JMP(†).

NOTE 09

a.   Writing, or executing, or storing too long an expression or program line.

b.   Nesting subroutines(†) too deeply.

NOTE 10

An intermediate or final result of a calculation exceeded the range of the calculator.

Appendix

NOTE 11

a. Pressing any half-key in the three left-hand keyblocks when:
   1) It is not part of a quote field; e.g. PRT ". . . ." and
   2) The key is not defined by some plug-in ROM.

b. Attempt to execute an 'enter' statement from the keyboard instead of in a program.

NOTE 12

a. Storing a program line [or loading a program or data(†) from a magnetic card] and exceeding the memory.

b. (†)No GTO or GSB preceding LOD when loading a program (from a magnetic card) under the control of the existing program.

NOTE 13

Attempt to record on a protected magnetic card.

NOTE 14 (†)

An additional card side is required when recording on, or loading from, a magnetic card. Press EXECUTE and insert the next card-side.

Appendix

NOTE 15(†)

Appearing after a program has been loaded from a magnetic card, indicates that the calculator does not have the same ROM's installed (in the same slots) as it did when the card (or cards) was recorded. This will not affect the running of the program as long as the particular ROM's required for that program are installed in the same slots (press CLEAR and run the program in the normal way).

Recordings made when no ROM'S are installed, do not result in NOTE 15 when they are loaded into calculators which do have ROM's installed.

NOTE 16

Attempt to use the printer when there is no paper in the printer. To continue using the calculator without printer paper: If the 'print' instruction came from the keyboard, press CLEAR; if from the program press STOP RUN-PROGRAM.

Appendix

ANOTHER EXAMPLE PROGRAM - N FACTORIAL

This program calculates N factorial (N!) for any positive integer value of N from zero to 69 (70! exceeds the range of the calculator).

$$N! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot N$$
$$6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$$
$$0! = 1 \text{ (by definition)}$$

Load the program at the beginning of memory. To run the program, press END RUN-PROGRAM; then enter a value for N and press RUN PROGRAM. The program automatically prints the values of N and N!, as shown below.

```
N=
                    6
N!=
                  720
```

```
0:
FXD 0;SPC 2⊢
1:
ENT "NEW N",A⊢
2:
1→B;1→C⊢
3:
IF A=0;GTO 6⊢
4:
IF C=A;GTO 6⊢
5:
C+1→C;BC→B;GTO 4
⊢
6:
PRT "N=",A,"N!="
,B⊢
7:
END ⊢
```

Appendix

START

1. PRESS: END
2. PRESS: RUN PROGRAM

REQUEST NEXT NUMBER

1. KEY NUMBER (N)
2. PRESS: RUN PROGRAM

1. STORE N → A
2. PARTIAL N! = 1 → B
3. MULTIPLIER = 1 → C

A = Current value for N
B = Partial result of N! (1, then 1·2, then 1·2·3, etc.)
C = Current multiplier term (1, then 2, then 3, etc.)

A = 0?   YES →   1. PRINT N (A)
                 2. PRINT N! (B)   →   END

NO

C = A?   YES

NO

1. NEXT MULTIPLIER   C + 1 → C
2. NEXT PARTIAL N!   BC → B

MEMORY
ERASE
145

EDIT

DECIMAL
FIXED N   FLOAT N
22   23

NORMAL   TRACE   DELETE   INSERT   RECALL   BACK   FORWARD
20   21   141   142   140   144   143

① ② ③

PAPER

DISPLAY
CLEAR
03

INPUT / OUTPUT
ENTER   DISPLAY   PRINT   SPACE N
24   25   27   26

PROGRAM
LIST   LOAD   RECORD
04   05   30

①		②		③	
S 44	10	D Ln EXPO 104 105	I ▼ 111 112	N 116	S 123
% 45	11	E 106	J SIN ARS 113	O 117	T 124
& 46	16	F ↑ 107	K COS ARC 114	P 120	U 125
' 47	17	G π 110	L TAN ART 115	Q 121	V 126
? 77	SPACE 40	H INT ABS	M TAB	R 122	W 127

/ 67   √ 134   ENTER EXP 136   (50   ) 51

* 52   7 67   8 70   9 71   " 72

− 55   4 64   5 65   6 66   , 54

+ 53   1 61   2 62   3 63   ; 73

0 60   · 56   EXECUTE 135

A 101   X 130   > 76   JUMP 31   END 32

B 102   Y 131   ≤ 74   IF 34   GO TO SUB 35

C 103   Z 132   = 75   FLAG N 36   RETURN 33

→ 137   R( ) 72   ≠ 43   SET CLEAR FLAG N 37   GO TO 100

RUN PROGRAM 133   STOP 41   STORE 02

The 9820A Keyboard

HEWLETT-PACKARD SALES AND SERVICE OFFICES

To obtain servicing information and order replace-
ment parts, contact the nearest Hewlett-Packard
Sales and Service Office in HP Catalog, or contact
the nearest regional office.

IN THE UNITED STATES

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

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

ILLINOIS
5500 Howard Street
Skokie 60076
(312) 677-0400

NEW JERSEY
W. 120 Century Road
Paramus 07652
(201) 265-5000

IN CANADA

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

IN EUROPE

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

INTERCONTINENTAL SALES REGION

Hewlett-Packard Company
3200 Hillview Avenue
Palo Alto, California 94304
(415) 326-7000