MODEL

10

hp HEWLETT-PACKARD 9810A CALCULATOR

OPERATING and PROGRAMMING

OPERATING AND PROGRAMMING

Computer
Museum

HEWLETT-PACKARD 9810A

Copyright Hewlett-Packard Company 1971

TABLE OF CONTENTS

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

KEY INDEX

A | F | K

B | G | L

C | H | M

D | I | N

E | J | O

Defined by the
plug-in ROM's (Page 3-1)

STATUS
3-9

π	$\frac{1}{x}$	x^2
4-11	4-20	4-20
b	int x	↑ ROLL
4-28	4-20	4-12
a	IN-DIRECT	↓
4-28		4-12
$y\to( )$	$y\overset{\rightarrow}{\leftarrow}( )$	$x\rightleftarrows y$
		4-12
$x\to( )$	$x\leftarrow( )$	↑
		4-12

4-25, -30,
-33, -34

DECIMAL

FLOAT	FIX ( )
●	●

— 4-3 —

MODE

RUN	PRGM
●	●

— 4-3, 5-12 —

KEY LOG	LIST
●	6-9

— 6-8 —

PROGRAM

LOAD	RECORD
INSERT CARD	
●	

— 5-28 —

PAPER
6-10

$\sqrt{x}$	CHG SIGN	ENTER EXP	CLEAR x
4-20	4-6	4-6	4-5

÷	7	8	9

×	4	5	6

−	1	2	3

+	0	·

4-14, -30, -34   4-5   4-6

CLEAR
4-6

CONTINUE
5-14

SET FLAG	IF FLAG	END
5-39	5-35	5 14

LABEL	IF $x<y$	PAUSE
5-13	5-35	5-15

FMT	IF $x=y$	STOP
5-16	5-35	5-14

PRINT SPACE	IF $x>y$	BACK STEP
5-16, 6-7	5-35	5-16

SUB RETURN	GO TO	STEP PRGM
5-45	5-12	5 15

Refer to the page number(s) shown for a description of the individual key functions.

MANUAL SUMMARY

Chapter 1: General Information.
Information for the owner; includes a list of optional equipment, turn-on instructions and a procedure to run an exerciser program.

Chapter 2: Simplified Operation.
Operating information selected especially for the user who wishes to make only simple calculations or to run pre-recorded programs.

Chapter 3: Operating Characteristics.
A 'hands-on' introduction to the calculator high-lighting its important features.

Chapter 4: Keyboard.
Describes making calculations from the keyboard without reference to programs (the program-writer will also need this information).

Chapter 5: Programming the Model 10.
Complete programming information: describes keys and features used exclusively for programming; how to load and run programs; magnetic card operations; etc. The first part of this chapter, 'Introduction to Programming', is intended for the novice programmer.

Chapter 6: Option 004 Printer.
Describes operation and use of the optional printer.

NOTE

The additional information required to operate a calculator with Option 001 is included on Page 4-24; similar information for Options 002 and 003 is included on Page 5-8.

The Hewlett-Packard Model 9810A Calculator is the central unit of the 9800 Calculating System; the system includes the basic calculator (the 9810A) and numerous optional plug-in and peripheral devices. A system can be built to suit your needs and can be added to, or changed, at any time.

The calculator operates either as a stand-alone unit or as the controller in a 9800 System. This manual describes the calculator as a stand-alone unit; separate manuals are supplied with each of the plug-in and peripheral equipments, detailing their use in the system and the part played by the calculator in controlling them.

The basic Model 9810A Calculator is shown in the photograph on the title page of this manual; its keyboard is shown on Page vi. The half-keys in the left-hand block of the keyboard enable the basic calculator to be tailored to your requirements. Operation of these keys is uniquely defined by each of various optional 'read-only-memories' (ROM's) which plug into the basic calculator; without the ROM's, the half-keys serve no function.

The basic machine is both a manually operated, and a programmable, desk-top calculator. It can efficiently solve both simple and complex problems in a wide range of disciplines, including business, statistics, education, the sciences, engineering, and so on. Addition of the optional ROM's enables the calculator to further specialize in these disciplines.

The basic calculator is fully programmable; its computer-like memories enable both program steps and data to be stored and manipulated. The standard program memory can store up to 500 program steps, and the standard data memory up to 51 data numbers. Programming is simple because the various keyboard operations become the program instructions; no special language need be learned. Standard programming features include separate data and program memories, conditional and unconditional branching, direct and indirect data storage and register arithmetic, relocatable programs, subroutines, and the ability to automatically load magnetic cards containing either program steps or data. An additional standard feature is program editing, which enables programs to be easily debugged and corrected, even though they are already stored in the memory.

INTRODUCTORY DESCRIPTION

CAUTION

PLEASE DO NOT APPLY OPERATING POWER TO YOUR CALCULATOR UNTIL YOU HAVE READ THE TURN-ON INSTRUCTIONS ON PAGE 1-7.

In addition to the ROM's and the peripheral devices previously mentioned, there are other products which include a printer and additions to the program and data-storage memories (these must be installed by Hewlett–Packard personnel). A complete list of optional devices is given later in this chapter.

SUPPLIED ACCESSORIES

The standard accessories supplied with each basic Model 9810A Calculator are listed in Table 1-1.

Table 1-1. Standard Accessories Supplied

DESCRIPTION	QUANTITY	-hp- PART NUMBER
Operating and Programming Manual	2	09810-90000
Program Pad	1	09810-90016
Math Pac (programs)	1	09810-70000
Exerciser Program (envelope containing 2 pre-recorded magnetic cards)	1	09810-90021
Magnetic Program Card with envelope	10	*See note below.
Power Cord	1	8120-1378
Dust Cover	1	4040-0894
Spare Fuses - all 230V Normal Blow ¼ in. dia. x 1¼ in. lg.		
1-amp	1	2110-0001
2-amp	1	2110-0002
6-amp	1	2110-0056
(If Option 004 Printer Installed) Roll of Printer Paper	3	See Chapter 6

*Additional quantities of magnetic cards may be purchased as follows:

    Magnetic program cards:
        Package of 10, -hp- Part No. 5060-9152.

Prices are available, upon request, from any HP Sales and Service Office (addresses are at the back of this manual).

Program pacs containing programmed solutions to practical problems in a wide range of disciplines are available for purchase. The Math pac is shipped with each calculator; for a complete list of other program pacs and for price information, contact any HP Sales Office.

PROGRAM PACS

The 'KEYBOARD' is a periodic magazine containing general information about Hewlett-Packard calculators. It includes articles and programs written by calculator users; descriptions of new equipment and of new program pacs; programming tips; and other articles of interest to calculator users.

KEYBOARD MAGAZINE

Service contracts are available for the calculator, options and other devices in your 9800 System. For further information contact any HP Sales Office.

SERVICE CONTRACTS

Additions to the basic calculator are classified into two groups: options which require that the calculator be modified and which must, therefore, be installed by qualified HP personnel, and products, such as read-only-memories (ROM's), which can be easily plugged into the calculator, and removed, by the user. A third group of additions, the peripheral devices, are additions to the system, rather than to the basic calculator. The items listed below are those available at the time this manual was printed (the printing date is on the back cover); other items will be available in the future.

OPTIONAL PRODUCTS

NOTE

In the basic calculator:
The program memory contains a total of 500 program steps.
The data-storage memory consists of 51 registers.
The left-hand block of keys is not defined.

---

1. OPTIONS
   (Operation of these options is fully described elsewhere in this manual).

   Option 001: Total of 111 data-storage registers (product number -hp- 11216A).

   Option 002: Total of 1012 program steps (product number -hp- 11217A).

   Option 003:Total of 2036 program steps (product number -hp- 11218A).

   Option 004: Printer (product number -hp- 11219A).

   (Options 002 and 003 are mutually exclusive.)

When an option is installed the calculator is referred to as a "Model 10 Calculator with Option(s) X X X, etc." If you order an option after purchasing your calculator, please quote the product number, instead of the Option number, on your order.

2. PLUG-IN (ROM's)
(Operation is described in the separate manuals supplied with each ROM.)

Some of the ROM's serve to uniquely define the half-keys in the left-hand block of the keyboard. Each of these ROM's has an overlay which can be placed over the half-keys to identify their functions when that ROM is installed. The keys, which protrude through the overlay, are then used in the same way as the other keys on the calculator. Other ROM's serve special functions, usually in conjunction with some other device.

ROM's can be plugged into, or removed from, the calculator in seconds; no tools are required.

MATHEMATICS ROM — -hp- 11210A:
Adds twenty-six mathematical functions and further programming features to the basic calculator. These include (among others) logarithms and exponential functions, trigonometric functions (in degrees, radians or grads), coordinate transformation, vector arithmetic, manipulation of complex numbers, automatic rounding to any power, automatic scaling for X-Y plotting, and a user-definable function.

STATISTICS ROM — -hp- 11214A:
Adds single-keystroke statistical computations to your basic calculator. These include χ^2; t; linear, multi-linear, and parabolic regressions; random number generation; accumulation of sums, sums of products, and sums of squares, for up to five variables; maximum/minimum search; and a 'correct' key to remove erroneous data.

DEFINABLE FUNCTIONS ROM — -hp- 11213A:
Enables up to nine user-definable functions to be stored at one time; each function can then be executed by a single keystroke. Stored functions can be easily deleted or changed at will. A 'protect' feature saves functions from being accidentally erased or changed when other programs are entered into the calculator. This ROM also has three special editing keys, FIND, DELETE and INSERT, which greatly simplify editing programs (both the definable functions and the programs in the rest of the calculator's memory).

PRINTER ALPHA ROM — -hp- 11211A:
Enables the 11219A Printer (Option 004) to print alphabetic characters and messages and to insert mnemonics into program listings.

3. PERIPHERAL EQUIPMENTS:
(Operation of each peripheral is fully detailed in the separate manual supplied with each device.)

A 9800 System is built to any desired configuration simply by plugging the peripheral devices into the back of the calculator. The input/output connector, at the rear of the calculator, can accept up to four peripheral devices at one time (see Figure 1-1).

MODEL 9860A MARKED CARD READER:
Allows data and program steps to be entered into the calculator from pencil-marked cards, bypassing the keyboard. The resultant time saving is highly desirable in a classroom or other place where many people may wish to use the calculator.

MODEL 9861A TYPEWRITER:
Facit (modified) Model 3841 Output Typewriter together with interface equipment, allows the calculator to control the typewriter functions and keys. Types formatted data, headings and messages in a variable field-width (limited only by platen size); also lists programs from the calculator's memory. The typewriter, equipped with a standard ECMA II keyboard, can be disconnected from the system and used as a stand-alone typewriter.

The inspection procedure enables you to confirm that your calculator is in the best possible condition upon receipt.

The calculator was carefully inspected both mechanically and electrically before it was shipped to you. It should, therefore, be free of any mars or scratches and in perfect electrical order (operational) upon receipt. You should carefully inspect your calculator for physical damage caused in transit and also check that the accessories listed in Table 1-1 are present. If there is any physical damage, file a claim with the carrier and notify HP (addresses of Sales and Service offices are given at the back of this manual).

To check the operation of the calculator, run the exerciser program (Page 1-8). Do not make this check until you have performed the turn-on procedure detailed on Page 1-7.

INSPECTION PROCEDURE

The Model 9810A Calculator has the following power requirements (refer to Figure 1-1).

LINE VOLTAGE: Nominally 115 and 230 volts ac and 'low-line' levels of each voltage. Dual switching enables any of the following four line voltages to be selected -- 230 or 200, 115 or 100 volts. At each switch setting, the calculator will operate with the line voltage at levels of within ±10% of the selected voltage. The operating range is, therefore, from 90 to 126.5 volts and from 180 to 253 volts.

POWER REQUIREMENTS

POWER REQUIREMENTS (continued)

LINE FREQUENCY: Nominally 50 Hertz and 60 Hertz; the calculator will operate within the range of 48 to 66 Hertz.

POWER CONSUMPTION: With no peripheral equipments attached, 150 voltamps maximum. There are three 'piggy-back' power terminals at the rear of the calculator to provide power to peripheral devices; no more than a total of 610 voltamps may be drawn from these three terminals (see Figure 1-1).

Input/Output Connectors for Peripheral Devices

Power Outlets for Peripheral Devices

Selects 'nominal' line voltage, 230 or 115V

Power Input

Selects 'normal' or 'low' line

Fan Filter
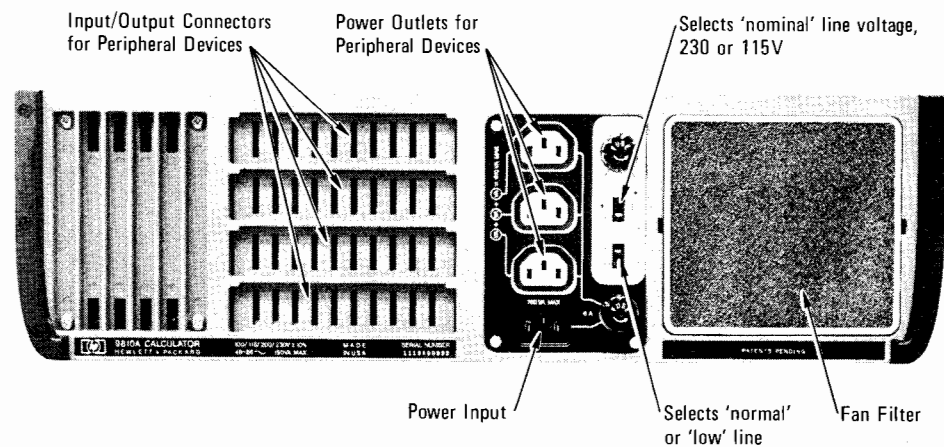
Figure 1-1 The Rear Panel

GROUNDING REQUIREMENTS

To protect operating personnel, the National Electrical Manufacturers' Association (NEMA) recommends that the calculator's keyboard and cabinet be grounded. The calculator is equipped with a three-conductor power cable which, when plugged into an appropriate power receptacle, grounds the cabinet and keyboard of the calculator. The round pin on the power cable's three-pronged connector is the ground connection.

FUSES

The calculator has two fuses located on the rear panel (see Figure 1-1); a 6-amp fuse and either a 1-amp fuse for 200V/230V operation, or a 2-amp fuse for 100/115V operation. Three spare fuses, a 6-amp, a 2-amp and a 1-amp, (listed in Table 1-1) are shipped with each calculator.

CAUTION

BEFORE CHANGING A FUSE, ENSURE THAT
THE CALCULATOR IS DISCONNECTED FROM
ANY POWER SOURCE.

To change a fuse, press on the fuse-holder and, at the same time, twist it in the direction of the arrow on the fuse-holder; withdraw the fuse and holder from the socket. Remove the fuse from the holder and substitute the replacement fuse. To re-install the fuse and holder in the calculator, reverse the removal procedure.

With the calculator disconnected from any ac power source, verify that the correct fuse is installed for the line voltage in your area. (Refer to the preceding paragraph for information regarding the fuses.)

TURN-ON PROCEDURE

Two slide switches, located on the rear panel (see Figure 1-1), must be set for the ac line voltage to be used. The switches are set by inserting a screwdriver into the slot on the switch and then by sliding the switch so that the white slot is opposite the line which indicates the required voltage.

a. Set the upper of the two switches to your (nominal) line voltage; in Figure 1-1 this switch is set to 115V.

b. Set the lower switch to the normal position (115V/230V); the switch in Figure 1-1 is set to this position.

NOTE

This switch should be set to the low position (100V/200V) only if the line voltage in your area proves to be too low; that is, if the calculator cannot operate properly when the switch is set to the normal position.

Switch the LINE OFF/ON switch, located on the right front of the calculator, to the OFF position. Connect the power cord to the power input connector (see Figure 1-1) at the rear of the calculator and plug the other end of the cord into a suitable ac power outlet.

Switch the LINE OFF/ON switch to the ON position; after a few seconds warm-up, the display will appear and several of the lights on the keyboard will come on indicating that the calculator is ready to operate.

**EXERCISER
PROGRAM**

The Exerciser Program tests the performance of the calculator. If the test runs properly, then the calculator can be assumed to be operating correctly. If the test cannot be run properly, and you are certain that there has been no operating error, then contact any HP Sales Office for assistance.

The steps of the test program will be loaded into the calculator from the two magnetic program cards listed in Table 1-1 (the instructions on the envelope are a condensed version of those given below).

The exerciser program serves several purposes. It may be used as an initial inspection check. It may be used to initially check any options, added at a later date, which increase program memory or data-storage memory. It may also be used at any time as a confidence check if you suspect that your calculator is malfunctioning.

This exerciser program is used to test the basic calculator (with or without Option 004) and Options 001, 002 and 003. A separate program for Option 004 (-hp- 11219A Printer) is given in Chapter 6 of this manual. The exerciser programs for any other devices are included in their separate manuals.

NOTE

When an option is installed, a decal is attached to the calculator identifying the particular option number. To determine which options are in your calculator, lift the flap on the top cover; the decals, if any, can be found in the shallow depression under the flap.
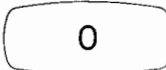
The halves of the two magnetic cards which contain the exerciser program are numbered in sequence: Side 1, Side 2, Side 3 and Side 4. The sides are required as follows:

    Basic Calculator (with or without Option 004) and
        Option 001 - Side 1.
    Option 002 - Sides 1 and 2.
    Option 003 - Sides 1, 2, 3 and 4.

NOTE

Before running the exerciser program unplug any peripheral devices or any plug-in ROM's. Switch the calculator off while doing this.

---

1. To set up the calculator to accept the program, press the keys in the order shown:

PRESS: [RUN] (STOP) (END) [FIX ( )] (0)

2. To load the program into the memory:

    PRESS: [LOAD] (the INSERT CARD lamp will light)

Insert the first magnetic card, with the printed side facing the keyboard and the arrow on Side 1 pointing downward, into the card reader's upper slot (the card reader is located above the RECORD and LOAD keys). The card will be automatically pulled through the card reader and partially ejected from the lower slot. When the card stops moving, remove it from the card reader.

NOTE

If the calculator has Option 002, do sub-step 2a (below).
If the calculator has Option 003, do sub-step 2b.
If the calculator has neither Option 002 nor Option 003, go directly to step 3.

---

a. If Option 002 is installed, load Side 2 of the magnetic card; press LOAD and insert the card, with the arrow on Side 2 pointing downward, into the upper slot of the card reader.

b. If Option 003 is installed, load the remaining three sides of the two magnetic cards (Sides 2, 3 and 4, in that order); for each side, press LOAD and insert the card, with the arrow on the required side pointing downward, into the upper slot of the card reader.

NOTE

During the rest of this procedure, if you press a wrong key, you should start again with step 3.

---

3. To start the program:

    PRESS: (END) (CONTINUE)

Regardless of the options installed in your calculator, the following display will appear.

temporary z	$0.$	— Total Program Steps
accumulator y	$500.$	
keyboard x	$51.$	— Total Data Registers

If you have the basic calculator (i.e. none of Options 001, 002 or 003), the display will be correct for your calculator; in this case, skip the next two steps (4 and 5) and go to step 6.

If your calculator has any of Options 001, 002, or 003, then the display must be changed so that it describes your calculator; in this case perform the appropriate parts of steps 4 and 5.

4. To correct the number of 'total program steps':

   a. If your calculator has neither Option 002 nor Option 003, but it does have Option 001, then perform only sub-step 5b and then do step 6.

   b. If your calculator has Option 002:

      PRESS: [2] [×]   (1000 appears in the display)

      and then do step 5.

   c. If your calculator has Option 003:

      PRESS: [4] [×]   (2000 appears in the display)

      and then do step 5.

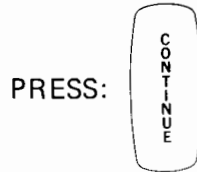5. To correct the number of 'total data registers':

   a. If your calculator does **NOT** have Option 001 (that is, if it has only 51 data-registers):

      PRESS: [5] [1]   (51 appears in the display)

   b. If your calculator has Option 001 (that is, it has 111 data-registers):

      PRESS: [1] [1] [1]   (111 appears in the display)

6. To run the program and observe the operation of the calculator:

PRESS: [CONTINUE]

Correct operation of the calculator is indicated by the following display (if the display is not correct, refer to step 7): The display will start to flash, with varying periods of darkness (from 1 to 5 seconds) between displays. Each display consists of three numbers:

temporary z	N
accumulator y	$5I.$ or $III.$
keyboard x	N

$N = 1$ thru N_{max}, where $N_{max} = 5$, 10, or 20.

The middle number, a constant, is the number of data registers in your calculator, 51 (basic) or 111 (Option 001). The other two numbers (N) start as 1 and then increment by 1, each time the display appears, until a maximum value (N_{max}) is reached. The value for N_{max} depends upon the total number of program steps in your calculator, thus:

OPTION	TOTAL PROGRAM STEPS	N_{max}
Basic	500	5
002	1000 (nominal)	10
003	2000 (nominal)	20

Once N_{max} is reached, the value for N again becomes 1 and the cycle is repeated. One complete cycle (1 to N_{max}) is sufficient to check operation of the calculator; however, the program will continue to run, with the display flashing until it is stopped.

To stop this program, hold the PAUSE key pressed until the display appears and remains fixed. To then continue the program, press CONTINUE; or, alternatively, to restart the program, stop it and then repeat the procedure starting at the beginning of step 3.

7. If the program cannot be run properly, then the calculator may not be operating correctly. On the other hand, the procedure for running the exerciser program is somewhat involved (especially for an operator

**EXERCISER
PROGRAM
(continued)**

who is not familiar with the calculator) so there is some possibility that an operating error may have occurred. To determine which:

a. Check once again which of Options 001, 002 and 003 (if any) are installed in your calculator.

b. Carefully repeat the entire procedure from the beginning of step 1; pay particular attention to those steps where a choice must be made (the choice always being dependent upon which options are installed in your calculator).

c. If the program still cannot be run properly and you are sure that there has been no operating error, then it may be assumed that the calculator is inoperative. In this case, please contact the nearest HP Sales and Service Office for assistance.

**CLEANING THE
CALCULATOR
AND FAN-FILTER**

Clean the calculator with a soft cloth, dampened in clean water or in water containing a soft soap or mild detergent. Do not use an excessively wet cloth or allow water to penetrate inside the calculator. In particular, do not use any abrasive materials, especially on the display window.

The fan-filter (located on the rear of the calculator) should normally be cleaned about every three months. Clean it by holding it under a running water-faucet or by washing it in warm soapy water and then rinsing in clean water. Dry the filter thoroughly before re-installing it.

The filter can be easily removed by using a small blunt instrument such as a screwdriver, a paper-knife or a nail file. Insert the instrument into one of the slots located on either side of the filter; the filter can then be snapped out by applying pressure toward the center of the filter and, at the same time, toward the back of the calculator. To replace the filter, snap in first one side and then the other.

This chapter is written especially for the individual who wishes to perform only simple arithmetic calculations and to run programs previously recorded on magnetic cards. Essentially, the calculator is assumed to be little more than a 10-key adding machine; no consideration is given to writing programs or performing complicated keyboard calculations.

The chapter contains selected topics, all of which are more fully discussed elsewhere in this manual.

A glance at the calculator's keyboard will show that the numerical keys (0 through 9 and the decimal point) closely resemble the layout of the keyboard of a typical business oriented, adding machine. The '5' key even has a slight bump, to facilitate touch typing when entering a long list of numbers into the calculator.

Before you can start to solve problems, there are some facts about the calculator that you should know; the explanation of these has been kept as brief as possible.

The turn-on procedure in Chapter 1 of this manual has instructions for verifying that the calculator is adjusted to your particular line voltage, that it has the correct fuse, and so on. If your calculator has already been working in your office, then you need not be concerned with such details and the following information will suffice.

TURN-ON

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; plug the other end of the cord into a suitable power outlet, such as a wall-socket. Plugs and connectors are keyed so that they cannot be connected improperly.

2. If the calculator is switched off (all lights on the keyboard are off and the display window is completely dark): The OFF/ON switch is located on the front of the calculator, below the keyboard and to the right. Set the switch to the ON position; several lights on the keyboard will come on and zeros will appear in the display, indicating that the calculator is ready to use.

The following steps prepare the calculator for making calculations.

PREPARING THE CALCULATOR

1. If there are no numbers visible on the display, or if the display is flashing, then the calculator is running a program. To stop the program:

    PRESS: [STOP]

**PREPARING
THE
CALCULATOR**
(continued)

2. The calculator must be in the 'run' mode, otherwise you cannot make calculations. If the light below the RUN key is not lit:

PRESS: ⬚ RUN ⬚

3. If an Option 004 Printer is not installed, ensure that the light below the KEYLOG key is out. If the light is on:

PRESS: ⬚KEY LOG⬚

(Pressing KEYLOG turns the light on if it was out, or turns it out if it was on.)

For complete details of printer operation, refer to Chapter 6.

4. If the display contains numbers other than zeros, you may wish to clear it. Clearing the display is not usually a necessary operation; however, it is convenient to start a calculation with a cleared display.

PRESS: ⬚CLEAR⬚

**THE
DISPLAY**

The display consists of three rows of numbers with names beside them. Each row is referred to as a 'register' and its name describes its use:

keyboard x: The numbers appear in this register as the numeric keys are pressed.

accumulator y: The results of arithmetic operations 'accumulate' in this register.

temporary z: Numbers can be temporarily held here while arithmetic operations are being performed in the other two registers.

Throughout the remainder of this chapter the registers will be identified as the 'X-register', the 'Y-register' and the 'Z-register' (or, simply, as X, Y and Z).

**POSITION OF
THE DECIMAL POINT**

The decimal point can be positioned anywhere in the display registers by pressing the 'FIX ( )' key and following it with any number key (0 through 9). For example:

a. If you are working only with whole numbers, that is, no decimal digits,

PRESS: ⬚FIX ( )⬚ ⬚ 0 ⬚

b. If you are dealing with dollars and cents (e.g. $20.37) then you will wish to see two decimal digits (the cents).

PRESS: [FIX ( )] (2)

or, if you wish to see hundredths of a cent,

PRESS: [FIX ( )] (4)

Occasionally you may see a number consisting of ten digits (with the decimal point located after the first digit) and followed by either a space, or a minus sign, and then two more digits. This type of display is called 'floating point'; if pressing the 'FIX ( )' key, followed by a number key, does not change the 'floating' number to a normal 'fixed' number, then refer to Chapter 3, where the difference between 'floating' and 'fixed-point' numbers is described.

ARITHMETIC

All arithmetic consists of four basic steps:

1. Enter the first number into the X-register.
2. Move that number into the Y-register.
3. Enter the second number into the X-register.
4. Do the calculation; the result appears in the Y-register.

$$
\begin{array}{lll}
\text{(ADD)} & y + x & \rightarrow y \\
\text{(SUBTRACT)} & y - x & \rightarrow y \\
\text{(MULTIPLY)} & y \times x & \rightarrow y \\
\text{(DIVIDE)} & y \div x & \rightarrow y
\end{array}
$$

Now do examples 1 through 4 below and observe that each example follows the four basic steps given above.

EXAMPLE 1: ADDITION
$$18 + 6 = 24 \rightarrow Y$$

PRESS: (CLEAR) [FIX ( )] (0)

(continued)

ARITHMETIC
(continued)

(Step 1) PRESS: [1] [8]

(Step 2) PRESS: [↑]

(Step 3) PRESS: [6]

(Step 4) PRESS: [+]

DISPLAY:

temporary z	0.
accumulator y	24.
keyboard x	6.

EXAMPLE 2: SUBTRACTION
$17.40 − $6.23 = $11.17 → Y

PRESS: [FIX( )] [2]

PRESS: [1] [7] [•] [4] [0]

PRESS: [↑]

PRESS: [6] [•] [2] [3]

PRESS: [−]

DISPLAY:

temporary z	24.00 ← †
accumulator y	11.17
keyboard x	6.23

†Unless you cleared the registers by pressing CLEAR before you started the calculation, the result of Example 1 (24.00) moved up to the Z-register when ↑ was pressed.

EXAMPLE 3: MULTIPLICATION

36 X 9 = 324 → Y

PRESS: (3) (6)

PRESS: (↑)

PRESS: (9)

PRESS: (×)

DISPLAY:

temporary z	11.17
accumulator y	324.00
keyboard x	9.00

Result of Example 2

EXAMPLE 4: DIVISION

120 ÷ 16 = 7.5 → Y

PRESS: (CLEAR) (not a necessary operation)

PRESS: (1) (2) (0)

PRESS: (↑)

PRESS: (1) (6)

PRESS: (÷)

DISPLAY:

temporary z	0.00
accumulator y	7.50
keyboerd x	16.00

**KEY
FUNCTIONS**

Following are brief descriptions of the keys already used (except 0 through 9 and the decimal point) and some other keys.

RUN   Selects the operating mode for making calculations and running programs.

FIX ( )   Followed by one of keys 0 through 9 positions the decimal point on the display.

CLEAR   Clears the display. Also clears the *a* and *b* storage registers (see Storing a Constant, below).

CLEAR *x*   Clears only the X-register. Use this key if you make a wrong number entry; press CLEAR x and then enter the correct number.

↑   Duplicates X in Y without changing X. Shifts Y up to Z. The contents of Z are lost.

↓   Opposite of ↑: Duplicates Z in Y without changing Z. Shifts Y down to X. The contents of X are lost.

$x \rightleftarrows y$ Exchanges X and Y without changing Z.

↑ ROLL   'Rolls' the display:
     Shifts X up to Y.
     Shifts Y up to Z.
     Shifts Z down to X.

+   Adds X to Y; the sum appears in Y. X and Z remain unchanged.

−   Subtracts X from Y; the difference appears in Y. X and Z remain unchanged.

×   Multiplies Y by X; the product appears in Y. X and Z remain unchanged.

÷   Divides X into Y; the quotient appears in Y. X and Z remain unchanged.

CHG SIGN   Changes the sign of the X-register. Arithmetic may be performed with negative numbers.

The calculator contains many registers which, if they could be seen, would appear similar to the display registers. These are the data-storage registers (fully described in Chapter 4) which are used to store numbers; the numbers can then be recalled at any time to the display and used in a calculation. Only two of these registers, named *a* and *b*, need be of concern here because they are the easiest to use.

Computer
Museum

Stores X into the *a*-register (or *b*-register) without changing the X-register.

or

Stores Y into the *a*-register (or *b*-register) without changing the Y-register.

or

Recalls *a* to the X-register without changing the *a*-register.

Recalls *b* to the X-register without changing the *b*-register.

Clears all three display registers and *a* and *b*.

EXAMPLE 5: STORING A CONSTANT

Suppose the unit-price of a certain item is $132.57 and you wish to calculate the cost of buying various quantities. To save entering the price for each calculation, you can store it as a constant and then recall it, with one keystroke, each time you need it.

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

First store the unit-price in the *a*-register.

PRESS: [FIX ( )] [2] [CLEAR]

PRESS: [1] [3] [2] [·] [5] [7]

PRESS: [x→( )] [a]

**STORING A
CONSTANT
(continued)**

Now make the calculations, using each quantity in turn and recalling a each time.

PRESS: [4] [7]

PRESS: [↑]

PRESS: [a]

PRESS: [×]

DISPLAY: 6230.79 → y

PRESS: [2] [9] [↑] [a] [×]

DISPLAY: 3844.53 → y

PRESS: [3] [6] [↑] [a] [×]

DISPLAY: 4772.52 → y

and so on for each quantity.

**ACCUMULATING
TOTALS**

There are several ways to accumulate totals in the calculator; two of the simpler methods are described here.

If you have a list of numbers which are to be totalled, all you need do is key each number into the X-register and press the 'plus' key to accumulate the total in the Y-register. Be sure that the Y-register is cleared first.

EXAMPLE 6: ACCUMULATING IN Y
Add 9, 36, 25, ... , etc.

PRESS: [CLEAR] (accumulative total in Y = 0)

PRESS: [9] [+]

DISPLAY: $9.$ → y

PRESS: [3] [6] [+]

DISPLAY: 45. → y

PRESS: [2] [5] [+]

DISPLAY: 70. → y

In Example 6, numbers were totalled as they were keyed into the calculator; sometimes, however, you will want to total the results of individual calculations. You can do this in one of the data-storage registers and then recall the final total to the display. The following keying sequence adds the number in the Y-register to the number in register b (the a-register can also be used in this way).

[y→( )] [+] [b]

EXAMPLE 7: ACCUMULATING IN STORAGE

Repeat the calculations of Example 5 and total the results, as they are calculated, in register b.

PRESS: [FIX( )] [2]

PRESS: [CLEAR] (ensures b-register contains zero)

Store the constant (the unit-price) in a:

PRESS: [1] [3] [2] [.] [5] [7]

PRESS: [x→( )] [a]

Make the first calculation:

PRESS: [4] [7] [↑] [a] [×]

Add the result to the total (currently zero) in register b:

PRESS: [y→( )] [+] [b]

(continued)

ACCUMULATING TOTALS
(continued)

Make the second calculation and add the result to the total:

PRESS: [2] [9] [↑] [a] [×]

PRESS: [$y \rightarrow ( )$] [+] [b]

Make the third calculation and add the result to the total:

PRESS: [3] [6] [↑] [a] [×]

PRESS: [$y \rightarrow ( )$] [+] [b]

To recall the final total in b to the X-register:

PRESS: [b]

DISPLAY: ($) 14847.84 → x

which is the total, in dollars, of the three calculations.

PROGRAMS

A program enables the calculator to automatically execute the keys necessary to make a particular calculation. First the program must be loaded into the calculator's memory; this teaches the calculator which keys are required and the order in which they are to be executed.

The program steps can be keyed into the calculator, or they can be loaded from magnetic program cards. Once loaded, the calculator can remember that program until another program is loaded over it, or until the calculator is switched off. Magnetic cards are a convenient way of permanently storing programs because they enable even long programs to be loaded in seconds.

Once a program is loaded, the calculation can be made any number of times. In general, it requires that you key in some data numbers and then start the program by pressing the CONTINUE key. In some cases, that will be all that is required; in other cases, the program may stop several times for you to key in more numbers.

PROGRAM USER-INSTRUCTIONS

Due to the versatility of the calculator and to the variety of programs, it is not possible to give a precise set of instructions which will enable you to run every program. Therefore, the person who asks you to run his program must give you complete and concise user-instructions to run that

program; if he does not, then you should ask for them. This is not a reflection on your intelligence, it is simply recognizing the fact that it is impossible for a non-programmer to run a program without the proper user instructions; even an experienced programmer would, in most cases, find it easier to write his own program rather than try to run an existing one without any user-instructions.

Here are some general guidelines to the type of information which user-instructions should contain. (Following these is a sample program and then instructions to record it on a magnetic card and to load it, from the card, into the calcultor.) The guidelines assume that the program is known to work properly and that it is already recorded on a magnetic card (or on more than one card, if it is a long program).

USER-INSTRUCTION GUIDELINES

1. What the program does.
2. How many magnetic program card sides have to be loaded.
3. Where in memory is the loading to start. . . . usually this will be at the beginning of memory (press END), but not always.
4. Where in memory to start running the program. . . usually this will be at the same place as in step 3, but, again, not always.
5. When to key in the required data numbers and when to press CONTINUE.
6. How to interpret the display each time the program stops and where to look for the results.
7. Any other information you may need.
8. A set of test data numbers with known results which you can use to run the program and check that you are following the instructions properly.

Before the example program can be demonstrated, it must be loaded into the calculator; please press the following keys, in exactly the same way as in the earlier examples. If you miss a key, or press a wrong key, do not try to correct the mistake, go right back to the beginning of the complete keying sequence. Ignore the unusual display which appears when you press the 'PRGM' key.

AN EXAMPLE PROGRAM

PRESS: [RUN] [END] [PRGM] [CLEAR] [STOP] [x→( )] [*a*]

PRESS: [↑ ROLL] [STOP] [↑] [↑] [*a*] [×] [y→( )]

PRESS: [+] [*b*] [GO TO] [5] [END] [RUN]

The program is now loaded.

**AN EXAMPLE
PROGRAM
(continued)**

USER INSTRUCTIONS
This program makes the same type of calculations which you made previously in Example 7. The program has been written so that you first key in the unit-price (which can be any value); as you then key in each quantity the program automatically calculates the quantity cost and accumulates the total cost. When all of the quantities have been keyed in (and you can key in any number of quantities you please), you can then recall the total cost from memory, just exactly as was done in Example 7.

NOTE
It is assumed that the program is already loaded.

Steps 1 through 7 (below) constitute a generalized procedure to run the program; specific keystrokes, using the numbers from Example 7 as test-data, follow the generalized procedure.

1. Position the decimal point, then start the program at the beginning of memory:

    PRESS: [FIX ( )] [2] [END] [CONTINUE]

    DISPLAY:

temporary z	0.00
accumulator y	0.00
keyboard x	0.00
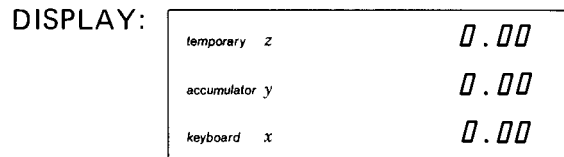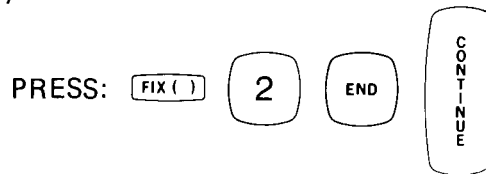
2. Key in the unit-price and press [CONTINUE]

    DISPLAY:

temporary z	0.00
accumulator y	Price
keyboard x	0.00
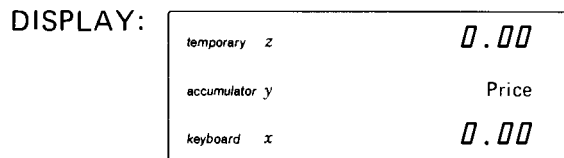
3. Key in the first quantity and press [CONTINUE]

DISPLAY:

temporary z	Quantity
accumulator y	Cost
keyboard x	Price

4. Key in the next quantity and press [CONTINUE]

       DISPLAY: Same as 3.

5. Repeat step 4 for each of all remaining quantities; do step 6 when all quantities have been used.

6. Recall the total cost:

       PRESS: [b]

       DISPLAY:

temporary z	Last Quantity
accumulator y	Last Cost
keyboard x	Total Cost

7. To run the program with a new unit-price or with different quantities:

       PRESS: [END] [CONTINUE]
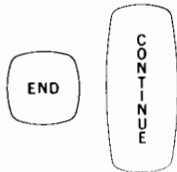
and go back to step 2.

Now run the program using the numbers from Example 7 as test-data: unit-price of $132.57 and quantities of 47, 29 and 36.

      (1)   PRESS: [FIX( )] [2] [END] [CONTINUE]

**AN EXAMPLE
PROGRAM**
(continued)

(2)   PRESS: [1] [3] [2] [·] [5] [7] [CONTINUE]

(3)   PRESS: [4] [7] [CONTINUE]

(4)   PRESS: [2] [9] [CONTINUE]

(5)   PRESS: [3] [6] [CONTINUE]

(6)   PRESS: [b]

FINAL DISPLAY:

temporary	z	36.00
accumulator	y	4772.52
keyboard	x	14847.84

**RECORDING
AND LOADING**

The procedure given here demonstrates how to record the example program on a magnetic card and then how to load it back into the memory (you will need one side of an unused magnetic card). This procedure is not necessarily correct for other programs.

NOTE

Before proceeding, please read the following part of the section titled 'Magnetic Program Cards' in Chapter 5: From the beginning of the section as far as (but not including) the paragraph describing the RECORD key.

To record (if the program is no longer in the memory, refer to the keying sequence on Page 2-11 to load it):

1. Go to the beginning of the program, which in this case is at the beginning of the calculator's memory:

    PRESS: [END]

2. Insert the card into the card-reader.

    PRESS: [RECORD]

    (as soon as the card stops moving, remove it from the card-reader).

The program is now recorded (it is suggested that you do not protect the recording because this program has little practical value except as a demonstration).

To load:

1. Go to the beginning of memory:

    PRESS: [END]

2. Insert the card, as in step 2 above,

    PRESS: [LOAD]

    (when the card stops moving, remove it from the card-reader).

The program is now loaded and ready to run exactly as before.

NOTES

This chapter of the manual contains general information of the type which does not readily classify with particular keys. Any special terms are explained wherever they first appear. The examples are not intended to teach the keyboard but to illustrate the particular points being made in the text. It is, however, recommended that you perform the examples because they will help you to develop a 'feel' for your calculator.

Chapter I of this manual contains an introductory description of the calculator and details of the options and peripheral devices available to build a 9800 System.

READ-ONLY MEMORIES (ROM)

The calculator uses several memories, which can be classified into two types -- 'read/write' and 'read-only'. The program memory and data-storage memory are of the read/write type, because information can be taken from them ('read') or stored into them ('write') at will.

Information in the read-only memories, on the other hand, is 'hardwired'; that is, it is permanently stored and cannot be changed by the user. (The word ROM, which also appears in Chapter I, is derived from the initial letters of the words Read-Only-Memory.)

The hardwired programs contained in the ROM's define the keys. Each time a key is pressed, one or more of these programs is used to execute the instruction, or to calculate the function, called for by that key. The basic calculator contains a ROM which defines all of the keys except the half-keys in the left-hand block. These half-keys are defined by the various plug-in ROM's, thus they can have any of several definitions, depending upon which ROM is installed. Without the plug-in ROM's, the calculator treats each of these half-keys as if it were a 'no-operation'.

The plug-in ROM's are easily installed by pushing them into the slots located on top of the calculator and behind the display. The calculator should be switched off whenever a ROM is to be installed or removed. The operating manual supplied with each ROM describes any special procedures; the manuals also detail in which slots particular ROM's may be installed.

NOTE
The calculator will not be damaged if a ROM is plugged in, or removed, while the calculator is switched on. However, it may result in some arbitrary display which can be cleared only by switching the calculator off, and leaving it off for a few seconds before switching it back on.

OPERATING CHARACTERISTICS

SECONDARY KEY-FUNCTIONS

In addition to the characters molded into the top of each key, there is also a secondary character stamped on the front of many of the keys. These characters represent a secondary key-definition which is used by the Model 11219A Printer (Option 004) if the Printer Alpha ROM (Model 11211A) is installed, and by the Model 61 Typewriter. In the basic calculator these characters have no meaning and may be ignored.

KEY COLORS

The keys are classified into groups by color. The color gives some information about each key; however, the grouping is not absolute and there is some overlap. Following is a brief description of the significance of each color.

YELLOW ORANGE: A 'caution' in that the CLEAR, LOAD and RECORD keys can change data and program steps stored either in the memory or on magnetic cards. Pushing one of these keys accidentally could result in losing useful information and wasting a great deal of programming time.

YELLOW GREEN: The CONTINUE and STOP keys are used to (respectively) start and stop program execution; it is, therefore, desirable that they be easily identifiable.

MINT GREY: Numeric and decimal point keys, used to enter data.

OLIVE GREY: Simple arithmetic functions. The inverted L-shape formed by these keys also accents the numeric keys, thus emphasizing their '10-key adding machine' function (see Chapter 2).

MOSS GREY: General purpose programmable keys.

DARK OLIVE GREY: Control keys, non-programmable.

THE DISPLAY

> **NOTE**
> a. If the calculator is not switched on, refer to the turn-on procedure in Chapter I.
> b. If the calculator is running a program (the display is flashing or blanked) press the STOP key.
> c. Press the RUN key.

The calculator's display consists of three lines labelled, from top to bottom, 'temporary z', 'accumulator y', and 'keyboard x'. Each line, called a 'register', can contain a data number consisting of one or more digits. As you will see shortly, it is possible to change the form of the display (reposition the decimal point, truncate insignificant digits, etc.) by pressing certain combinations of keys. Within limits, you may display the numbers in any form to suit your needs and yet you will not be affecting the actual numbers because they are also stored elsewhere in the calculator.

The numbers seen in the display are derived from the numbers stored in three special registers: the X-register, the Y-register and the Z-register. These are the three working registers; calculations and operations which appear to be occurring in the display, actually occur in the working registers. Then, when the calculator is idle, that is it is neither calculating nor running a program, the numbers in the X, Y and Z registers are displayed in the corresponding display registers: the contents of the X-register in 'keyboard x', and so on.

It is important to understand that the format of the displayed number, and to some extent even the display itself, is purely a user convenience; it does not affect the accuracy of the calculator. Apart from this, however, there is no reason, from the operator's point of view, why a distinction need be made between the three working registers and the three display registers; therefore, throughout the remainder of this manual, the two types of register will be assumed to be the same.

Keyboard x - This register displays numbers as they are entered from the keyboard, one digit at a time. (See note on Page 3-2.)

> EXAMPLE:
> Enter 6.34 into the X-register.
>
> Press the following keys in the order shown, from left to right, line-by line ['FIX (2)' positions the decimal point].
>
> PRESS: [FIX ( )] [2] [CLEAR]
>
> PRESS: [6] [•] [3] [4]
>
> DISPLAY: $6.34 \rightarrow x$
>
> ("6.34 appears in the X-register").

Accumulator y - the result of an arithmetic operation between two numbers, one in the Y-register and one in the X-register, appears in the Y-register.

> EXAMPLE:
>
> 6.34 ÷ 2 = 3.17
> (with 6.34 → x)
>
> PRESS: [↑] [2] [÷]
>
> DISPLAY: $3.17 \rightarrow y$
> $2.00 \rightarrow x$

Temporary z - used to temporarily store a number while arithmetic operations are performed on the numbers in the X- and Y-registers. The number in Z can then be returned to Y when it is again required.

EXAMPLE:

$20 \div (3 + 2) = 4$
(20 will be stored in Z while 3 and 2 are added.)

Enter 20:

PRESS:　[2]　[0]　[↑]

DISPLAY: 20.00　→　y

Enter the remaining data:

PRESS:　[3]　[↑]　[2]

DISPLAY: 20.00　→　z
3.00　→　y
2.00　→　x

Add the numbers in X and Y (without affecting Z):

PRESS:　[+]

DISPLAY: 5.00　→　y

Return the numbers in Z and Y to Y and X respectively:

PRESS:　[↓]

DISPLAY: 20.00　→　y
5.00　→　x

Divide:

PRESS:　[÷]

DISPLAY: 4.00　→　y

The calculator stores all data and performs all calculations in 'floating-decimal point'. The display, on the other hand, may be either in floating- or fixed-decimal point (selected by the FLOAT or FIX keys, respectively).

A 'fixed-point' number is one which appears in the form in which numbers are most commonly written, with the decimal point correctly located (e.g. 123.45).

A 'floating-point' number is one written in a convenient shorthand: the decimal point is always located immediately after the most significant digit (excluding zero) and the number is followed by an exponent. The exponent, written as a positive or negative power of 10, represents the number of places and the direction which the decimal point should be moved to express the number as a fixed point number. The following examples illustrate the relationship between fixed-point and floating-point numbers.

FIXED   FLOATING
(a) 1234.5   = 1.2345×10^3 ←⁓ (exponent)
(b) 0.0012345 = 1.2345×10^{-3}
(c) 1.2345   = 1.2345×10^0

In the calculator, when numbers are displayed in floating-point the exponent appears as a two-digit number to the right of the display; if the exponent is negative, then the minus sign also appears.

EXAMPLE:
   Display example (b), shown above, in fixed-point and then in floating-point format.

PRESS: [CLEAR] [FIX( )] [8]

PRESS: [.] [0] [0]

PRESS: [1] [2] [3] [4] [5]

DISPLAY: 0.0012345 → x

PRESS: [FLOAT]

DISPLAY: $1.234500000-03$ → x

exponent (negative)

**GUARD
DIGITS**

Regardless of the way in which a number is displayed, the calculator always stores the number in floating-point. Furthermore, regardless of the number of digits entered, or displayed, the number is stored with 12 significant digits and a two-digit exponent.

Up to, and including, 10 significant digits and the two exponent digits can be displayed. The remaining two significant digits, which are not displayed, are called the 'guard digits'. The purpose of the guard digits (which are not to be confused with the exponent digits) is to maintain greater than 10-place accuracy during calculations and also to automatically round the tenth displayed digit.

The examples which follow illustrate the guard digits. The first example shows that they do maintain accuracy; the second shows that they do round the tenth digit and also shows you a method of viewing the guard digits.

EXAMPLE:

Divide 6.000000001 by 3 and then multiply the result by 3.

PRESS: [CLEAR] [FIX ( )] [9]

PRESS: [6] [·]

PRESS: [0] [0] [0]

PRESS: [0] [0] [0]

PRESS: [0] [0] [1]

DISPLAY: *6.000000001*   → *x*

PRESS: [↑] [3] [÷]

DISPLAY: *2.000000000*   → *y*
             3.000000000   → *x*

If there were no guard digits, then, with the numbers currently displayed in the X- and Y-registers, you would expect the subsequent multiplication to result in 6.000000000, and not the 6.000000001 originally entered.

PRESS: ⨯

DISPLAY: $6.000000001 \rightarrow y$

Accuracy is maintained because of the guard digits.

The next example shows how you may easily view the guard digits at any time.

EXAMPLE:

If the π key is pressed, the value for pi is entered with 12 digits, although only the first ten are displayed. The last displayed digit is rounded up, from 3 to 4, because the guard digits contain 60.

PRESS: FLOAT π ↑

DISPLAY: $3.141592654\ 00 \rightarrow y$

last digit rounded

exponent

By subtracting 3.1 from pi, the guard digits can be seen:

PRESS: 3 · 1 −

DISPLAY: $4.159265360-02 \rightarrow y$

(1) (2)

exponent

(1) = formerly 4, the last displayed digit
(2) = formerly the guard digits

With the fixed-point display, if the number in any one display register 'overflows' (i.e., it becomes too large to be displayed with the decimal point in its current position) then that number will automatically be displayed in floating-point.

REGISTER OVERFLOW AND UNDERFLOW

EXAMPLE:

PRESS: CLEAR FIX( ) 7

This specifies that there can be up to 7 digits to the right of the

decimal point and, therefore, no more than 3 to the left if the display is to remain in fixed-point format.

PRESS: [1] [2] [3] [·] [0]

PRESS: [4] [5] [6] [7] [8] [9]

DISPLAY: 123.0456789 $\rightarrow x$

PRESS: [FIX ( )] [8]

DISPLAY: $1.230456789 \ 02$ $\rightarrow x$

After 'FIX (8)' was pressed, only two digits could appear to the left of the decimal point; because the number to be displayed had three digits preceding the decimal point, overflow occurred and the number was automatically displayed in floating-point.

Now experiment by pressing the FIX key and following it by any one of the numeric keys (0 through 9). Each time you do this observe that the decimal point shifts to a position specified by the numeric key pressed. Also notice that the last displayed digit is automatically rounded, depending upon the value of the non-displayed digits. Any time you press 'FIX (7)' you will see that, no matter how you have previously changed the display, the number still appears as it was originally entered (provided that no other keys were pressed to change the original number entry).

When the displayed number 'underflows' (i.e. becomes too small for any significant digits to be displayed) then the displayed number does not change to floating-point. Instead, zeros appear in the display; however, the significant digits are still stored in the calculator in floating-point, and will still be included in any subsequent calculation.

EXAMPLE:

PRESS: [FIX ( )] [5]

PRESS: [·] [0] [0]

PRESS: [0] [0] [4]

DISPLAY: 0.00004 $\rightarrow x$

PRESS: [FIX] 4

DISPLAY: 0.0000 → x

The number underflows and zeros appear in the X-register.

PRESS: [FLOAT]

DISPLAY: $4.000000000-05$ → x

Again, accuracy is maintained.

In either case, 'overflow' or underflow', significant digits are not lost because the calculator always operates in floating-point, with 12 significant digits and the two-digit exponent, regardless of the display.

The dynamic range of the calculator is from $\pm10^{-98}$ to $\pm9.999999999(99)$ $\times$ 10^{98}. Whenever this range is exceeded during a calculation, the STATUS light (see below) lights and the 'overflowed' register contains one of the following:

RANGE

$\pm9.999999999(00) \times 10^{98}$, if the exponent would have exceeded 98; zero, if the exponent would have been less (more negative) than -98.

Calculations which normally result in zero, such as subtracting a number from a number equal to itself, are not considered to exceed the range of the calculator.

Even though numbers with exponents equal to 10^{99} or 10^{-99} can be entered from the keyboard into the calculator, such (keyed-in) numbers should never be included in a calculation. If they are, the result will usually be meaningless and may or may not turn on the STATUS light.

The STATUS light poses the question "What happened?". It alerts the user that some special condition exists and that an investigation should be made to determine what that condition is. With one exception, the light indicates that an error has occurred. Errors can be of two types: procedural, such as calling for a non-existent data-storage register; or mathematical, such as dividing by zero or exceeding the range of the calculator. If a program is running, a procedural error will stop the program whereas a mathematical error will not. Once the light is lit, it cannot be turned off unless the program has stopped; it then goes off as soon as the next key is pressed (provided that the key pressed does not constitute another error, thus turning the light on again).

STATUS

STATUS
(continued)

The one STATUS condition which is not an error occurs whenever the calculator is 'initialized'; this occurs at turn-on or (very rarely) if there is a sudden and drastic loss in line-power.

NOTE

The remainder of this chapter contains a description of the conditions which light the STATUS light. It is recommended that you skip this material and refer to it only as you need it. Most of the material here will be meaningless to you if you have read only this far in the manual because it is based on information given in the rest of this manual.

STATUS
CONDITIONS

Other than at turn-on, the STATUS light indicates that an error has occurred. Non-drastic errors, such as division by zero, do not stop program execution. On the other hand, drastic errors, such as designating a non-existent address, do stop program execution and in some cases destroy data in the display registers. This does not present any problem, however, because drastic errors normally occur only in new programs, when they are first being tested and debugged.

Before the conditions which light the STATUS light are detailed, here is some background information necessary to an understanding of STATUS light operation.

The basic calculator has 500 program locations (addresses 0000 through 0499), Option 002 has 1012 (addresses 0000 through 1011) and Option 003 has 2036 (addresses 0000 through 2035). In all three versions of the calculator the program counter can be stepped up to 2035, both in 'program' and in 'run' modes. In the basic calculator and in Option 002, even though the extra addresses can be displayed, no program steps can be loaded into them. All addresses above 0499 in the basic calculator, and above 1011 in Option 002, appear to contain 00 (the key-code for zero). If the extra steps are executed in the 'run' mode, all they can do is load zeros into the X-register until the last step, 2035, has been executed. After step 2035 an 'end of memory' program is executed.

The 'end of memory' program consists of three steps permanently stored in what would be addresses 2036, 2037 and 2038. The steps are: CLEAR x, $\div$, END. The program clears the X-register and divides zero into the Y-register. The division by zero results in $9.999999999 \times 10^{98}$ appearing in Y and lights the STATUS light. The END stops program execution and sets the program counter to address 0000. Several of the STATUS conditions, listed below, result in the 'end of memory' program being executed.

Following is a list of conditions which light the STATUS light and the symptoms which can be observed for each condition.

1. TURN—ON:

   Selects RUN and FLOAT.

   Clears the display, the data-storage memory and the program memory, by loading zeros throughout.

   Sets program counter to address 0000.

   Clears the SET FLAG and any subroutine return-addresses.

   > **NOTE**
   >
   > When items 2 and 3 occur in a program, the program does not stop; the STATUS light remains lit after execution of the program is completed.

2. EXCEEDING THE RANGE (does not stop program execution):

   If the upper limit is exceeded, contains $\pm9.999999999 \times 10^{98}$ in the Y-register, or in the data register; the guard digits contain zeros.

   Contains zero if the lower limit is exceeded.

3. ILLEGAL MATHEMATICAL OPERATIONS (does not stop program execution):

   a. Division by zero results in the upper limit of the calculator's range being exceeded.

   b. Calculating the square-root of a negative number results in the square root of the absolute value of that number being displayed.

   c. Illegal operations associated with the plug-in ROM's are detailed in the separate ROM manuals.

4. REACHING THE END OF MEMORY:

   If the last programmable step in memory is executed, and it is not either a halt or a branch instruction, then the program counter automatically steps up to, and executes, the 'end of memory' program. The last programmable steps are 0499 in the basic calculator, 1011 in Option 002 and 2035 in Option 003.

5. NON—EXISTENT DATA STORAGE REGISTERS:

   The basic calculator has 51 data registers (49 numeric and a and b).... non-existent registers are 049 and above. Option 001 has 111 data registers (109 numeric and a and b).... non-existent registers are 109 and above.

   a. If the non-existent register is specified from the keyboard (either directly or indirectly) no operation occurs.

b. If specified (directly or indirectly) by program steps, no operation occurs and the program stops with the program counter pointing to the next step. For example:

(i)	STEP	KEY
	0266	$x \to (\ )$
	0267	0
	0268	5
	0269	0
	0270	CHG SIGN
	0271	+

(ii)	STEP	KEY
	0266	$x \to (\ )$
	0267	5
	0268	0
	0269	CHG SIGN
	0270	+

In case (i) the program stops, in the basic calculator, with the counter pointing to CHG SIGN in step 0270 (CHG SIGN is not executed).

In case (ii) a short-form address is used; here the counter points to the 'plus' in step 0270. The CHG SIGN in step 0269 is not executed but it does terminate the address.

Occasionally, when indirectly addressing in an Option 001 calculator, an improper address will be interpreted as a proper address. This occurs if the improper address contained in the indirect register meets the following two conditions:

　　Its absolute value is equal to or greater than 10^3.

　　Its three most significant digits constitute one of the numbers 100 through 108.

Such improper addresses will designate (proper) addresses 100 through 108, respectively.

6. NON—EXISTENT PROGRAM ADDRESSES GREATER THAN 2035:

    a. Given from the keyboard - no operation occurs.

    b. Given as program steps - program execution stops with the program counter pointing to the step preceding the first digit of the address, as shown by the arrows in the following examples:

```
  ──▶   GO TO            GO TO        ──▶   IF  x = y          IF  x = y
         2        ──▶  SUB / RETURN          2        ──▶  SUB / RETURN
         0              2                    0              2
         4              0                    4              0
         0              4                    0              4
                        0                                   0
```
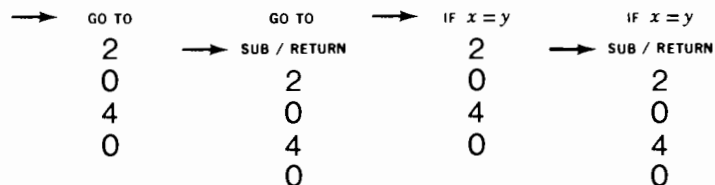
7. NON—EXISTENT PROGRAM ADDRESSES LESS THAN 2036:

    a. Given from the keyboard - the program counter points to the specified address but the STATUS light **does not** light. If CON-TINUE is pressed, loads zeros into the X-register until step 2035 is reached, then executes the 'end of memory' program.

b. Given as program steps - the program counter branches to the address specified and continues program execution; zeros are loaded into the X-register until step 2035 is executed, the 'end of memory' program is then executed.

8. NON—EXISTENT LABELS:

a. Given from the keyboard - no operation occurs.

b. Given as program steps - program execution stops with the program counter pointing to the step following the label-call (the step is not executed).

<div align="center">

GO TO

LABEL

A

→   ↑

</div>

9. NESTING SUBROUTINES 6-DEEP:

Program execution stops with the program counter pointing at the return-address, that is the first step after the last one used to call the sixth subroutine.

<div align="center">

GO TO

SUB / RETURN

0

3

2

0

→ $x \rightleftarrows y$

</div>

10. EXECUTING A RETURN WITH NO RETURN-ADDRESS:

Program execution stops with the program counter pointing to the step immediately following the address containing the RETURN.

<div align="center">

SUB / RETURN

→   CLEAR

</div>

11. RECORDING ON PROTECTED MAGNETIC CARDS:

The STATUS light remains on only as long as the protected card is moving through the card reader. No recording is made and the program counter continues to point at the address to which it was originally set. When the card stops, the INSERT CARD light remains on; either insert an unprotected card or press the STOP key.

NOTES

This chapter describes operation of the basic calculator from the keyboard. The keys of immediate interest are, mostly, those contained in the two center blocks of the keyboard. Each key is described in two parts ..... a brief explanation in bold print, which may be used for quick reference, followed by more detailed information and examples. The keyboard presentation near the front of this manual (Page vi) contains an index of the pages on which the 'bold' print explanations appear. Keys with multi-functions may appear in several places.

The characters stamped on the front of the keys define the use of these keys with the -hp- 11219A Printer (Option 004) and the Model 9861A Typewriter.

The half-keys in the left-hand block serve no function on a basic calculator which has no plug-in ROM's.

The extreme right-hand block of keys is used almost exclusively for programming purposes, described in Chapter 5.

In general, keys operate in the same way whether used from the keyboard or as steps in a program. However, there are occasional differences between keyboard and program operation; any such differences will be noted where applicable. All keys can be used as program steps except where noted.

Examples are presented in either one of two formats, depending upon which is more suitable for the particular example.

FORMAT I
Keys are pressed in the order shown, from left to right, first line then second line, etc.

PRESS: [CLEAR x] [2]

PRESS: [ENTER EXP] [CHG SIGN] [1] [0]
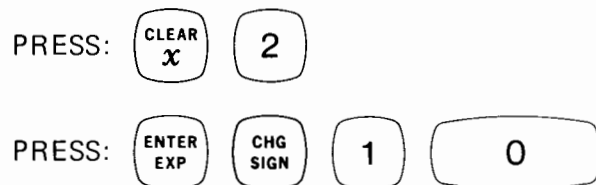
FORMAT II
Keys are pressed in step sequence 1, 2, 3, 4, etc.

STEP	KEY
1	CLEAR x
2	2
3	ENTER EXP
4	CHG SIGN
5	1
6	0

Use of Format II will help to introduce present 'non-programmers' to programming because this is a simplified way of writing a program.

Statements describing the contents of the display registers are shown as, for example:

DISPLAY: $9.00 \longrightarrow y$

interpreted as '9.00 appears in (or goes to) the Y-register'.

NOTE

If the light below the KEYLOG key is lit and there is no printer installed (see Chapter 6), then the keyboard will be inoperative; to turn the light out

PRESS: [KEY LOG]

If the light below the RUN key is not lit,

PRESS: [RUN]

It will be assumed throughout this section that this light is on.

INITIALIZE KEYS

The LINE OFF-ON switch applies ac power to the calculator. Refer to Chapter 1 for power requirements and turn-on instructions.

LINE
OFF ◖█ ON

At turn-on, the calculator is automatically 'initialized' as follows:

Turns on the STATUS light (Chapter 3).

Selects RUN and FLOAT modes (this chapter).

Clears the display and data-storage registers by loading zeros through-out (this chapter).

Clears the program memory by loading the key code for zero (00) in all program step locations (Chapter 5).

Sets the program counter to 0000 (Chapter 5).

Clears the FLAG (Chapter 5).

Clears any subroutine 'return-addresses' (Chapter 5).

The RUN and PRGM keys (not programmable) select the calculator's mode of operation; the lights immediately below the keys light to indicate which mode has been selected.

`RUN` `PRGM`

`RUN` 'Run' mode is used for all calculator operations except those noted under PRGM, below. 'Run' operations include all calculations, running programs, all magnetic card operations, etc.

`PRGM` 'Program' mode is used when loading programs into memory from the keyboard and when editing programs (refer to Chapter 5).

The FLOAT and FIX keys (not programmable) select the mode of display, 'floating' or 'fixed' decimal point, respectively; the lights below the keys light to indicate the selected mode. Choice of display mode is purely a user convenience, it does not affect the calculator's accuracy of calculation.

`FLOAT` `FIX ( )`

(continued)

NOTE

For a more complete description of the two modes and for examples, refer to the following paragraphs in Chapter 3:

'Floating- and Fixed-Point', 'Guard Digits', and 'Register Overflow and Underflow'.

FLOAT   FIX ( )

(continued)
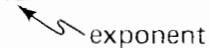
INITIALIZE KEYS

FLOAT   Numbers of up to (and including) ten significant digits are displayed in floating decimal point (or scientific notation); the tenth digit is automatically rounded according to the contents of the two guard digits. The exponent, or power of ten multiplier, appears as two digits to the right of the number; the minus sign is included for negative exponents.

EXAMPLE:

NUMBERS: 12345.67898 = 1.234567898×10^4
DISPLAY: /.234567898 04
                              exponent

FIX ( )   'FIX ( )' followed by 'n' (any one of numeric keys 0 through 9) selects the fixed point display; 'n' positions the decimal point on the display. Numbers of up to (and including) ten significant digits are displayed as they are commonly written, with no exponent and the decimal point correctly located. The last displayed digit is automatically rounded if there are any less significant digits not being displayed.

'n' specifies the maximum number of digits which can be displayed to the right of the decimal point. As illustrated in Chapter 3, if the number becomes too large to be displayed with the decimal point in its current position, then overflow occurs. The overflowed display register automatically switches to floating decimal point.

If the number to be displayed becomes too small for any significant digits to be displayed, then underflow occurs, but the number does not switch to floating decimal point. In this case, the display will contain zeros but the significant digits will still be contained in the calculator.

DATA ENTRY KEYS

The numeric keys, 0 through 9, are used to enter numbers into the X-register. Numbers are entered serially, the last digit entered becoming the least significant digit.

0

through

9

EXAMPLE:

Enter: 12345.06789

PRESS: [FIX ( )] [5]

PRESS: [CLEAR] [1] [2] [3] [4] [5] [·]

PRESS: [0] [6] [7] [8] [9]

DISPLAY: *12345.06789* → x

See Enter Exp key for further examples of data entry.

CLEAR x — Clears only the X display register (0 → X). It is not necessary to use CLEAR x before a new data entry unless the previous entry is not terminated.

A number in the X-register is **terminated** as soon as any operation has been performed (e.g. ↑, +, etc.). In this case, the new data entry automatically replaces the old and CLEAR x is not required.

A number is **not terminated** if no subsequent operation has been performed; that is, if the last key used was one of the following:

[0] through [9], [·], [ENTER EXP] or [CHG SIGN]

(if CHG SIGN was used as part of the data entry). In this case, if CLEAR x (or CLEAR) is not used, the next number entered will not replace the unterminated number but become a part of it.

Use CLEAR x when correcting a wrong data entry.

CLEAR x

NOTE

Many of the examples in this section include the CLEAR x or CLEAR keys only because the preceding examples may have left an unterminated number in the X-register.

DATA ENTRY KEYS

CLEAR

Clears the display registers (0 → X, Y, Z). Clears the a and b data-storage registers (see Page 4-24) but does not affect any other data-storage register. Clears the SET FLAG key (explained under programming). 'CLEAR' has no effect upon the program steps stored in the programmable memory.

In general, it is not necessary to use CLEAR before each new calculation because, except in the case of an unterminated data entry (see CLEAR x key), the new data automatically replaces the old. Do not use CLEAR if data numbers stored in the a and b registers are to be retained.

·

Enters the decimal point into the X-register. Regardless of the display mode selected, it is not necessary to use the decimal point key when entering integers in fixed point format; if the decimal point is not used, it will be assumed to have followed the last digit entered (see ENTER EXP key for examples).

CHG SIGN

Changes the sign of the contents of the X-register; the number in X may be either terminated or unterminated (see CLEAR x key). If ENTER EXP precedes CHG SIGN, then the sign of the exponent is changed without affecting the sign of the number (see ENTER EXP key for examples).

ENTER EXP

Initializes the exponent in the X-register so that the next digit entries (0 through 9) and CHG SIGN affect only the exponent.

The exponent must be entered as a one- or two-digit number, the last digit entered becomes the least significant digit; if a third is entered, it will terminate the current data entry and become the first digit of a new data entry.

The calculator automatically corrects the exponent, according to the position of the decimal point, so as to display the number in scientific notation, when floating point display is selected, and to position the decimal point when fixed point display is selected (assuming overflow has not occurred).

DATA ENTRY KEYS

When entering negative exponents the CHG SIGN key should normally follow immediately after the ENTER EXP; however, it may also be used immediately after the first or second digit of the exponent.

If ENTER EXP follows any operation, except data entry, then 1 is entered into the X-register; thus it is not possible to change the exponent of a terminated number in X by means of the ENTER EXP key. However, an unterminated number in the X-register can be multiplied directly by any number of powers of 10 by entering a sequence of exponents (see Example 6 on Page 4-10).

The following examples illustrate some typical keying sequences used for data entry. Each example shows several keying sequences all of which produce the same display (except that, in some cases, non-significant zeros are blanked).

For clarity, the examples specify a particular display mode; however, it should be remembered that the display mode affects neither the data entry nor the accuracy of any calculations.

The CLEAR x key is required in most of these examples because the previous data entry is not terminated.

EXAMPLE 1:

Enter: 1×10^0 (1.0, .01 $\times 10^2$, etc.)

PRESS: [FLOAT]

PRESS: [CLEAR x] [ENTER EXP]

or

PRESS: [CLEAR x] [1]

or

PRESS: [CLEAR x] [·] [0] [1] [ENTER EXP] [2]

DISPLAY: $1.$ $\underline{00}$ → x

↖ exponent

ENTER
EXP

(continued)

DATA ENTRY KEYS

Enter: 1×10^3 $(10 \times 10^2, 1000,$ etc.)

PRESS: [CLEAR x] [ENTER EXP] [3]

or

PRESS: [CLEAR x] [1] [0] [ENTER EXP] [2]

or

PRESS: [CLEAR x] [1] [0] [0]

PRESS: [0]

DISPLAY: $1.000 \qquad 03 \quad \longrightarrow \quad x$

EXAMPLE 3:

Enter: 4×10^{-3} $(.004, .4 \times 10^{-2},$ etc.)

PRESS: [CLEAR x] [4] [ENTER EXP] [CHG SIGN] [3]

or

PRESS: [CLEAR x] [·] [0] [0] [4]

or

PRESS: [CLEAR x] [·] [4] [ENTER EXP] [CHG SIGN] [2]

DISPLAY: $4. \qquad -03 \quad \longrightarrow \quad x$

DATA ENTRY KEYS

EXAMPLE 4:

Enter: -4×10^3

PRESS: [CLEAR x] [CHG SIGN] [4] [ENTER EXP] [3]

or

PRESS: [CLEAR x] [4] [CHG SIGN] [ENTER EXP] [3]

DISPLAY: $-4.$ 03 $\rightarrow x$

EXAMPLE 5:

Enter: 1.2345×10^3 (1234.5, 12345×10^{-1}, etc.)

PRESS: [CLEAR x] [1] [·] [2] [3] [4] [5]

PRESS: [ENTER EXP] [3]

or

PRESS: [CLEAR x] [1] [2] [3] [4] [·] [5]

or

PRESS: [CLEAR x] [1] [2] [3] [4] [5]

PRESS: [ENTER EXP] [CHG SIGN] [1]

DISPLAY: 1.2345 03 $\rightarrow x$

**ENTER
EXP**

(continued)

DATA ENTRY KEYS

If the number in the X-register has not been terminated (no operation has been performed since the number was entered), then it can be multiplied by any number of powers of ten.

EXAMPLE 6:

$$768.32 \times 10^2 \times 10^6 \times 10^{-15} \times 10^9 = 7.6832 \times 10^4$$

PRESS: **CLEAR** x

PRESS: **7** **6** **8** **·** **3** **2**

PRESS: **ENTER EXP** **2**

PRESS: **ENTER EXP** **6**

PRESS: **ENTER EXP** **CHG SIGN** **1** **5**

PRESS: **ENTER EXP** **9**

DISPLAY: $7.6832 \qquad 04 \qquad \longrightarrow x$

NOTE

When multiplying an unterminated number by powers of ten, as in Example 6, care should be taken to ensure that the range of the calculator is not exceeded. If it is, then the final exponent will not be correct.

In this particular case, if the calculator's range is exceeded, the STATUS light will not light because the type of operation performed in Example 6 constitutes a data entry rather than a calculation.

DATA ENTRY KEYS

π **Enters the value for pi into the X-register.**

EXAMPLE:

PRESS: [FLOAT]

PRESS: π

DISPLAY: 3.141592654 00 → x
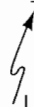
Last digit (actually 3) is rounded up
because the guard digits contain . . . 60

↑
Lost
$z \nearrow \rightarrow Z$
$y \rightarrow Y$
$x \rightarrow X$

↓
$z \rightarrow Z$
$y \rightarrow Y$
$x \searrow \rightarrow X$
Lost

↑
ROLL
$z \rightarrow Z$
$y \rightarrow Y$
$x \rightarrow X$

DISPLAY CONTROL KEYS

The four control keys (↑, ↓, Roll ↑ and x ⇄ y) are used to reposition the contents of the three display registers. Combinations of these keys allow the contents of any display register to be moved to any other display register. The examples below show only what each key does; other examples in this chapter illustrate actual usage of the keys in a calculation.
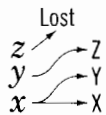
↑ **The contents of the Z-register are lost.**
The contents of the Y-register shift to Z.
The contents of the X-register appear in both Y and X.

EXAMPLE:

PRESS: FIX ( )   0

PRESS: CLEAR x   4   ↑   3   ↑

DISPLAY: 4.   → z
          3.   → y
          3.   → x
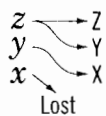
↓ **The contents of the Z-register appear in both Z and Y.**
The contents of the Y-register shift to X.
The contents of the X-register are lost.

EXAMPLE:

(with the display shown in the previous example),

PRESS: ↓

DISPLAY: 4.   → z
          4.   → y
          3.   → x
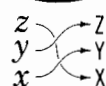
↑ ROLL **'Rolls' the display up without losing information.**
Shifts the contents of X to the Y-register.
Shifts the contents of Y to the Z-register.
Shifts the contents of Z to the X-register.

DISPLAY CONTROL KEYS

To 'Roll down' use two successive Roll ↑'s.

EXAMPLE:

PRESS: 1   ↑   2   ↑   3

DISPLAY: $1.$ → z
          $2.$ → y
          $3.$ → x

To 'Roll up'

PRESS: [↑ ROLL]

DISPLAY: $2.$ → z
          $3.$ → y
          $1.$ → x

To 'Roll down'

PRESS: [↑ ROLL]   [↑ ROLL]

DISPLAY: $1.$ → z
          $2.$ → y
          $3.$ → x

$x \rightleftarrows y$ Exchanges the contents of the X and Y registers without affecting the contents of Z.

EXAMPLE:

PRESS: 1   ↑   2   ↑   3

DISPLAY: $1.$ → z
          $2.$ → y
          $3.$ → x

PRESS: $x \rightleftarrows y$

DISPLAY: $1.$ → z
          $3.$ → y
          $2.$ → x

$x \rightleftarrows y$

$z \longrightarrow Z$
$y \diagdown \diagup Y$
$x \diagup \diagdown X$
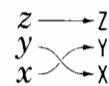
ARITHMETIC KEYS

The arithmetic keys can be divided into two groups, each of four keys: +, −, X, ÷, which operate on the numbers in the X and Y registers, and $\sqrt{x}$, x^2, $1/x$ and INT X, which operate directly on the number in the X-register.

These four keys perform the indicated arithmetic operations upon two numbers, the one in the Y-register (the first operand) and the other in the X-register (the second operand). The result of each operation is substituted for the number in the Y-register. The numbers in the X- and Z-registers remain unchanged.

NOTE

'First operand' and 'Second operand' are used here in the sense that the second operand is always considered to alter the first operand in order to produce a result. For example:

$$a - b = R$$
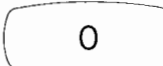
The second operand (b) is subtracted from the first operand (a) to produce the result (R).

All four keys are also used with the data-storage keys to perform operations on data stored in the memory. This use is explained under 'Data Storage' later in this chapter.

Adds the number in the X-register to the number in the Y-register. The sum appears in Y; the X and Z registers remain unchanged.

EXAMPLE:

$8 + 4 = 12$

PRESS: FIX( )   0

(If contents of X are unterminated, press 'CLEAR x').

PRESS: 8   ↑   4   +

DISPLAY: original contents of y → z

       $12.$ → y

       $4.$ → x

ARITHMETIC KEYS

− Subtracts the number in the X-register from the number in the Y-register. The difference appears in Y; the X and Z registers remain unchanged.

EXAMPLE 1:

17 − 9 = 8

PRESS: [1] [7] [↑] [9] [−]

DISPLAY: original contents of y → z
 8.   → y
 9.   → x

EXAMPLE 2:

47 − 19 − 8 + 5 = 25
(contents of Z to remain unchanged)

PRESS: [4] [7] [$x \rightleftarrows y$] [1] [9] [−]

PRESS: [8] [−] [5] [+]

DISPLAY: original contents of z → z
 25.   → y
 5.   → x

× Multiplies the number in the Y-register by the number in the X-register. The product appears in Y; the X and Z registers remain unchanged.

EXAMPLE 1:

$[(3 \times 4) - 6] \times 8 = 48$

PRESS: [3] [↑] [4] [×]

(continued)

$+$

$-$

$\times$

$\div$

(continued)

ARITHMETIC KEYS

PRESS: [6] [$-$] [8] [$\times$]

DISPLAY: original contents of y $\longrightarrow$ z

$48.$ $\longrightarrow$ y

$8.$ $\longrightarrow$ x

EXAMPLE 2:

$5^3 = 5 \times 5 \times 5 = 125$

PRESS: [5] [↑] [$\times$] [$\times$]

DISPLAY: $125.$ $\longrightarrow$ y

$\div$ **Divides the number in the Y-register by the number in the X-register. The quotient appears in Y; the X and Z registers remain unchanged.**

EXAMPLE:

$36 \div 9 = 4$

PRESS: [3] [6] [↑] [9] [$\div$]

DISPLAY: original contents of y $\longrightarrow$ z

$4.$ $\longrightarrow$ y

$9.$ $\longrightarrow$ x

The following example includes the use of all four keys; the calculation is performed by two methods to show how steps can be saved.

EXAMPLE:

$$\frac{(3 \times 4) + (8 - 9)}{(8 \times 2) - 6} = 1.1$$

Method I.

a. Starting with the numerator, solve separately for the quantities in parentheses, then add them and store the result;

b. solve the denominator in the same way;

c. recall the numerator and divide.

ARITHMETIC KEYS

PRESS: [FIX ( )] (1)

Then press the keys in the step sequence given below.

STEP	KEY	DISPLAY REGISTERS			NOTES
		X	Y	Z	
1	CLEAR	0	0	0	--
2	3	3	0	0	--
3	↑	3	3	0	--
4	4	4	3	0	--
5	×	4	12	0	$(3 \times 4) \rightarrow y$
6	ROLL ↑	0	4	12	store in Z
7	8	8	4	12	--
8	$x \rightleftarrows y$	4	8	12	--
9	9	9	8	12	--
10	−	9	−1	12	$(8 - 9) \rightarrow y$
11	↓	−1	12	12	recall (3×4) to Y
12	+	−1	11	12	$(3 \times 4) + (8 - 9) \rightarrow y$
13	ROLL ↑	12	−1	11	store in Z
14	8	8	−1	11	--
15	$x \rightleftarrows y$	−1	8	11	--
16	2	2	8	11	--
17	×	2	16	11	$(8 \times 2) \rightarrow y$
18	6	6	16	11	--
19	−	6	10	11	$(8 \times 2) - 6 \rightarrow y$
20	↓	10	11	11	recall $(3 \times 4) + (8 - 9)$ to Y
21	÷	10	1.1	11	Result = 1.1 $\rightarrow y$

This 'program' can be shortened by several steps without changing the procedure:

1. The CLEAR key is, in most cases, an unnecessary step;

2. step 6 is unnecessary if step 8 is changed to ↑:

STEP	KEY	X	Y	Z
5	×	4	12	0
6				
7	8	8	12	0
8	↑	8	8	12
9	9	9	8	12

The contents of the registers is the same, in both cases, at step 9 and one step has been saved. Similarly, step 13 can be deleted and step 15 changed to ↑.

+

−

×

÷

(continued)

ARITHMETIC KEYS

Method II.

If the problem is solved by separately adding (+)8 and (−)9 in the numerator, then more steps can be saved. This approach changes the total number of steps from 21 to 16 (17 if CLEAR has to be used).

$$\frac{(3 \times 4) + (8 - 9)}{(8 \times 2) - 6} = 1.1$$

STEP	KEY	DISPLAY REGISTERS			NOTES
		X	Y	Z	
1	3	3	·	·	--
2	↑	3	3	·	--
3	4	4	3	·	--
4	×	4	12	·	$(3 \times 4) \rightarrow y$
5	8	8	12	·	--
6	+	8	20	·	$(3 \times 4) + 8 \rightarrow y$
7	9	9	20	·	--
8	−	9	11	·	$(3 \times 4) + (8 - 9) \rightarrow y$
9	8	8	11	·	--
10	↑	8	8	11	$(3 \times 4) + (8 - 9)$ stored in Z
11	2	2	8	11	--
12	×	2	16	11	$(8 \times 2) \rightarrow y$
13	6	6	16	11	--
14	−	6	10	11	$(8 \times 2) - 6 \rightarrow y$
15	↓	10	11	11	recall $(3 \times 4) + (8 - 9)$ to Y
16	÷	10	1.1	11	Result = 1.1 $\rightarrow y$

The following example illustrates an accumulative process using the arithmetic keys; the ↑ and ↓ keys allow the partial totals to be temporarily stored in, and recalled from, the Z-register.

The sum of the products, $n_1 n_2 + n_3 n_4 + n_5 n_6 + \ldots$ etc., is used in this example; however, this 'program' is particularly important as it can be easily adapted to solve other expressions (such as product of the sums) as will be explained following the program.

ARITHMETIC KEYS

STEP	KEY	NOTES
1	Enter n_1	use any digits for n
2	↑	--
3	Enter n_2	--
4	×	$(n_1 \, n_2) \rightarrow y$
5	Enter n_3	--
6	↑	$(n_1 \, n_2)$ stored in Z
7	Enter n_4	--
8	×	$(n_3 \, n_4) \rightarrow y$
9	↓	recall $(n_1 \, n_2)$ to Y
10	+	$(n_1 \, n_2) + (n_3 \, n_4) \rightarrow y$
11	Enter n_5	--
12	↑	$(n_1 \, n_2) + (n_3 \, n_4)$ stored in Z
13	Enter n_6	--
14	×	$(n_5 \, n_6)$
15	↓	recall $(n_1 \, n_2) + (n_3 \, n_4)$ to Y
16	+	$(n_1 \, n_2) + (n_3 \, n_4) + (n_5 \, n_6) \rightarrow y$
17	Enter n_7	
----etc.----		

Notice that, after the initial quantities have been entered and multiplied at step 4, the step sequence is repetitive, steps 5 through 10, 11 through 16 and so on.

The program can be adapted to solve other expressions by changing the arithmetic instructions; for example, if the × and + keys are interchanged, then the product of the sums is solved $(n_1 + n_2) \, (n_3 + n_4) \, (n_5 + n_6) \ldots$ etc.

If the × keys are changed to ÷, then the sum of the quotients is solved:

$$\frac{n_1}{n_2} + \frac{n_3}{n_4} + \frac{n_5}{n_6} = \ldots \text{ etc.}$$

More complex expressions, such as:

$$\frac{n_1 + n_2}{n_3} + \frac{n_4 + n_5}{n_6}$$

can also be solved using this general program form; however, in these cases, additional steps are required.

ARITHMETIC KEYS

$\sqrt{x}$ x^2 $\frac{1}{x}$ int x
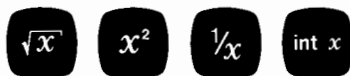
These four keys perform the indicated operations directly on the number in the X-register without affecting either the Y- or Z-registers. The result appears in X.

$\sqrt{x}$ Calculates the square-root of the contents of the X-register; the result appears in X.

EXAMPLE:

$$\sqrt{9} = 3$$

PRESS: FIX( )   2

PRESS: 9 $\sqrt{x}$

DISPLAY: 3.00 → x

x^2 Calculates the square of the contents of the X-register; the result appears in X.

EXAMPLE 1:

$$4^2 = 16$$

PRESS: 4 x^2

DISPLAY: 16.00 → x

EXAMPLE 2:

$$5^4 = (5^2)^2 = 625$$

PRESS: 5 x^2 x^2

DISPLAY: 625.00 → x

(see the X key for raising numbers to odd powers.)

ARITHMETIC KEYS

EXAMPLE 3:

$$x = \sqrt{a^2 + b^2}, \quad \sqrt{3^2 + 4^2} = 5$$

PRESS: (3) (x^2) (↑) (4) (x^2) (+)

PRESS: (↓) ($\sqrt{x}$)

DISPLAY: *5.00*   → x

EXAMPLE 4:

Area of a circle is πr^2, where r is the radius; if r = 6.5 units, then area = 132.73 square units.

PRESS: (6) (·) (5) (x^2) (↑) (π) (×)

DISPLAY: *132.73*   → y

($1/x$) **Calculates the reciprocal of the contents of the X-register; the result appears in X.**

EXAMPLE 1:

$$\frac{1}{9.8} = .10204$$

PRESS: (FIX ( )) (5)

PRESS: (9) (·) (8) ($1/x$)

DISPLAY: *0.10204*   → x

$\sqrt{x}$

x^2

$\frac{1}{x}$

int x

(continued)

ARITHMETIC KEYS

EXAMPLE 2:

$$x^{-n} = \frac{1}{x^n}$$

$$5^{-4} = \frac{1}{5^4} = 0.0016$$

PRESS: (5) (x^2) (x^2) ($\frac{1}{x}$)

DISPLAY: *0.00160* ⟶ x

int x Removes the fractional part of the number in the X-register without affecting the sign or the integer part.

EXAMPLE 1:

integer of −5.9 = −5

PRESS: FIX ( ) (1)

PRESS: (CHG SIGN) (5) (•) (9)

DISPLAY: *-5.9* ⟶ x

PRESS: (int x)

DISPLAY: *-5.0* ⟶ x

PRESS: FIX ( ) (0)

ARITHMETIC KEYS

Repeat the above example [except do not press 'FIX (1)']; notice that, when −5.9 is entered, the displayed number is rounded to −6. The final result (−5) is still correct, however, because the choice of the display does not affect accuracy.

EXAMPLE 2:

Convert 5.72° to degrees (°) and minutes (')
5.72° = 5° 43.2'

PRESS: [FIX( )] [2]

Enter 5.72:

PRESS: [5] [•] [7] [2] [↑]

Separate degrees from minutes (fractional part):

PRESS: [int x] [—]

Transfer degrees to the Z-register

PRESS: [x⇄y] [↑]

Convert the fractional part to minutes by multiplying by 60:

PRESS: [6] [0] [×]

DISPLAY: (°) 5.00 → z
(') 43.20 → y

**DATA
REGISTERS**

DATA STORAGE KEYS

Apart from the X, Y and Z display registers, the basic calculator contains a further 51 registers which are similar in form to the display registers; these registers are used for data storage. One complete data number can be stored in any register; it can then be recalled at any time for use in a calculation.

NOTE

Data storage memory is entirely separate from the program memory; storing data does not affect the space available for storing program steps.

Forty-nine of these registers are named (addressed) 000 through 048; the remaining two are special registers addressed as a and b (not to be confused with the alpha characters A and B).

An optional add-on feature (-hp- 11216A) allows the storage memory to be increased by a further 60 storage registers (049 through 108) giving a total of 111 registers. Option 001 calculators have these additional registers installed. The three-digit addressing enables the extra memory to be added without changing any operating procedures; the extra registers are used in exactly the same way as the original registers in the basic calculator.

Data storage registers may be used in any order; each register is selected from the keyboard (or program) by pressing the keys corresponding to the address. Depending upon the operation to be performed, certain keys must precede the address.

NOTE

It is possible to select a register whose address contains leading zeros without using all three digits (e.g. '28' for register number 028); this is explained under 'Short-form addressing', at the end of this section.

If an error in addressing is made so that a non-existent register is selected, the STATUS light lights and the calculator stops without performing any operation on the memory (see Page 3-11). Non-existent registers are 049 and above for the basic calculator or 109 and above if the extra memory is installed (see Indirect Register-Addressing on Page 4-35).

DATA STORAGE KEYS

> **NOTE**
>
> Throughout the remainder of this section storage registers are referred to as, for example, 'reg. 032' or 'reg. a', etc. The contents of a particular storage register are referred to by the register name appearing in parentheses; for example, '(reg. 012)' means 'the data number stored in, or contents of, reg. 012'.

When the calculator is switched on, the data registers are automatically cleared. However, it is not always necessary to clear a storage register before storing a number because the new data automatically substitutes for the old data. The CLEAR key clears both the a and b registers without affecting any other storage register.

These keys provide access to the data storage registers. They are 'initializer' keys in that they do nothing by themselves but must be followed by other keystrokes before any operation is performed.

There are four operations involving the data-storage registers:

1. Direct data storage and recall.
2. Direct storage-register arithmetic.
3. Indirect data storage and recall.
4. Indirect storage-register arithmetic.

Each of these operations is described, in turn, in the following pages.

For 'direct' operations the keying sequence selects the required register by means of a 'direct' address; for example,
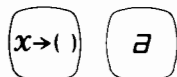
stores the contents of the X-register into reg. a. Here a is the 'direct' address.

**DATA
REGISTERS**
(continued)

DATA STORAGE KEYS

For 'indirect' operations the keying sequence selects the required register via an 'indirect' address; for example,

$$x \rightarrow (\ ) \quad \boxed{\substack{\text{IN-}\\\text{DIRECT}}} \quad \boxed{a}$$

stores the contents of the X-register into the register specified by the contents of reg. a. Here, a is the 'indirect' address and the contents of reg. a select the 'direct' address.

Both 'direct' and 'indirect' operations can be performed either from the keyboard or in a program; however, in general, 'indirect' operations are program-oriented rather than keyboard-oriented operations.

**DIRECT STORAGE
& RECALL**

Direct storage and recall enables displayed numbers to be stored in the memory and enables stored numbers to be recalled to the display. The direction of the arrows on the keys indicates in which direction the data is to move.

$\boxed{x \rightarrow (\ )}$ Stores the contents of the X-register into the storage register selected by the next key(s) pressed: a or b or 000 through 048 (through 108 in Option 001 calculators).

The contents of the X-register remain unchanged.

EXAMPLE 1:

Store π in the b register.

PRESS: $\boxed{\pi}$ $\boxed{x \rightarrow (\ )}$ $\boxed{b}$

RESULT: $\pi \longrightarrow$ reg. b

EXAMPLE 2:

Store π in reg. 027.

PRESS: $\boxed{\pi}$ $\boxed{x \rightarrow (\ )}$ $\boxed{0}$ $\boxed{2}$ $\boxed{7}$

RESULT: $\pi \longrightarrow$ reg. 027

DATA STORAGE KEYS

y→( ) Stores the contents of the Y-register into the storage register selected by the next key(s) pressed: a or b or 000 through 048 (through 108 in Option 001 Calculators).

The contents of the Y-register remain unchanged.

No example is given as the y → ( ) key stores the contents of Y in exactly the same way as the x → ( ) key stores the contents of X.

x←( ) Recalls, to the X-register, the contents of the storage register selected by the next keys pressed: 000 through 048 (through 108 in Option 001 Calculators).

The contents of the recalled register remain unchanged. It is not necessary to use the x ← ( ) key when recalling data to X from the a or b registers; simply press those keys to recall the data, from the selected register, to X.

EXAMPLE 1:

Recall π from reg. b (stored previously).

PRESS: [CLEAR x]   This is not a necessary operation; it is included only to demonstrate that π is recalled.

PRESS: [b]

DISPLAY: π → x
(π also remains in reg. b)

EXAMPLE 2:

Recall π from reg. 027 (stored previously).

PRESS: [CLEAR x]

PRESS: [x←( )] [0] [2] [7]

DISPLAY: π → x
(π also remains in reg. 027)

DATA STORAGE KEYS

$\boxed{y\vec{\leftarrow}(\ )}$ Exchanges the contents of the Y-register with the contents of the storage register selected by the next key(s) pressed: a or b or 000 through 048 (through 108 in Option 001 Calculators).

EXAMPLE:

Enter any number into the Y-register and exchange with the contents of reg. 027 (π was stored in reg. 027 in an earlier example).

PRESS: $\boxed{6}$ $\boxed{\uparrow}$

PRESS: $\boxed{y\vec{\leftarrow}(\ )}$ $\boxed{0}$ $\boxed{2}$ $\boxed{7}$

RESULT: 6 $\rightarrow$ reg. 027
             (reg. 027) $\rightarrow$ y

The $y\rightleftarrows(\ )$ key is particularly useful, especially in a program, for moving data from one memory location to another.

PRESS: $\boxed{y\vec{\leftarrow}(\ )}$ $\boxed{a}$

PRESS: $\boxed{y\vec{\leftarrow}(\ )}$ $\boxed{0}$ $\boxed{3}$ $\boxed{2}$

PRESS: $\boxed{y\vec{\leftarrow}(\ )}$ $\boxed{0}$ $\boxed{3}$ $\boxed{3}$

PRESS: $\boxed{y\vec{\leftarrow}(\ )}$ $\boxed{0}$ $\boxed{1}$ $\boxed{6}$

RESULT: (y) $\rightarrow$ reg. a
        (reg. a) $\rightarrow$ reg. 032
        (reg. 032) $\rightarrow$ reg. 033
        (reg. 033) $\rightarrow$ reg. 016
        (reg. 016) $\rightarrow$ y

$\boxed{a}$ $\boxed{b}$

Used, when storing and recalling data, to select reg. a or reg. b, respectively.

DATA STORAGE KEYS

STORAGE: see $\boxed{y\xleftarrow{\ }( )}$, $\boxed{x\rightarrow( )}$ and $\boxed{y\rightarrow( )}$.

RECALL: Press either key to recall the contents of the selected register to X (the contents of the recalled register remain unchanged).

The 'CLEAR' key clears both of the a and b registers.

The a and b registers are particularly useful as they require a minimum of keystrokes for storage and recall.

NOTE

If the Mathematics ROM (-hp- 11210A) is installed, the a and b registers serve a special function, in conjunction with ACC +, ACC −, and RCL keys. This enables minimal keystroke operation of vector arithmetic and manipulation of complex numbers (i.e. using the 'i' operator, where $i = j = \sqrt{-1}$).

The following example illustrates a typical use of the data storage memory. For simplicity, only reg. a is used; however, the technique illustrated is applicable to all data storage registers.

EXAMPLE:

Multiplication by a constant;

a series of numbers (n_1, n_2, n_3, . . . etc.) are to be multiplied by a constant (k).

Let k = 1.684, n_1 = 3, n_2 = 11.2, n_3 = . . . etc.

The value for k is first stored in memory so that it will not be necessary to reenter the constant for each calculation:

PRESS: $\boxed{\text{FIX ( )}}$ $\boxed{4}$

Enter k and store in reg. a:

PRESS: $\boxed{1}$ $\boxed{\cdot}$ $\boxed{6}$ $\boxed{8}$ $\boxed{4}$ $\boxed{x\rightarrow( )}$ $\boxed{a}$

Enter n_1, recall k and multiply:

PRESS: $\boxed{3}$ $\boxed{\uparrow}$ $\boxed{a}$ $\boxed{\times}$

DISPLAY: (kn_1) 5.0520 $\rightarrow y$

(continued)

DIRECT STORAGE
& RECALL
(continued)

DATA STORAGE KEYS

Enter n_2, recall k and multiply:

PRESS: (1) (1) (·) (2) (↑) (a) (×)

DISPLAY: (kn_2) 18.8608 → y

Enter n_3 . . . . .etc.

DIRECT
STORAGE-REGISTER
ARITHMETIC

Direct storage-register arithmetic enables arithmetic operations to be performed using a number stored in the memory without first recalling that number to the display. Four arithmetic keys (+, −, ×, ÷) are used in conjunction with the storage and recall keys. In addition to indicating where the results of the operation will be stored, the arrow on each storage or recall key also points to the first operand (defined in the note on Page 4-14).
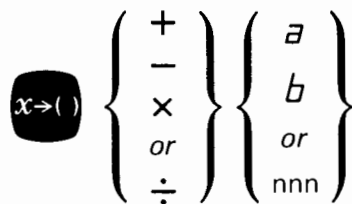
$$ \boxed{x\to(\ )} \left\{ \begin{array}{c} + \\ - \\ \times \\ or \\ \div \end{array} \right\} \left\{ \begin{array}{c} a \\ b \\ or \\ nnn \end{array} \right\} $$

Performs the indicated arithmetic operation between the number in the X-register (the second operand) and the number in the selected register (the first operand).

The result is stored in the selected register and the X-register remains unchanged.

EXAMPLE 1:

Subtract the contents of the X-register from the contents of reg. b.

PRESS: ($x\to(\ )$) (−) (b)

RESULT: (reg. b) − (x) → reg. b

DATA STORAGE KEYS

EXAMPLE 2:

Divide the contents of reg. 042 by the contents of the X-register.

PRESS: [x→( )] [÷] [0] [4] [2]

RESULT: (reg. 042) ÷ (x) ➞ reg. 042

$$[y→( )] \left\{ \begin{array}{c} + \\ - \\ \times \\ or \\ \div \end{array} \right\} \left\{ \begin{array}{c} a \\ b \\ or \\ nnn \end{array} \right\}$$
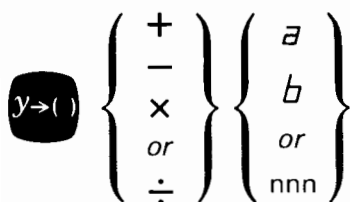
Performs the indicated arithmetic operation between the number in the Y-register (the second operand) and the number in the selected register (the first operand).

The result is stored in the selected register and the Y-register remains unchanged.

EXAMPLE:

Add the contents of the Y-register to the contents of reg. 038.

PRESS: [y→( )] [+] [0] [3] [8]

RESULT: (reg. 038) + (y) ➞ reg. 038

DATA STORAGE KEYS
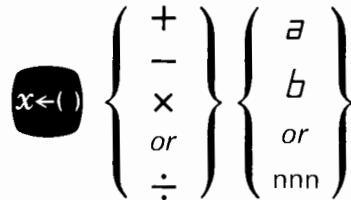
$$x{\leftarrow}(\ )\left\{\begin{matrix}+\\-\\\times\\or\\\div\end{matrix}\right\}\left\{\begin{matrix}a\\b\\or\\nnn\end{matrix}\right\}$$

Performs the indicated arithmetic operation between the number in the selected register (the second operand) and the number in the X-register (the first operand).

The result appears in the X-register and the number in the selected register remains unchanged.

EXAMPLE:

Divide the number in the X-register by the number in reg. a.

PRESS: $x{\leftarrow}(\ )$ $\div$ a

RESULT: $(x) \div (\text{reg. } a) \longrightarrow X$

$$y{\vec{\leftarrow}}(\ )\left\{\begin{matrix}+\\-\\\times\\or\\\div\end{matrix}\right\}\left\{\begin{matrix}a\\b\\or\\nnn\end{matrix}\right\}$$
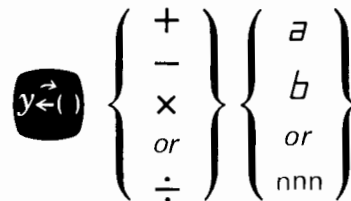
Performs the indicated arithmetic operation between the number in the selected storage register (the second operand) and the number in the Y-register (the first operand).

The result appears in the Y-register and the number in the selected register remains unchanged.

NOTE

In this application, the $y{\vec{\leftarrow}}(\ )$ key does not have the 'exchange' connotation (Page 4-28). The bold arrow, pointing to the left on the key, indicates that, when performing register arithmetic, the key works in the same fashion as the $x \leftarrow (\ )$ key.

DATA STORAGE KEYS

EXAMPLE:

Subtract the number in reg. 039 from the number in the Y-register.

PRESS: $y\overset{\rightarrow}{\leftarrow}(\ )$ $-$ 0 3 9

RESULT: $(y) - (reg.\ 039) \rightarrow y$

**INDIRECT
STORAGE & RECALL**

$\left.\begin{array}{c} x{\rightarrow}(\ ) \\ y{\rightarrow}(\ ) \\ x{\leftarrow}(\ ) \\ or \\ y\overset{\rightarrow}{\leftarrow}(\ ) \end{array}\right\}$ $\boxed{\substack{IN-\\DIRECT}}$ $\left\{\begin{array}{c} a \\ b \\ or \\ nnn \end{array}\right\}$
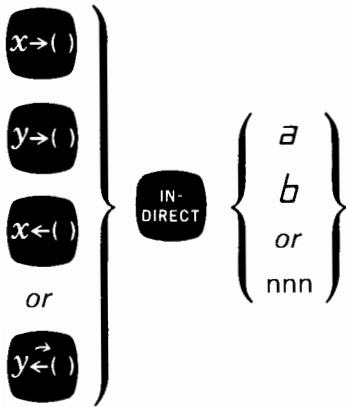
Indirect storage and recall is the same as direct storage and recall except that the required register is selected via an indirect address. The keying sequence uses the INDIRECT key to select the indirect register-address; the contents of that register then select the required direct register.

NOTE

Care should be taken to ensure that the indirect register contains a suitable number, otherwise a non-existent register may be designated. Proper register designations are discussed in detail under 'Indirect register addressing' on Page 4-35.

EXAMPLE 1:

Store the contents of the X-register into the register designated by the contents of reg. 038.

Let (reg. 038) = 22

PRESS: $x{\rightarrow}(\ )$ $\boxed{\substack{IN-\\DIRECT}}$ 0 3 8

(continued)

DATA STORAGE KEYS

RESULT: (x) → reg. 022

The contents of the X-register and of reg. 038 remain unchanged.

EXAMPLE 2:

Exchange the contents of the Y-register with the contents of the register designated by the contents of reg. a.

Let (reg. a) = 3

PRESS: [$\overrightarrow{y \leftarrow ()}$] [IN-DIRECT] [$a$]

RESULT: (y) → reg. 003
(reg. 003) → y

The contents of reg. a remain unchanged.

**INDIRECT
STORAGE-REGISTER
ARITHMETIC**

[$x \rightarrow ()$]
[$y \rightarrow ()$]
[$x \leftarrow ()$]
or
[$\overrightarrow{y \leftarrow ()}$]

[IN-DIRECT] $\left\{ \begin{array}{c} + \\ - \\ \times \\ or \\ \div \end{array} \right\}$ $\left\{ \begin{array}{c} a \\ b \\ or \\ nnn \end{array} \right\}$
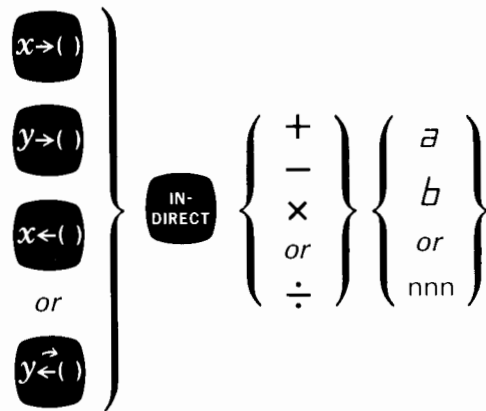
Indirect storage-register arithmetic is the same as direct storage-register arithmetic, except that the required register is selected via an indirect address. The keying sequence uses the INDIRECT key to select the indirect register-address; the contents of that register then select the required direct register. The INDIRECT key may be used either before the arithmetic key (as shown above) or after the arithmetic key; the result is the same in either case.

See the NOTE under "Indirect Storage and Recall" on Page 4-33.

DATA STORAGE KEYS

EXAMPLE 1:

Multiply the number, stored in the register designated by the contents of reg. 014, by the contents of the Y-register.

Let (reg. 014) = 19

PRESS: $\boxed{y\rightarrow(\ )}$ $\boxed{\substack{\text{IN-}\\\text{DIRECT}}}$ $\boxed{\times}$ $\boxed{0}$ $\boxed{1}$ $\boxed{4}$

RESULT: (reg. 019) $\times$ (y) $\longrightarrow$ reg. 019

The contents of the Y-register and of reg. 014 remain unchanged.

EXAMPLE 2:

Divide the number in the X-register by the number stored in the register designated by the contents of reg. b.

Let (reg. b) = 8

PRESS: $\boxed{x\leftarrow(\ )}$ $\boxed{\substack{\text{IN-}\\\text{DIRECT}}}$ $\boxed{\div}$ $\boxed{b}$

RESULT: (x) $\div$ (reg. 008) $\longrightarrow$ x

The contents of reg. 008 and of reg. b remain unchanged.

When indirectly addressing a storage register, care must be taken to ensure that the indirect register contains a proper address to designate the required register.

INDIRECT REGISTER-ADDRESSING

Proper addresses are numbers 0 through 48 for the basic calculator or 0 through 108 in Option 001 calculators.

  a. The address may be negative; the sign will be ignored.

  b. The address need not be an integer, the fractional part of the number (considered as a fixed decimal point number) will be ignored.

Examples of proper addresses:

1	$\longrightarrow$ designates reg. 001
−6	$\longrightarrow$ designates reg. 006
28.368	$\longrightarrow$ designates reg. 028
1.02368×10^2	$\longrightarrow$ designates reg. 102
3.642×10^{-2}	$\longrightarrow$ designates reg. 000

DATA STORAGE KEYS

Improper addresses are:

a. Numbers with an absolute value greater than 48.99 . . . . . 9 in the basic calculator.

b. Numbers with an absolute value greater than 108.99 . . . . . 9 in Option 001 calculators.

The *a* and *b* registers cannot themselves be addressed indirectly, but they can be used as indirect addresses to designate other (numeric) direct addresses.

**SHORT-FORM
ADDRESSING**

A short-form address consists of a register-address with all leading zeros dropped; '3' for reg. 003, '38' for reg. 038, etc.

NOTE

This information is also applicable to the 4-digit addresses used by the program memory (described in Chapter 5).

The short-form address may be used either from the keyboard or in a program; however, the address must be properly terminated. For example, consider this keying sequence:

| x→( ) | 0 | 0 | 3 |

as soon as the '3' is pressed the address is automatically terminated and the operation is performed (in this case, the number in X is stored in reg. 003). This sequence may be shortened to:

| x→( ) | 3 |

This time, when '3' is pressed, the address remains unterminated because the calculator has no way of deciding whether the address is selecting reg. 003 or, say, 038. The calculator, therefore, will do nothing until the address is terminated.

Any key, except the digit keys and CONTINUE (given from the keyboard), on the basic calculator, can be used to terminate the address; however, the terminating key will also be executed.

DATA STORAGE KEYS

For example:

$x \rightarrow (\ )$ 3   ↑

will store the contents of X in reg. 003 and then execute the 'up' instruction.

Similarly,

$x \rightarrow (\ )$ 3   8 a

will store the contents of X in reg. 038 and then recall the contents of reg. a to X.

If no subsequent operation is desired when the address is being made from the keyboard, then the STOP key can be used:

$x \rightarrow (\ )$ 3   STOP

In this case, the STOP acts as a 'no operation' but still terminates the address so that the number is stored.

In general, there is little to be gained by using this short-form addressing from the keyboard; the beginner, who may be confused by the results, is advised not to use it, either from the keyboard or in a program.

On the other hand, short-form addressing in a program does offer some advantages; using this method of addressing in complex programs saves considerable space in the program memory. This has the effect of enabling more program steps to be stored.

Indirect data-storage addresses can be nested, to any depth, by repeating the INDIRECT key the required number of times.

NESTING INDIRECT ADDRESSES

EXAMPLE:

Let (reg. a) = 1
and (reg. 001) = 2

PRESS: $x \rightarrow (\ )$ IN-DIRECT   IN-DIRECT a

RESULT: $(x) \rightarrow$ reg. 002

NOTES

INTRODUCTION TO PROGRAMMING

This chapter describes the programming keys (those contained in the extreme right-hand block of the keyboard) and explains how to program the calculator. The keys described in the preceding chapter operate as before, whether used from the keyboard or as steps in a program.

A program is a sequence of instructions telling the calculator what it must do to solve a particular problem. The calculator can 'remember' the steps of the program and can execute them any number of times. The rules for programming the calculator are relatively simple because the program steps are the same as the keyboard operations; no special language need be learned. Difficulties encountered in writing a program are more likely to result from the complexity of the particular program than from the application of the rules.

The calculator cannot interpret the programmer's requirements, it can only obey instructions; it is essential, therefore, that the programmer knows exactly what each key does and that he be exact in giving instructions to the calculator. Despite the 'exactness' requirement, it is not necessary to be unduly concerned about making errors in a program; the design of the calculator is such that errors are usually easy to recognize and correct, even after the program has been loaded into the calculator.

As stated earlier, a program is a sequence of instructions to the calculator. In the preceding section of this manual, whenever an example was to be performed, you, the operator, were 'programmed'. You were asked to press keys in a given sequence to obtain a particular result; (in most cases) if the sequence was not followed exactly the result was not correct. In a similar manner, you, the programmer, must now give a sequence of (correct) instructions to the calculator.

As an example, try the following:

> Enter three numbers (A, B, and C) into the calculator so that A appears in the Z register, B in the Y register and C in X; choose simple numbers, say A = 6, B = 5 and C = 3.

The display should now be:

$$
\begin{array}{lll}
\text{(A)} & 6. & \rightarrow z \\
\text{(B)} & 5. & \rightarrow y \\
\text{(C)} & 3. & \rightarrow x
\end{array}
$$

Next press whatever keys are necessary to solve (A × B)/C, in this case (6 × 5)/3. As you press each key, write it in the KEY column in the table below (starting in step 0000) and also make a note, in the appropriate columns, of the effect on the three display registers. Do not include entering A, B and C as program steps, these will be entered each time before the program is run.

INTRODUCTION TO PROGRAMMING

STEP	KEY	DISPLAY		
		X	Y	Z
Enter the numbers		C(=3)	B(=5)	A(=6)
0000				
0001				
0002				
0003				
0004				
0005				
0006				
0007				
0008				
etc.				

If you now have the correct result of the calculation (i.e. 10) in one of the display registers, then you have just written a useable program. It may not necessarily be the shortest or most efficient program, but it will still work in the calculator for any values of A, B and C. The program is complete except that you should add an END as the last step; END instructs the calculator to stop running the program because the calculation is finished. Two (of several possible) programs to solve (A X B)/C are shown on Page 5-3.

Here is the procedure to load and run your program or either one of the example programs.

    To load the program:

        PRESS: ⌈ RUN ⌉ (END)

        PRESS: ⌈ PRGM ⌉

        PRESS: The keys in the step sequence of the program . . . . .
                for program II, for example, this would be

                (÷) (↓) (×) (END)

        PRESS: ⌈ RUN ⌉

    To run (and rerun) the program:

        PRESS: (END)

INTRODUCTION TO PROGRAMMING

PRESS: digit keys to enter A, B and C into Z, Y and X respectively; e.g.

$$\boxed{6} \quad \boxed{\uparrow} \quad \boxed{5} \quad \boxed{\uparrow} \quad \boxed{3}$$

PRESS: $\boxed{\text{CONTINUE}}$ the program runs and the answer appears in the display.

PROGRAM I

STEP	KEY	DISPLAY		
		X	Y	Z
Enter the numbers		C	B	A
0000	ROLL ↑	A	C	B
0001	ROLL ↑	B	A	C
0002	×	B	A x B	C
0003	↓	A x B	C	C
0004	$x \rightleftarrows y$	C	A x B	C
0005	÷	C	$\dfrac{A \times B}{C}$	C
0006	END	final display		

PROGRAM II

STEP	KEY	DISPLAY		
		X	Y	Z
Enter the numbers		C	B	A
0000	÷	C	$\dfrac{B}{C}$	A
0001	↓	$\dfrac{B}{C}$	A	A
0002	×	$\dfrac{B}{C}$	$\dfrac{A \times B}{C}$	A
0003	END	final display		

**WRITING A
PROGRAM**

INTRODUCTION TO PROGRAMMING

In general, program writing is composed of three major steps:

    a). Define the problem.
    b). Decide how the problem is to be solved.
    c). Write the steps for the calculator.

Example of program writing:

    a). Define the problem . . . what is the program supposed to do?

         Write a program to solve $(A \times B)/(A + B)$ for any values of A and B; when the program is complete, display the result and display A and B.

    b). Decide how the problem is to be solved . . . what steps would you take to solve the problem on paper?

         The easiest way to do this is by means of a 'flow chart'; the initial flow chart should be as simple as possible.

```
        ┌──────────────┐
        │  Multiply:   │
        │   A X B      │
        └──────┬───────┘
               │
               ▼
        ┌──────────────┐
        │    Add:      │
        │   A + B      │
        └──────┬───────┘
               │
               ▼
        ┌──────────────┐
        │   Divide:    │
        │   A X B      │
        │   ─────      │
        │   A + B      │
        └──────┬───────┘
               │
               ▼
        ┌──────────────┐
        │   Display    │
        │  the results │
        └──────────────┘
```
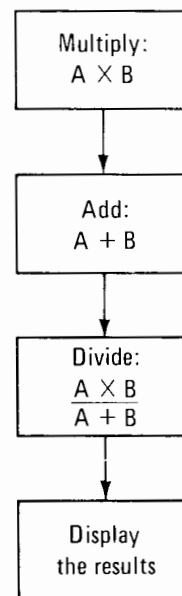
Next, draw the flow chart in greater detail adding specific notes such as where data is to be stored, etc. It may be necessary, if the problem is complex, to draw several versions of the flow chart. Here is the final version of the flow chart;

INTRODUCTION TO PROGRAMMING

Enter A and B manually

$$A \rightarrow y$$
$$B \rightarrow x$$

I
Store
A $\rightarrow a$
B $\rightarrow b$

II
Multiply:
A × B

III
Recall from
storage and
Add:
A + B

IV
Divide:
$$\frac{A \times B}{A + B}$$

V
Recall from
storage

VI
Stop and
display:
$$\frac{A \times B}{A + B} \rightarrow z$$
A $\rightarrow y$
B $\rightarrow x$
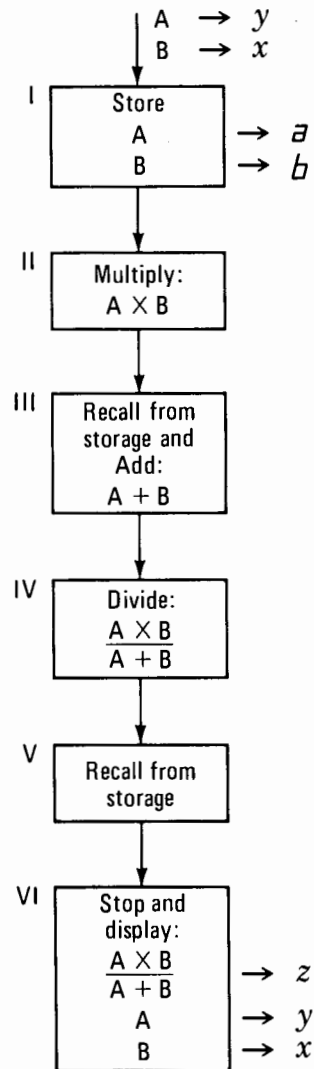
c). Write the steps for the calculator . . . what keys would you press to make the same calculation from the keyboard?

Write the steps exactly as the keys would be pressed if the operations were to be made manually; at each step note the effect of the operation on the display and storage registers (the program pad, supplied with each calculator, may be used for this). Operations such as data entry should not be included as steps of the program.

INTRODUCTION TO PROGRAMMING

Starting with I in the final flow chart, A and B, (currently in Y and X respectively) are stored in the memory:

STEP	KEY	DISPLAY			STORAGE	
		X	Y	Z	a	b
0000	$y\to(\ )$	B	A	-	-	-
0001	a	B	A	-	A	-
0002	$x\to(\ )$	B	A	-	A	-
0003	b	B	A	-	A	B

(II of the flow chart) Multiply A $\times$ B.

0004	$\times$	B	A x B	-	A	B

(III of the flow chart) recall from storage and add, A + B (only A need be recalled as B is already in X).

0005	$x\gets(\ )$	B	A x B	-	A	B
0006	$+$	B	A x B	-	A	B
0007	a	A + B	A x B	-	A	B

(IV of the flow chart) Divide, (A $\times$ B) $\div$ (A + B)

0008	$\div$	A + B	$\dfrac{A \times B}{A + B}$	-	A	B

(V and VI of the flow chart) recall A and B from storage and stop to display the result.

0009	a	A	$\dfrac{A \times B}{A + B}$	-	A	B
0010	$\uparrow$	A	A	$\dfrac{A \times B}{A + B}$	A	B
0011	b	B	A	$\dfrac{A \times B}{A + B}$	A	B
0012	END		final display			

INTRODUCTION TO PROGRAMMING

Not all programs will be as straight-forward as the preceding one; most programs will contain loops and branches (explained later) involving decision making by the calculator. However, the basic approach to program writing remains the same: define the problem, decide how the problem is to be solved and then write the steps for the calculator.

Here is the complete program; an extra column, titled 'Key Code', has been added because this program will be used later in this section to illustrate program operations. Use of the key codes is explained later in this chapter.

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE a	STORAGE b
0000	$y \rightarrow (\ )$	40	B	A	-	-	-
0001	a	13	B	A	-	A	-
0002	$x \rightarrow (\ )$	23	B	A	-	A	-
0003	b	14	B	A	-	A	B
0004	×	36	B	A x B	-	A	B
0005	$x \leftarrow (\ )$	67	B	A x B	-	A	B
0006	+	33	B	A x B	-	A	B
0007	a	13	A + B	A x B	-	A	B
0008	÷	35	A + B	$\dfrac{A \times B}{A + B}$	-	A	B
0009	a	13	A	$\dfrac{A \times B}{A + B}$	-	A	B
0010	↑	27	A	A	$\dfrac{A \times B}{A + B}$	A	B
0011	b	14	B	A	$\dfrac{A \times B}{A + B}$	A	B
0012	END	46	—— final display ——				

PROGRAM MEMORY

LOCATIONS

The program memory in the basic Model 10 Calculator contains 500 locations, each of which has a unique 4-digit address; addresses consist of the numbers 0000 through 0499. One step (one keystroke) of a program can be stored in each location.

Extra sections of program memory can be added at any time, increasing the original 500-step program memory in the basic calculator either to a total of 1012 steps (addressed as 0000 through 1011), or to a total of 2036 steps (0000 through 2035). The use of 4-digit addressing throughout the program memory enables the extra memory to be added without changing any operating procedures; the extra memory is used in exactly the same way as the original 500-step memory in the basic calculator. Option 002 calculators have 1012 program steps and Option 003 calculators have 2036.

NOTE

The program memory is entirely separate from the data-storage memory; storing data does not affect the space available for storing program steps (and vice-versa).

A program is loaded into the memory simply by pressing the required keys; it is not necessary to first clear the memory because the new program steps are automatically substituted for the old.

PROGRAM COUNTER

When a program is being either loaded or run, a mechanism known as the 'program counter' automatically steps sequentially through the memory. At any given time, the program counter contains the address of a memory location; that is, the counter 'points to' the location whose address it contains. As soon as some operation has been performed at that location (i.e. a program step either loaded or executed), the counter increments by 1 and points to the next location.

The program counter can be set by the user to point to any desired location; this operation is known as 'setting the program counter' or as 'addressing the memory'.

STEP SEQUENCE

When a program is to be loaded, the user selects the starting address by setting the program counter to point to the address of the location which is to contain the first step. Then, as the program is loaded, by pressing the appropriate keys, the steps are automatically stored in sequential locations, beginning at the address specified.

PROGRAM MEMORY

When the program is to be run, the user selects the starting address as before; the steps of the program are then executed in numerical order, except where the program includes branching instructions (Page 5-10).

Generally programs start at the beginning of memory, at step 0000, unless more than one program is in the memory at the same time.

There are two ways to select a location in memory; either by setting the program counter to a fixed address or by instructing the program counter to search for a 'label'. A fixed address is selected by keying the 4-digits of the address and preceding them by the 'GO TO' key;

ADDRESSING THE MEMORY

GO TO   0   2   6   5

sets the program counter to address 0265.

The label search, on the other hand, selects an address identified by the label's position in memory. A label consists of two keys, LABEL followed by any other programmable key (with the exception of END), for example:
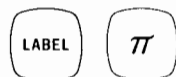
LABEL π

The operator can insert a label anywhere in the memory. Once the label is in memory, the address specified by that label can then be found by (in this case) the following keying sequence:

GO TO   LABEL π

This results in the program counter going to the beginning of the program memory (address 0000) and searching sequentially through the memory for the label (the keys 'LABEL, π' not preceded by a 'GO TO'). As soon as the π part of the label is found, the program counter points to the next address and this becomes the selected address, identified by that particular label.

BRANCHING

PROGRAM MEMORY

Branching instructions in a program cause the program counter to go to some location in memory other than the next sequential location; the program then continues, sequentially, from the new location. A branch can be either one of two types: 'conditional' or 'unconditional'.

If the branch is 'conditional' then the calculator makes the decision (based upon a specified condition) whether to branch or not; as a simplified example, the calculator can be programmed to ask "Is the number in X larger than the number in Y?" If the answer is NO, the program does not branch; if the answer is YES, the program counter branches to some other (specified) address and the program continues from there (conditional branching is fully described later in this chapter). If the branch is 'unconditional' then the calculator has no option, it must branch to the address specified in the program (see GO TO key).

SUBROUTINES

The Model 10 has subroutine capability. A subroutine is a sequence of program steps which is to be used many times, perhaps in several different programs, yet need be stored in only one place in the memory. A program may 'call for' (i.e. branch to) the subroutine at any specified time; when the steps of the subroutine have been executed, the program counter automatically returns to the 'calling' program, to the step following the program step which originally 'called' the subroutine. Use of subroutines saves not only considerable space in the program memory, but also considerable program-writing time. (See the SUB/RETURN key for complete information on the use of subroutines.)

**BLANKED
DISPLAY**

When the calculator is running a program, the display is blanked; as soon as the program stops, the display returns.

Many programs are short enough that the display will do no more than blink before reappearing; a long program which contains many branching instructions may leave the screen blanked for several seconds, or even much longer.

It is possible to write an error into a program which results in the calculator continually branching in a circle so that the display never returns; if this should occur, press the STOP key, the program will then stop and the display will return.

PROGRAM MEMORY

Each of the programmable keys has a two-digit key code; the code is shown in the Appendix at the back of this manual. When a program is being loaded, the code for each key pressed is stored in the program memory, one key-code per location.

The key codes are not required for program writing, however, the programmer should be aware of them because they are used for editing purposes (see Page 5-20). They are also printed by the printer when making a list of program steps (Chapter 6).

Those keys which are shown in the Appendix as having three-digit codes are not useable as program steps.

NOTE

The beginning programmer is advised not to use short-form addresses until he is familiar with the basic branching techniques.

The short-form address for the program memory is similar to that used for the data-storage memory (see Short-Form Addressing, Chapter 4) in that leading zeros may be dropped and an operational key used to terminate the address. Whether or not the terminating operation is executed, will depend upon whether the address came from the keyboard or from a program. For example, this sequence in a program causes the program counter to branch to address 0045 without executing the 'plus' (+) key:

(GO TO)   (4)   (5)   (+)

The same sequence given from the keyboard causes both the branch and execution of the 'plus' key.

When addressing from the keyboard, the address can be terminated by most keys; however, the terminating key will be executed, and this may result in some unwanted operation. It is acceptable to terminate the address by switching to the 'program' mode (i.e. pressing PRGM) but not by starting program execution (i.e. pressing CONTINUE). Pressing the STOP key will always terminate an address without producing unwanted results.

[RUN] [PRGM]

PROGRAM KEYS

The RUN and PRGM (program) keys select the calculator's mode of operation: the lights below the keys indicate the current mode.

PROGRAM MODE:
Used when loading program steps, from the keyboard, into the memory and when editing a program (i.e. verifying that the steps were loaded correctly).

Program mode enables program steps stored in the memory to be displayed; shown below is a typical display. Each display register contains an address (corresponding to a location in the program memory) followed by the key code for the program step currently stored at that address.

  a. The Y-register always contains the address of the location to which the program counter is currently pointing.

  b. The Z-register always contains the next lower-numbered address and the X-register always contains the next high-numbered address.

$$0006--------33 \quad \rightarrow \quad z$$
$$0007--------13 \quad \rightarrow \quad y$$
$$0008--------35 \quad \rightarrow \quad x$$

RUN MODE:
Used for all calculator operations except those listed under PROGRAM (above). 'Run' operations include all keyboard calculations, setting the program counter, running programs, all magnetic card operations, etc.

[GO TO]

[GO TO] This key, followed by an address, is used to set the program counter to the selected address in the program memory (see Page 5-9). For example, location 0138 is selected by:

[GO TO] [0] [1] [3] [8]

If the keying sequence is given from the keyboard, the program counter sets to the address designated and the calculator then waits for further keyboard instructions.

If the keying sequence is encountered as program steps, then an unconditional branch is made to the address indicated and the program step stored at that address is executed. The program then continues to run automatically from that address.

GO TO is also used in conjunction with the LABEL and SUB/RETURN keys to address the memory (see description of LABEL and SUB/RE-TURN).

PROGRAM KEYS

Labels (briefly described under Addressing the Memory, Page 5-9) are used to identify any desired location in memory. Labels must consist of two keys, LABEL followed by any other programmable key (except END):

LABEL ⬜ (any)

A specific label cannot be used to identify more than one location at any one time. If it is, then only the first (i.e. the lowest numbered location) will be valid.

Any number of different labels can be used at one time, limited to the number of keys available to follow the LABEL key.

GO TO LABEL ⬤ (any)

These keys instruct the program counter to search for a particular label. The search begins at address 0000 and continues sequentially through the memory until the label (i.e. 'LABEL, any' not preceded by a 'GO TO') is found. The search ends at the address immediately following the 'any' part of the label (the operation specified by 'any' is not executed).

If the instruction to search came from the keyboard, the calculator waits for the next key to be pressed. If the instruction to search came from the program then execution automatically continues at the new address.

For example:

STEP	KEY		STEP	KEY
0098	---		0362	---
0099	GO TO		0363	LABEL
0100	LABEL		0364	÷
0101	÷		0365	↑
0102	etc.		0366	etc.

After step (address) 0101 the search for the label starts (at address 0000); when the label is found, program execution continues with the 'up' instruction at address 0365. The 'divide' instruction in step 0101 constitutes part of the label-call and the one in step 0364 part of the label; in neither case does an actual divide operation occur.

Branching to a label offers considerable advantages over branching to a fixed address. Programs whose addresses are identified only by labels can be stored anywhere in the memory; they can also be moved easily because there are no addresses to be changed. Also, any time the program is to be corrected (i.e. steps changed, added or deleted) there are no fixed

LABEL

(continued)

PROGRAM KEYS

addresses to be checked in case they must now be changed as a result of the corrections.

The main disadvantage of using labels is that a search takes more time (depending upon the location of the label in memory) than does a branch to a fixed address. Usually, this will have no significance because, in this case, 'time' constitutes only a few thousandths of a second. If time is a significant factor, then the programmer may still take advantage of the labels; he can write his original program with labels and then, when the program has been completely checked, change the labels to the appropriate fixed addresses. Alternatively the labels can be located in the lower-numbered addresses.

CONTINUE

When given from the keyboard, starts automatic program execution beginning at the present address in the program memory. CONTINUE may be used in a program as a 'no operation' and as a termination for a short-form address. It cannot be used from the keyboard to terminate a short-form address.

STOP

Stops execution of the program.

If the STOP key is pressed while a program is running, then the program will stop after completing the current operation.

STOP's used as program steps enable the user to affect the results of a program by 'breaking in' at specified places. When the program is stopped, the user can enter data, or make any keyboard operations and then continue automatic execution of the program.

END

END should be the last (i.e. the highest numbered), and only the last, step in a program. It stops program execution and resets the program counter to address 0000.

When used from the keyboard, END is the equivalent of keying 'GO TO, 0, 0, 0, 0' and may, therefore, be used instead of that sequence.

During magnetic program card operations the recording and loading processes stop as soon as an END has been transferred (see Page 5-28).

END clears any subroutine return-address to which a return has not yet been made (Page 5-45).

PROGRAM KEYS

PAUSE Causes program execution to pause, and the display to return, for approximately ¼ second; this enables the partial results of calculations to be viewed. Execution of the program automatically continues after the pause.

Successive PAUSE instructions increase the duration of the pause by ¼ second increments (there will be a slight blink in the display as each successive PAUSE is executed).

PAUSE may be used as a 'conditional stop' where the user, rather than the calculator, decides whether or not to stop the program. The stop is achieved by holding pressed any key (other than STOP) during program execution; the program then stops when the next PAUSE instruction is encountered in the program. The key pressed is not executed, it serves only to cause the stop when the PAUSE is encountered. If there is no PAUSE in the program then pressing any key except STOP has no effect. Once the program is stopped, execution does not then resume until CONTINUE is pressed.

NOTE
STOP cannot be pressed to make this type of 'conditional stop' because there is no way to predetermine at which step program execution will stop.

STEP PRGM The 'step program' key is not programmable; it is used, from the keyboard only, to single-step programs.

In the 'run' mode (i.e., RUN light on) it single-steps program execution; each time STEP PRGM is pressed the program counter is incremented by one so that one program step is executed.

In the 'program' mode (i.e. PRGM light on) STEP PRGM enables the program steps stored in memory to be viewed. Each time STEP PRGM is pressed, the program counter is incremented by one, so that the address and key-code displayed in the Y register shifts to Z, the address and key-code in X shift to Y and the next higher address (and key-code) appears in X.

Use of STEP PRGM is fully detailed under Editing and Correcting A Program, starting on Page 5-20.

PROGRAM KEYS

BACK STEP This key is not programmable. Operation of BACK STEP is similar to operation of STEP PRGM except that BACK STEP decrements the program counter by one each time it is pressed.

BACK STEP should not be used in the 'run' mode; it should be used only in 'program' mode.

Using BACK STEP in the 'run' mode produces confusing results because, although the program counter is decremented each time the key is pressed, program steps cannot be 'un-executed'.

Use of BACK STEP is fully detailed under Editing and Correcting A Program, starting on Page 5-20.

PRINT SPACE Causes the Option 004 printer to print the contents of the X-register; successive PRINT/SPACE instructions cause the printer to space after the initial print. For complete details refer to Chapter 6.

If the printer is not installed in the calculator, then the PRINT/SPACE instruction acts as a STOP operation; it will not otherwise affect program execution, except that it will terminate a short-form address.

FMT The FORMAT key is used to redefine other keys, to control some plug-in ROM's and peripheral devices and also to control some operations involving the magnetic card reader. For specific 'format' commands, refer to the manuals for the individual accessories or peripheral equipment and to the description of the magnetic program cards, later in this chapter.

All keys which are part of a format command (i.e., FMT and the next key) will act as 'no-operations' if the appropriate accessory or peripheral device is not connected; such commands will not affect program execution, except that they will terminate a short-form address.

PROGRAM KEYS

Some FMT commands redefine the entire keyboard for use with optional plug-in or peripheral devices. All subsequent keys then remain redefined until a terminating instruction, usually a single 'FMT', is given. For example:

> 'FMT FMT' results in the character stamped on the front of each subsequent key being printed by the Option 004 Printer if the -hp-11211A Printer Alpha ROM is installed. Printing is then terminated as soon as another 'FMT' key is encountered.

When a FMT command of this type is given and the appropriate plug-in or peripheral device is not installed, then the normal function of the subsequent keys will be executed, instead of their supposedly redefined functions.

To avoid such unwanted execution, substitute a CONTINUE for each of the keys to be redefined, including the FMT command and the terminating command.

PROGRAM OPERATION

'Program Operation' describes methods used to load and run and to edit programs.

The generalized instructions given here may have to be varied slightly depending upon the requirements for the individual programs. Each program should have a set of user-instructions detailing the operating procedures for that particular program. Even though the program is intended for use only by the programmer the operating procedures for a program are easily forgotten; it is often time consuming and difficult to recreate them later.

The section titled 'Program User-Instructions' in Chapter 2 details the type of information user-instructions should contain. Refer also to the programs in the Math Pac, supplied with your calculator, for further examples of user-instructions.

LOAD & RUN PROGRAMS

To load and run a program from the keyboard:

STEP 1: Address the memory:

PRESS: [RUN] (if the RUN light is not lit)

PRESS: [GO TO] [ ] (Starting Address)

STEP 2: Load the program.

PRESS: [PRGM]

PRESS: Keys in the step sequence of the program.

PRESS: [RUN]

STEP 3: Run the program.

PRESS: [GO TO] [ ] (Starting Address)

Enter data as required and press CONTINUE to start program execution at the current address. Press CONTINUE to resume program execution any time the program stops for a data entry, etc.

(The following paragraphs discuss each of the preceding three steps.)

PROGRAM OPERATION

(STEP 1) The calculator must always be in the 'run' mode whenever the memory is to be addressed. The starting address for loading the program must be numeric (even when the starting address for running the program is to be a label). The address may be keyed as a short-form address because it will be terminated by the PRGM key.

If, and only if, the starting address is 0000, then END may be pressed instead of the key sequence 'GO TO, 0, 0, 0, 0'.

(STEP 2) Program steps can be keyed into the memory only when the calculator is in 'program' mode. If any programmable key is pressed, it will be loaded as a program step into the memory.

As each program step is loaded, the display scrolls upward one position so that the code for the key pressed, and the address in which it is now stored, appears in the Z-register. The Y-register always contains the address into which the next step will be loaded. (If an error is made during entry, refer to Editing and Correcting a Program, on Page 5-20.)

It is good practice always to switch back to 'run' mode (press RUN) as soon as a program is loaded; this will prevent any changes being made accidentally in the programmable memory.

(STEP 3) The starting address for running the program will often, but by no means always, be the same as the starting address for loading the program.

The address may be either a 4-digit numeric address or a label. The short-form address should never be used when starting a program because of the uncertainty of its being properly terminated by the next keystroke. The most likely next step will be either a data entry or pressing CONTINUE and neither of these will terminate the address properly.

The user can affect the program results by making any (appropriate) keyboard operations, either before the program is started or whenever it stops for, say, a data entry. Data may be entered and stored in the memory, arithmetic operations may be performed on that data, and so on. Any special requirements for such operations should be detailed in the user-instructions for the individual programs.

EXAMPLE:

Load and run the (A X B)/(A +B) program from Page 5-7.

PRESS: [RUN] (if necessary)

PRESS: [END] (equivalent of GO TO, 0, 0, 0, 0)

(continued)

**LOAD & RUN
PROGRAMS
(continued)**

PROGRAM OPERATION

PRESS: [PRGM]

PRESS: the keys in the step sequence of the program.

PRESS: [RUN] [END]

Enter the data (A and B).

PRESS: numeric keys for A

PRESS: [↑]

PRESS: numeric keys for B

DISPLAY: any number $\rightarrow$ z
               A $\rightarrow$ y
               B $\rightarrow$ x

PRESS: [CONTINUE]

DISPLAY: $\dfrac{A \times B}{A + B}$ $\rightarrow$ z
              A $\rightarrow$ y
              B $\rightarrow$ x

The preceding program can be rerun by entering new values for A and B and then pressing CONTINUE; the END instruction, at address 0012, automatically resets the program counter to 0000 as well as stopping program execution.
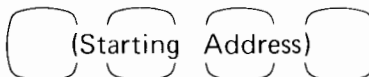
**EDITING
PROGRAMS**

Editing enables programs which are in the memory to be checked, on a step-by-step basis, in either the 'program' or the 'run' mode.

PROGRAM MODE:
    Used to verify that the program steps have been loaded correctly.

PRESS: [RUN] (if necessary)

PRESS: [GO TO] [ ] (Starting Address) [ ]

PROGRAM OPERATION

PRESS: ⎡PRGM⎤ (The starting address is in the Y-register.)

PRESS: ⎛STEP⎞
⎝PRGM⎠

Each time STEP PRGM is pressed, the program counter is incremented by one and the display scrolls up; this enables the key-code for each program step to be viewed in turn (see the appendix at the back of this manual for the key-codes).

PRESS: ⎛BACK⎞
⎝STEP⎠

The BACK STEP key has the opposite effect from the STEP PRGM key; the program counter is decremented by one, so that the display scrolls down, each time BACK STEP is pressed.

As an example step through the program loaded during the example on Page 5-19 and check that the addresses, and the key-codes stored in them, are correct.

RUN MODE:
Used to verify program operation. The program is treated as if it is to be run automatically except that the CONTINUE key is not used; instead, the STEP PRGM key is pressed to execute each step in turn. As each step is executed, the results appear in the display registers exactly as if the same key was being pressed. Data must be entered at the correct steps to check proper operation of the program.

NOTE
1. Neither STEP PRGM nor BACK STEP will terminate a short-form address.
2. Do not use the BACK STEP key in the 'run' mode; the result will be meaningless.
3. There are some situations in which the operation of STEP PRGM, though predictable, is not quite as expected. These are explained after the next example.

EXAMPLE:

Single-step the program from Page 5-7.

(With the program loaded in the memory)

PRESS: ⎡RUN⎤ (if necessary)

(continued)

**EDITING
PROGRAMS
(continued)**

PROGRAM OPERATION

PRESS: [END]

PRESS: Digit keys for A

PRESS: [↑]

PRESS: Digit keys for B

PRESS: [STEP PRGM]

Each time the STEP PRGM key is pressed, observe that the display contains the results of the corresponding operation, as shown in the program.

In some situations (in the 'run' mode only) operation of the STEP PRGM key produces unexpected results. This occurs for short-form addresses (for both the program and data storage memories), and for labels.

Short-Form Address:

(a)

STEP	KEY
0015	GO TO
0016	3
0017	6
0018	×
----	---
0036	↑

With 4-digit addressing the branch is made as soon as STEP PRGM is pressed at the fourth digit. The above key sequence, however, contains a short-form address, therefore STEP PRGM must be pressed at step 0018 before the branch is made. The multiply operation in Step 0018 is 'seen' only as an address termination, it will not be executed. The next time STEP PRGM is pressed, the 'up' operation in step 0036 will be executed.

(b)

STEP	KEY
0324	$x \rightarrow (\ )$
0325	3
0326	×
----	---

Short-form addressing for the data storage memory is similar to short-form addressing for the program memory; in example b, STEP PRGM must be pressed at step 0326 before the store operation is performed. In this case,

PROGRAM OPERATION

because no branch is involved, the multiply operation in step 0326 is executed at the same time.

Labels:

STEP	KEY
0020	GO TO
0021	LABEL
0022	A
----	---
0366	LABEL
0367	A
0368	×

Any time STEP PRGM is pressed at a LABEL key, the next key is 'looked at' simultaneously. There are then two possibilities depending on whether or not a GO TO preceded the LABEL key.

  (a) If the whole of the above sequence is single-stepped, then STEP PRGM need be pressed only at steps 0020 and 0021 before the program counter goes to 0368 (apparently) skipping steps 0022, 0366 and 0367. The third time STEP PRGM is pressed the 'multiply' will be executed.

  (b) If the program counter is set by some other means to point to address 0366 (e.g., GO TO, 0, 3, 6, 6) and STEP PRGM is next pressed, then the counter (apparently) misses only step 0367.

The editing feature enables program steps to be changed, deleted or added, without reloading the entire program.

CORRECTING PROGRAMS

Changing and deleting steps constitute the simplest case.

     PRESS: [RUN] (if necessary)

     PRESS: (GO TO) ((Address of incorrect step))

     PRESS: [PRGM]

     the step to be changed will be in the Y-register.

     PRESS: the correct key or, if the step is to be deleted, CONTINUE ('no operation').

The corrected step will now be in the Z-register.

     PRESS: [RUN]

PROGRAM OPERATION

EXAMPLE:

Change the (A × B)/(A + B) program (Page 5-7) to solve for AB
(A + B). This requires only that step 0008 be changed from 'divide'
to 'multiply'.

(With the program stored in the memory.)

PRESS: `RUN` `GO TO` `0` `0`

PRESS: `0` `8` `PRGM`

DISPLAY: 0007 - - - - - - - 13 → z
          0008 - - - - - - - 35 → y
          0009 - - - - - - - 13 → x

step to be changed

PRESS: `×`

step changed

DISPLAY: 0008 - - - - - - - 36 → z
          0009 - - - - - - - 13 → y
          0010 - - - - - - - 27 → x

PRESS: `RUN`

The program may now be run as before except that the solution
will be for AB (A + B).

NOTE

As this program is also to be used in the next
examples, repeat the above procedure but change
step 0008 back to 'divide'.

**If additional steps are to be inserted into a program, then the remaining
steps in the program must be moved down in the memory, to make room
for those additional steps.** Where the program is short, or where the
addition is to be made near the end of the program, it is a simple matter
to key in the additional steps and then to key the remaining steps into
their new locations. Where there are many steps to be changed, or moved,
keying them all into memory can be a lengthy task. Here are two methods
which can be used to simplify inserting steps into long programs.

PROGRAM OPERATION

METHOD I.

In this method, all of the steps which are to be moved down in the memory are recorded on a magnetic card. The recorded steps are then re-loaded from the card after the additional steps have been inserted.

For simplicity assume that the program of Page 5-7 is a long program and that it is desired to insert a PAUSE instruction between steps 0007 and 0008.

To correct the program:

1. Record the steps which are to be moved, starting with step 0008;

    PRESS: [GO TO]   [O]   [O]

    PRESS: [O]   [8]

    INSERT: magnetic card into the card reader (see instructions on Page 5-28).

    PRESS: RECORD

2. Load the extra step (PAUSE) into address 0008;

    PRESS: [GO TO]   [O]   [O]   [O]

    PRESS: [8]   PRGM

    PRESS: [PAUSE]

    PRESS: [RUN]

3. Re-load the remaining steps from the magnetic card (the program counter is already pointing to address 0009 because the previous operation used address 0008).

    INSERT: magnetic card into the card reader.

    PRESS: LOAD

(continued)

PROGRAM OPERATION

The steps from the card will automatically be moved down in the memory. The program may now be run as before, except that there will be a pause; during the pause the partial result of the program will be displayed.

When Method I is used, care must be taken to correct any branching addresses which might now be wrong as a result of the changes.

METHOD II.
The second method of inserting steps consists of a 'patch-in' technique. This method has the advantage over Method I that very few program steps need be moved to different addresses.

Using the same example as in Method I, it is required to add a PAUSE instruction between steps 0007 and 0008 of the program from Page 5-7. Here is part of the program as originally written:

STEP	KEY	KEY CODE
----	---	--
0007	a	13
0008	÷	35
0009	a	13
0010	↑	27
0011	b	14
0012	END	46

To insert the steps:

1. Starting at the first address to be changed (0008), load instructions to branch to some unused part of the memory. In this example, the branching instruction requires that steps originally in addresses 0008 through 0010 be deleted.

	STEP	KEY	KEY CODE
	0007	a	13
Branch ⎰	0008	GO TO	44
Instruction ⎱	0009	LABEL	51
	0010	π	56
	0011	b	14
	0012	END	46

PROGRAM OPERATION

2. In an unused part of the memory insert the steps to be added, the steps which were deleted and instructions to branch back to the original program. The program may then be run as before except that the partial result will be seen when the PAUSE is executed.

KEY	KEY CODE
LABEL	51
π	56
PAUSE	57
÷	35
a	13
↑	27
GO TO	44
0	00
0	00
1	01
1	01

MAGNETIC PROGRAM CARDS

Both program steps and data can be recorded on a magnetic card (Figure 5-1) and then loaded into the calculator any time in the future. The card, which is six inches long, has two sides, each of which is recorded on individually. The sides are identified only with the word 'SIDE', space being left for the user to write any identifying information he pleases. Either pencil or ballpoint pen may be used to write on the card; neither will affect the magnetic properties of the card.
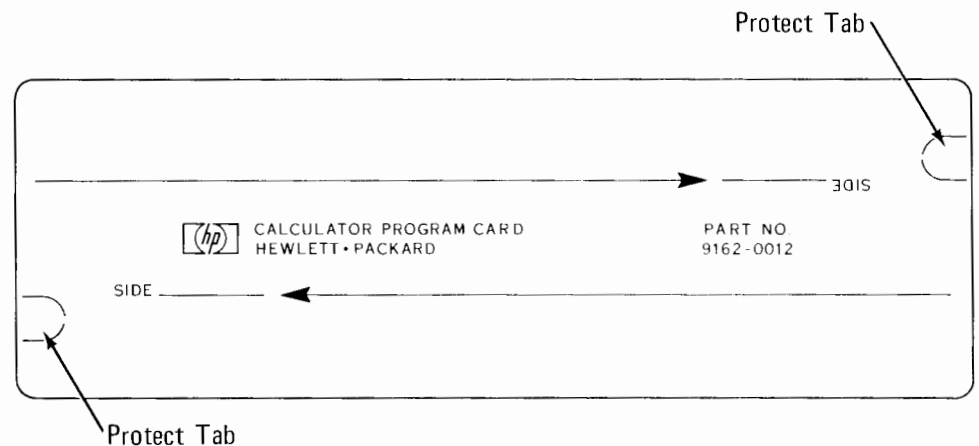
Figure 5-1. The Magnetic Program Card

CAPACITY OF THE CARD

Sufficient program steps can be stored on a six inch magnetic card that the entire basic program memory (500 steps) can be recorded on one card-side. The complete memory in Option 002 calculators (1012 steps) requires two card-sides, and in Option 003 calculators (2036 steps) requires four card-sides.

When data is recorded, the basic data memory of 49 numeric registers (not including a and b, which cannot be recorded) requires one card-side; Option 001 calculators, which have 109 numeric registers, require two card-sides.

Occasionally, one card-side is not sufficient to record the amounts stated above; this is normal and presents no problem because the INSERT CARD light indicates that another card-side is required.

INSERTING CARDS INTO THE READER

To record or load, insert the magnetic card into the upper slot of the card-reader as shown in Figure 5-2. Orient the card so that the printed face is towards the keyboard and the arrow on the card-side to be used is pointing down. Lean the card slightly towards the keyboard and insert it approximately an inch into the slot until a stop is felt. Do not force the card into the slot, it will be automatically pulled through the reader when the appropriate keys are pressed.

MAGNETIC PROGRAM CARDS
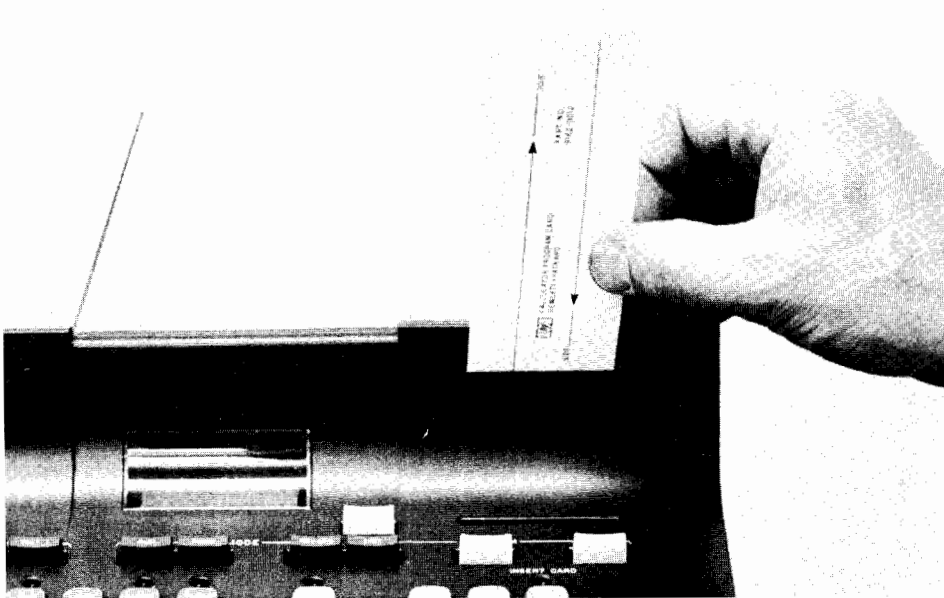
Figure 5-2. Inserting the Card

During the record (or load) process, the INSERT CARD light is lit. If the light remains lit when the card stops moving through the card reader, then the recording (or loading) is not complete and another card must be inserted. Recording (or loading) stops either when an END statement has been transferred or when the end of the memory has been reached.

PROTECTING THE RECORDING

Program steps or data, once recorded, can be loaded into the calculator any number of times. New information can be recorded over previously recorded information on a card provided that the card is unprotected. To permanently protect the information recorded on a card-side, remove the protect tab (see Figure 5-1) from that side of the card. Any attempt to record on the protected card-side lights the STATUS light and does not affect the recording currently on the card.

RECORD

The RECORD and LOAD keys, which are not programmable, control the recording and loading of programs (not data) on the magnetic cards. (Other keys are also used for these purposes but under different circumstances; these will be explained later.)

To record a program which is in the memory:

    1. Insert an unprotected card into the card reader as described above.

(continued)

RECORD

(continued)

MAGNETIC PROGRAM CARDS

2. Set the program counter to the beginning address of the program to be recorded:

PRESS: [RUN] (if necessary)

PRESS: (GO TO) ((Starting Address))

The address may be short-form because it will be terminated when the RECORD key is pressed in the next step. The address may be a LABEL, but if it is, then the label will not be recorded. END may be pressed instead of GO TO, 0, 0, 0, 0.

3. Record the program:

PRESS: [RECORD]

The card reader motor transports the card through the reader and partially ejects it from the bottom slot. When the card stops moving, recording has also stopped; remove the card from the card reader.

NOTE

If the card is not pulled through the card reader, then it was probably not inserted properly. Do not press any keys; remove the card from the card reader and re-insert it.

---

4. If the INSERT CARD light is still on, then the program is not completely recorded and another card is required. Turn the card over and re-insert it into the card reader (printed side facing the keyboard) so that the other card-side can be recorded, or insert a new card. It is not necessary to readdress the memory or press the RECORD key; as long as the INSERT CARD light is on, the card reader will automatically continue recording as soon as the card is inserted. Continue inserting cards as necessary until the entire program is recorded. (The recording process can be stopped at any time by pressing the STOP key; this will probably result in the recording being incomplete.)

MAGNETIC PROGRAM CARDS

NOTE

Unless STOP is pressed, the recording process stops when an END instruction has been recorded; the program counter remains pointing to the address containing the END. If there is no END, then recording continues right through the memory and stops only when the 'end of memory' program (see Status Conditions in Chapter 3) has been recorded.

---

5. To avoid confusion in the future, identify the card by writing on the card such information as the name of the program, the number of the card-side (if more than one side was used) and so on. Also, if required, protect the recording by removing the protect tab (it is recommended that you first verify that the recording is correct, because the steps on a protected card-side cannot be changed).

An example recording procedure is given after the LOAD key explanation.

TO RECORD A PROGRAM IN SECURE MODE?
1. Set the program printed by [END] or [GO TO] on...
2. Hit [FORMAT] [RECORD]

To load a program from a magnetic card into the calculator's memory: `LOAD`

1. Insert the card, into the card reader, with the printing facing the keyboard and the arrow on the required card-side pointing down.

2. Set the program counter to the address which is to contain the first step:

    PRESS: (GO TO) ((Starting Address))

    The address may be short-form but it would not normally be a label.

3. Load the program:

    PRESS: [LOAD]

    As for recording, the loading continues until either an END is loaded, or, if there is no END, until the last address in the memory is reached. If the INSERT CARD light remains lit when the card stops then another card-side is required. (Loading can be stopped at any time by pressing the STOP key.)

(continued)

MAGNETIC PROGRAM CARDS

LOAD

(continued)

4. If the loading stops at an END, then the program counter will be pointing to the address which contains that END. If another card is now inserted and LOAD is pressed, then the first new step loaded will be substituted for the END from the previous program. This is a useful feature as it enables programs, part programs, and subroutines to be chain-loaded and then, if required, recorded as a composite program.

EXAMPLE:

Record the (A × B)/(A + B) program (Page 5-7) on a magnetic card and then load the program from that card into another part of the memory (address 0020, for example).

1. Ensure that the steps of the program have been correctly keyed into the calculator.

2. Record the program:

   PRESS: ⬚ RUN ⬚ (if necessary)

   Insert an unprotected card into the card reader.

   PRESS: ⬚ END ⬚   ⬚RECORD⬚

3. Load the program from the card into the calculator; start the loading process at address 0020 (in this program the change in position in the memory can be accomplished easily because there are no branching addresses to be changed):

   PRESS: ⬚ GO TO ⬚   ⬚ 0 ⬚   ⬚ 0 ⬚

   PRESS: ⬚ 2 ⬚   ⬚ 0 ⬚
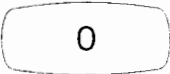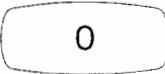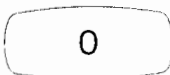
   Insert the card into the reader using the same card-side as in step 2.

   PRESS: ⬚ LOAD ⬚

4. The program may now be run as before except that the starting address (0020) must be set each time the program is to be run:

MAGNETIC PROGRAM CARDS

PRESS: [GO TO] [0] [0]

PRESS: [2] [0]

Enter the value for A into the Y-register and for B in the X-register.

PRESS: [CONTINUE]

[FMT] [GO TO] Automatically loads and starts execution of a program recorded on a magnetic card. The two keys are the equivalent of the following key sequences given together:

AUTOMATIC 'LOAD & RUN'

GO TO, 0, 0, 0, 0, LOAD — which loads a program, starting at address 0000, from the card in the card reader, and

GO TO, 0, 0, 0, 0, CONTINUE — which starts program execution at address 0000, as soon as the END has been loaded from the magnetic card.

'FMT, GO TO' may be given from the keyboard. Alternatively, it may be included as the last steps executed in a program, so as to 'call-for' the next program to be automatically loaded and run. In this way the parts of a very long program may be conveniently linked.

All of the rules applicable to the normal load routine also apply to this automatic load except that program execution starts automatically as soon as an END is loaded.

MAGNETIC PROGRAM CARDS

RECORDING DATA

FMT **x→( )** Causes the data stored in the numeric registers to be recorded on a magnetic card. Recording starts at register 000 and continues sequentially until the contents of all of the numeric registers has been recorded. The INSERT CARD light remains lit if another card side is required (see 'Capacity of the Card' on Page 5-28). Recording may be stopped at any time by pressing the STOP key.

The contents of the *a* and *b* registers cannot be recorded.

If 'FMT, x → ( )' came from a program, then program execution automatically continues as soon as the recording is complete. (Press CONTINUE to resume program execution if the recording was stopped by pressing the STOP key.)

LOADING DATA

FMT **x←( )** Causes the data stored on a magnetic card to be loaded into the numeric data storage registers. Loading starts at register 000 and continues until all of the registers have been loaded.

The INSERT CARD light remains lit if another card side is required. Loading can be stopped at any time by pressing STOP.

The contents of the *a* and *b* registers are not affected by the loading process.

If 'FMT, x ← ( )' came from a program, then program execution automatically continues as soon as the loading process is complete. (Press CONTINUE to resume program execution if loading was stopped by pressing STOP.)

PROGRAM KEYS - CONDITIONAL BRANCHING
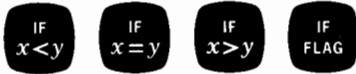
The four 'IF' keys are used for 'conditional branching' (see Page 5-10). Each key tests the condition defined on that key; the subsequent program operation is then determined by the result of the test.

The three keys which contain the 'x' and 'y' symbols compare the numbers contained in the X and Y registers. The three conditions defined are:

IF x < y
'If the number in X is less than the number in Y . . . . '

IF x = y
'If the number in X equals the number in Y . . . . '

IF x > y
'If the number in X is greater than the number in Y . . . . '

The IF FLAG key is a special case, it tests a condition definable by the user (see SET FLAG, Page 5-39, for a complete explanation).

For all four conditions there can be only two results to a test; either the condition is 'met' (YES) or the condition is 'not met' (NO).

CONDITION NOT MET:
When the tested condition is not met (NO), the program automatically skips (ignores) the next four steps and continues normal execution at the fifth step following the 'IF'.

CONDITION MET:
When the tested condition is met (YES), the program automatically goes to the next step following the 'IF'; subsequent operation then depends upon the type of key-sequence contained in the four steps following the 'IF' key. The simplest case is a key-sequence consisting of a 4-digit address; the program branches to that address, executes the step contained there, and then continues program execution sequentially from that address.

NOTE

The other types of key sequence which can be used after an IF key require that the programmer remember numerous exceptions to the branching rule. For this reason, those sequences are discussed in a separate section, 'Advanced Branching Techniques', at the end of this chapter.

Computer Museum

IF
$x<y$

IF
$x=y$

IF
$x>y$

IF
FLAG

(continued)

PROGRAM KEYS - CONDITIONAL BRANCHING

When an IF instruction (other than IF FLAG) is encountered, the two numbers in X and Y are automatically rounded before the test is made. In each register, the tenth digit of the number is rounded according to the value of the guard digits; the guard digits are then set equal to zero. Thus the numbers tested actually have the same values as would appear in a floating point display with all ten significant digits displayed. After the test has been made, the numbers in X and Y retain their rounded values.

In most cases these rounded values are the desired values for further use. However, sometimes the slight loss in accuracy resulting from the rounding may not be desireable; in these cases, provision can be made, in the following way, to retain accuracy. Before a test is made the two numbers can be stored in the data-storage memory; after the test, the stored numbers can be recalled and substituted for the rounded numbers.

In the following sequence the conditional key in step 0236 tests to determine if the numbers in X and Y are equal:

STEP	KEY
0236	IF $x=y$
0237	0
0238	3
0239	4
0240	7
0241	↑

If the condition is not met (NO the numbers are not equal): four steps, 0237 through 0240, are skipped and the 'up' in 0241 is executed.

If the condition is met (YES the numbers are equal): the program 'sees' address 0347 (contained in steps 0237 through 0240), it branches to that address and executes whatever instruction is contained there.

The next program is a simple counting routine which illustrates a typical use for an 'IF' key (in this case, IF $x < y$). The program sums any positive number (N) until the total (Σn) is equal to, or greater than, 100. 'IF $x < y$' tests to determine whether or not 100 has been equalled or exceeded.

The final display shows N in the Z-register, 100 in Y and ΣN (the final Σn) in X.

If N is negative, then 100 is never reached and the program sums (−)N to (−)infinity.

Load the program, starting at address 0000; to run the program, press END, enter any (positive) value for N into the X-register and press the CONTINUE key.

PROGRAM KEYS - CONDITIONAL BRANCHING

FLOW CHART
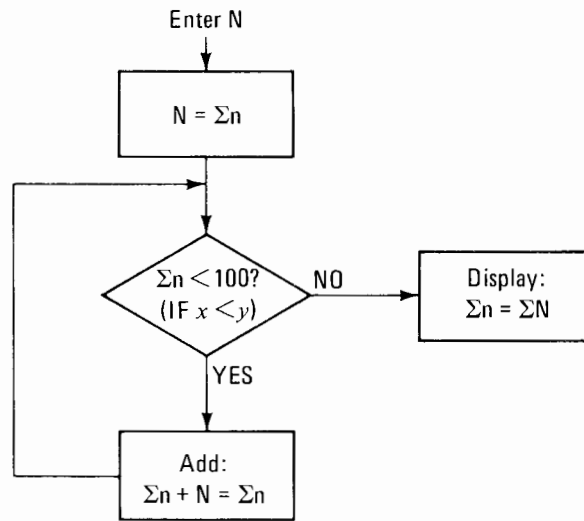
At step 0008, if Σn is less than 100, the condition tested is met (YES); the program continues to the next steps and, as these constitute an address, it branches (to step 0014). When Σn is no longer less than 100 (i.e. $\Sigma n = \Sigma N \geqslant 100$), the program skips steps 0009 through 0012 and executes the STOP at step 0013. (Notice the use of the short-form address in step 0018.)

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE ∂
0000	$x \rightarrow (\ )$	23	N	-	-	-
0001	∂	13	N	-	-	N
0002	$\uparrow$	27	N	N	-	
0003	ENTER EXP	26	1	N	-	
0004	2	02	100	N	-	
0005	$\uparrow$	27	100	100	N	
0006	∂	13	N = Σn	100	N	
0007	PAUSE	57	Σn	flashing display 100	N	
0008	IF $x < y$	52	Σn	100	N	
0009	0	00	Σn	100	N	
0010	0	00	Σn	100	N	
0011	1	01	Σn	100	N	
0012	4	04	Σn	100	N	
0013	STOP	41	Σn = ΣN	final display 100	N	
0014	$x \leftarrow (\ )$	67	Σn	100	N	
0015	$+$	33	Σn	100	N	
0016	∂	13	(Σn + N) =Σn	100	N	
0017	GO TO	44	Σn	100	N	
0018	7	07	Σn	100	N	
0019	END	46				

IF
$x<y$

IF
$x=y$

IF
$x>y$

IF
FLAG

(continued)

PROGRAM KEYS - CONDITIONAL BRANCHING

The next program calculates N! (N factorial) for values of N up to, and including, 69.

N! = N(N−1)(N−2) . . . . . (N−N+2)(1)
6! = 6 × 5 × 4 × 3 × 2 × 1 = 720
0! = 1 (by definition)

FLOW CHART

n = current multiplier term (N, N−1, N−2, etc.)
P = partial result of N!

Enter N

Store N
N = n
1 = P

n < 1?

YES

NO

Multiply:
P × n = P

Subtract:
n − 1 = n

P = N!
Recall n
Display
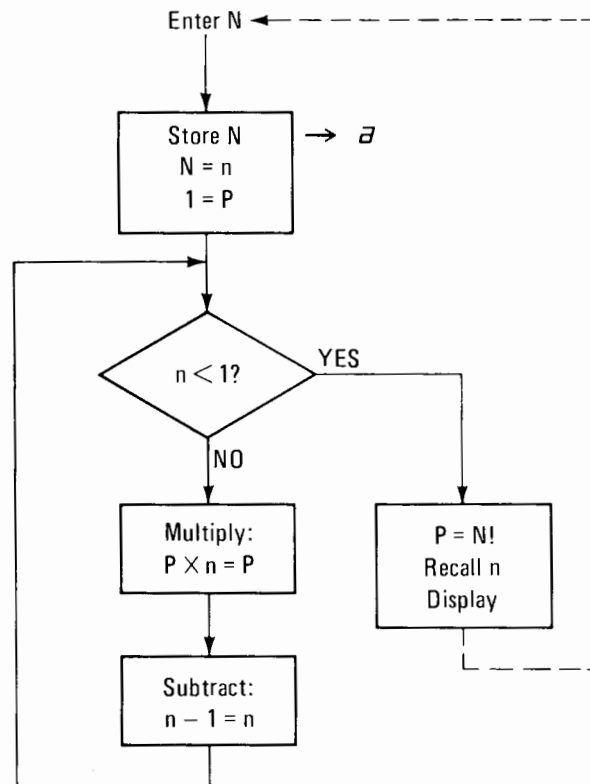
Load the program, starting at address 0000; to run the program, press END, enter the value for N into the X register and press CONTINUE. This program is arranged so that, once the program has been run, the program counter automatically sets to address 0000 ready for the next value of N; the END at step 0020 does this.

PROGRAM KEYS - CONDITIONAL BRANCHING

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE a	STORAGE b
0000	$x \rightarrow (\ )$	23	N	-	-	-	-
0001	a	13	N	-	-	N	-
0002	↑	27	N	N = n	-		-
0003	1	01	1	n	-		-
0004	$x \rightarrow (\ )$	23	1	n	-		-
0005	b	14	1	n	-		1 = P
0006	IF $x > y$	53	1	n	-		
0007	0	00	1	n	-		
0008	0	00	1	n	-		
0009	1	01	1	n	-		
0010	7	07	1	n	-		
0011	$y \rightarrow (\ )$	40	1	n	-		
0012	×	36	1	n	-		
0013	b	14	1	n	-		P x n = P
0014	—	34	1	n − 1 = n	-		
0015	GO TO	44	1	n	-		
0016	6	06	1	n	-		
0017	a	13	N	n = 0	-		
0018	↑	27	N	N	0		
0019	b	14	P = N!	N	0		
			N!	N	0		
0020	END	46	N!	N	0		

final display

The SET FLAG key establishes the condition to be tested by the IF FLAG key (Page 5-35).

The 'YES' condition is established by giving SET FLAG, either from a program or from the keyboard.

The 'NO' condition is established by clearing the flag; the flag is cleared automatically whenever the calculator is switched on or by the CLEAR or IF FLAG (given from the keyboard or from a program).

The flag enables the programmer to select the conditions which will determine whether a conditional branch is to be made. The types of condition which require the flag, and the actual use of the flag, are best illustrated by examples; the next two programs serve this purpose.

SET
FLAG

(continued)

PROGRAM KEYS - CONDITIONAL BRANCHING

The following program illustrates a situation in which the flag must be set from the keyboard. The program calculates the average value ($\overline{X}$) of N data points. The formula used is:

$$\frac{X_1 + X_2 + X_3 \ ..... + X_n}{N} = \frac{\Sigma X_i}{N} = \overline{X}$$

As can be seen from the flowchart, the program is arranged so that each time a data point is entered it is added to the accumulative total. At the same time, a counting routine counts the number of data points. When there are no more data points to be entered the user sets the flag, thus establishing the 'YES' condition and causing the average value of the data points to be calculated.
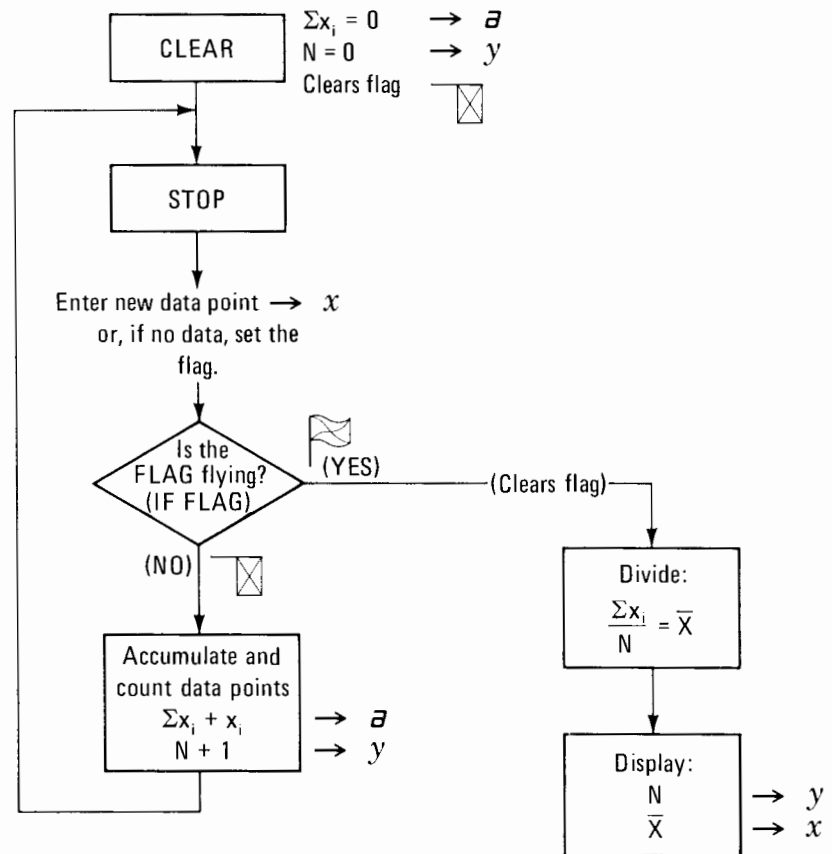
Load the program starting at address 0000; to run the program:

1. PRESS: END   CONTINUE

PROGRAM KEYS - CONDITIONAL BRANCHING

2. Enter the first data point (X_1) into the X-register and

PRESS: [CONTINUE]

3. Enter the next data point into X and press CONTINUE.

4. Repeat step 3 for each one of the remaining data points.

5. PRESS: [SET FLAG] [CONTINUE]

| STEP | KEY | KEY CODE | DISPLAY | | | STORAGE |
			X	Y	Z	∂
0000	CLEAR	20	0	0	0	$0 = \Sigma x_i$
0001	STOP	41	x_i	previous N		Σx_i
0002	IF FLAG	43	x_i	N		
0003	0	00	x_i	N		
0004	0	00	x_i	N		
0005	1	01	x_i	N		
0006	5	05	x_i	N		
0007	$x \rightarrow (\ )$	23	x_i	N		
0008	+	33	x_i	N		
0009	∂	13	x_i	N		$\Sigma x_i + x_i = \Sigma x_i$
0010	1	01	1	N		Σx_i
0011	+	33	1	N + 1 = N		
0012	CLEAR x	37	0	N		
0013	GO TO	44	0	N		
0014	1	01	0	N		
0015	$y \rightarrow (\ )$	40	x_i	N		
0016	$\div$	35	x_i	N		
0017	∂	13	x_i	N		$\dfrac{\Sigma x_i}{N} = \overline{X}$
0018	∂	13	$\overline{X}$	N	0	$\overline{X}$
0019	END	46				

enter / previous N

final display

SET
FLAG

(continued)

PROGRAM KEYS - CONDITIONAL BRANCHING

The next program illustrates use of the SET FLAG instruction as a program step. This program calculates the following alternating series:

$$1 - 3 + 5 - 7 + 9 - \ldots \ldots \infty$$

NOTE

A series with simple terms is chosen as this program is intended to illustrate use of the IF FLAG and SET FLAG to alternately branch to either side of a loop. The branching technique can, however, be easily adapted to any series with more complex terms.

In the program, each term ($\pm$n) of the series is generated and added to the accumulative total (P). As each term is added, one (1) is added to a counter (N), to count the number of terms.

The sign of n must be alternated (+ or $-$); the flag is used to determine whether to leave n positive or change the sign to ($-$). This is shown in the partial flow chart given below:
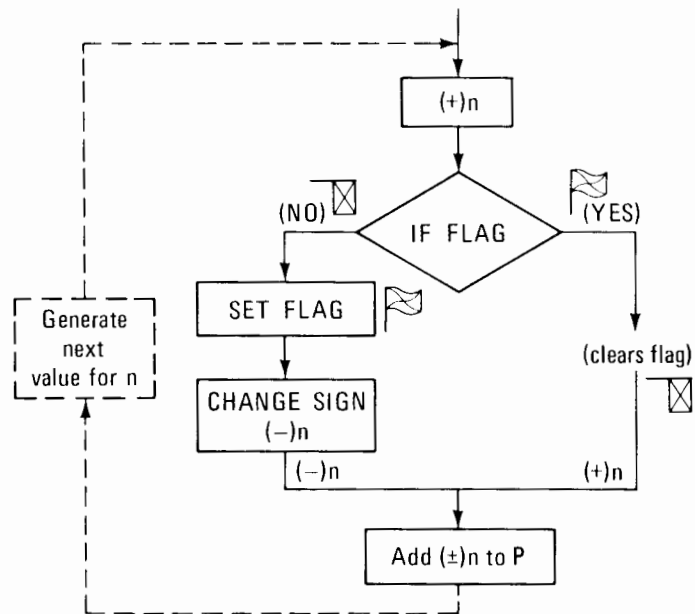
If the flag is set:

a. Positive (+)n is added to P;

b. The flag is cleared (whenever an IF FLAG instruction is encountered the flag is cleared automatically).

PROGRAM KEYS - CONDITIONAL BRANCHING

The next time through, the flag is not set:

a. Negative $(-)$n is added to P;

b. The flag is set (by a program step) so that the next time through $(+)$n will be added.

FLOW CHART: for series $1 - 3 + 5 - 7 + 9 \ldots \ldots \infty$

    n = current term of the series
    N = number of terms added
    P = the accumulative total of the series.

```
        ┌────────────────┐  ┌─┐
        │   CLEAR a, b   │──┤╳│
        │   and FLAG     │  └─┘
        └───────┬────────┘
                │
        ┌───────┴────────┐
        │  P ⎫           │
        │  N ⎬ = 1       │
        │  n ⎭           │
        └───────┬────────┘
                │
        ┌───────┴────────┐    P  →  a
        │     Store      │    N  →  b
        └───────┬────────┘
  ┌─────────────┤
  │     ┌───────┴────────┐    P    →  z
  │     │    PAUSE       │    N    →  y
  │     │  to display    │   (±)n  →  x
  │     └───────┬────────┘
  │     ┌───────┴────────┐
  │     │  |n| + 2 = n   │
  │     └───────┬────────┐
  │             │
  │  (NO)╳  ◇───┴───◇  ⚑(YES)
  │       ◇  IF FLAG  ◇
  │        ◇─────────◇
  │        │          │
  │  ┌─────┴────┐⚑     │ (Clears flag)
  │  │ SET FLAG │      │  ╳
  │  └─────┬────┘      │
  │  ┌─────┴──────┐    │
  │  │CHANGE SIGN │    │
  │  │   (−)n     │    │
  │  └─────┬──────┘    │
  │   (−)n │          │(+)n
  │        └────┬──────┘
  │     ┌───────┴────────┐    →  a
  │     │ Add (±)n to P  │
  │     │ Add 1 to N     │    →  b
  │     └───────┬────────┘
  └─────────────┘
```

SET FLAG
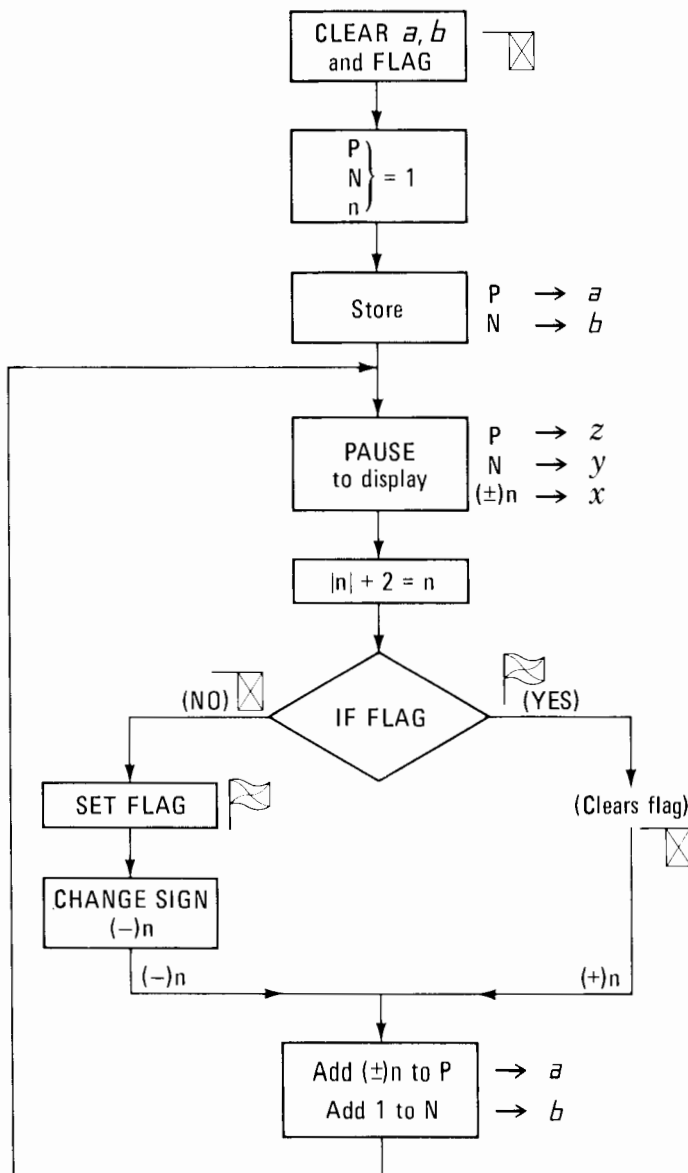
(continued)

PROGRAM KEYS - CONDITIONAL BRANCHING

Load the program starting at address 0000; to run the program:

PRESS: [END] [CONTINUE]
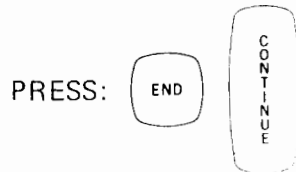
To stop the program and display P, N and n:

PRESS: [PAUSE] (press CONTINUE to resume program execution).
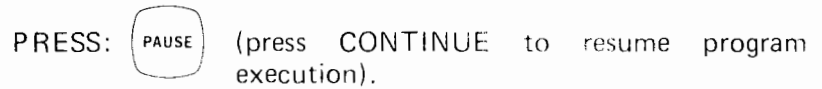
(Note the use of x^2, $\sqrt{x}$ to ensure that n is always positive at step 0011.)

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE a	STORAGE b
0000	CLEAR	20	0	0	0	0	0
0001	1	01	1	0	0		
0002	$x\to(\ )$	23	1	0	0		
0003	a	13	1	0	0	1=P	
0004	$x\to(\ )$	23	1	0	0	P	
0005	b	14	1	0	0		1=N
0006	↑	27	N	P	0		N
0007	↑	27	1=n	N	P		
0008	PAUSE	57		flashing display			
0009	PAUSE	57	±n	N	P		
0010	x^2	12					
0011	$\sqrt{x}$	76	|n|=n	N	P		
0012	↑	27	n	n	N		
0013	2	02	2	n	N		
0014	+	33	2	n+2=n	N		
0015	↓	25	n	N	N		
0016	IF FLAG	43					
0017	0	00					
0018	0	00					
0019	2	02					
0020	3	03					
0021	SET FLAG	54					
0022	CHG SIGN	32	−n	N	N		
0023	$x\to(\ )$	23	±n	N	N		
0024	+	33	±n	N	N		
0025	a	13	±n	N	N	P+(±)n=P	
0026	↑	27	±n	±n	N	P	
0027	1	01	1	±n	N		
0028	$x\to(\ )$	23	1	±n	N		
0029	+	33	1	±n	N		
0030	b	14	1	±n	N		N+1=N
0031	a	13	P	±n	N		N
0032	↑	27	P	P	±n		
0033	b	14	N	P	±n		
0034	ROLL ↑	22	±n	N	P		
0035	GO TO	44					
0036	8	10					
0037	END	46					

PROGRAM KEYS - SUBROUTINES

The SUB/RETURN key is used to call (SUB), and to return from (RETURN), a subroutine (see also Page 5-10). Keying sequences are shown below.

Subroutines may be nested up to a depth of five. An attempt to nest to a depth of six is an error; the program will stop and the STATUS light will be activated (for complete details refer to 'STATUS CONDITIONS' at the end of Chapter 3).

Both the initialization process, which occurs at turn-on, and the END key (given from either a program or the keyboard) automatically reset the nesting to a depth of zero so that all five depths are then available.

Programs stored as subroutines may also be used as stand-alone programs (illustrated on Page 5-50).

SUB.

| GO TO | SUB/RETURN | (Starting address of subroutine) |

The above keying sequence is used in the calling program to (unconditionally) call a subroutine; the starting address in the sequence may be a 4-digit numeric address, a short-form numeric address (properly terminated) or a label ('LABEL, any'). If the address is a label then the first two steps of the subroutine must also be that label. The keying sequences used to conditionally call a subroutine are detailed under Advanced Branching Techniques at the end of this chapter. Execution of the steps of the subroutine starts automatically as soon as the subroutine is called.

RETURN.
SUB/RETURN must also be included as the last step to be executed in the subroutine. The RETURN causes a branch to the 'return-address' in the calling program. The return-address is always the address immediately following the last step used to call the subroutine. Execution of the calling program then continues automatically, starting at the return-address.

The following program steps illustrate a call and the subsequent return.

Calling Program

STEP	KEY
----	----
0025	GO TO
0026	SUB/RETURN
0027	0
0028	0
0029	3
0030	7
0031	$x \leftarrow (\ )$
----	----

Subroutine

STEP	KEY
0037	$x \rightarrow (\ )$
0038	∂
----	----
----	----
0058	PAUSE
0059	SUB/RETURN

**SUB
RETURN**
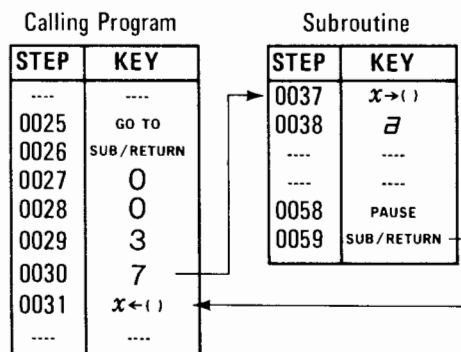
(continued)

PROGRAM KEYS - SUBROUTINES

Instead of executing the step at address 0031, the program branches to address 0037 and executes the steps of the subroutine. At address 0059, the 'RETURN' causes the program to branch back to the return-address (0031) and continue execution of the calling program from there.

NESTING SUBROUTINES
Any number of subroutines may be called individually during a program; however, it is also possible to use more than one subroutine at one time.

One subroutine can call a second subroutine which, in turn, can call a third subroutine, and so on. This multiple-calling is known as 'nesting'. The calculator can remember (store) from one to five return-addresses at a time so that the subroutines may be nested any depth up to a depth of five. Calling a sixth subroutine without first returning to at least the fourth depth constitutes an error because the calculator cannot remember more than five return-addresses at one time.

Returns are made on a 'last-in, first-out' basis, the return always being made to the last return-address stored. As soon as the return is made, that return-address is forgotten (erased from storage) so that the previous address now becomes the 'last' one. Thus the returning order is always the opposite of the calling order.

NOTE

When preparing to run a program which nests subroutines, press the END key to ensure that all five levels of nesting are available; END automatically erases any stored return-addresses to which the return has not yet been made.

The next program illustrates the use of a subroutine; the N! program (Page 5-38), altered slightly, is used as the subroutine.

The program calculates possible combinations (C) of n objects taken k at a time. For example:

If a box contains (n) 15 differently colored balls, how many possible color combinations (C) are there if (k) 5 balls are selected (and then returned) at a time?

Answer = 3003

The formula used in the 'possible combinations' program is:

$$C_k^n = \frac{n!}{k!\,(n-k)!}$$

The program calculates C for any values of n and k up to, and including, 69.

PROGRAM KEYS - SUBROUTINES

SIMPLIFIED FLOWCHART

```
┌──────────────┐
│    (n−k)!    │
└──────────────┘
        │
        ▼
┌──────────────┐
│      k!      │
└──────────────┘
        │
        ▼
┌──────────────┐
│   k! (n−k)!  │
└──────────────┘
        │
        ▼
┌──────────────┐
│      n!      │
└──────────────┘
        │
        ▼
┌──────────────┐
│   n!         │
│ ─────── = C  │
│  k! (n−k)!   │
└──────────────┘
```

FINAL FLOWCHART

```
┌──────────────┐
│ CLEAR, STOP  │
└──────────────┘
        │
        ▼
   Enter data      n ⟶ y
                   k ⟶ x
        │
        ▼
┌──────────────┐
│   Store:     │
│     n        │  ⟶ 000
│     k        │  ⟶ 001
└──────────────┘
        │
        ▼
┌──────────────┐
│  Subtract:   │
│    n−k       │
└──────────────┘
        │
        ▼
┌──────────────┐
N! ◄──│   (n−k)      │
      │   (n−k)!     │
      │   Store      │  ⟶ 002
└──────────────┘
        │
        ▼
┌──────────────┐
│   Recall k   │
N! ◄──│    (k)       │
      │    k!        │
└──────────────┘
        │
        ▼
┌──────────────┐
│ Multiply and │
│   Store:     │
│  k! (n−k)!   │  ⟶ 002
└──────────────┘
        │
        ▼
┌──────────────┐
│   Recall n   │
N! ◄──│    (n)       │
      │    n!        │
└──────────────┘
        │
        ▼
┌──────────────┐
│   Divide:    │
│   n!         │
│ ─────── = C  │
│  k! (n−k)!   │
└──────────────┘
        │
        ▼
┌──────────────┐
│  Display:    │
│     C        │  ⟶ z
│     n        │  ⟶ y
│     k        │  ⟶ x
└──────────────┘
```

```
┌──────────────┐
│ Subroutine ◄─│──( )
│     N!       │──▶
└──────────────┘
```

SUB
RETURN
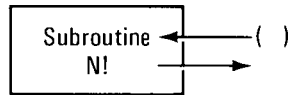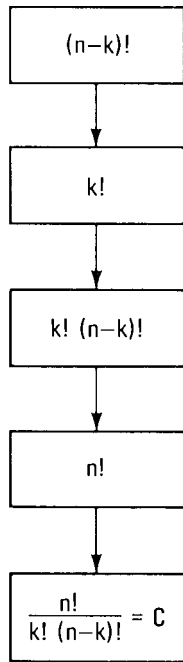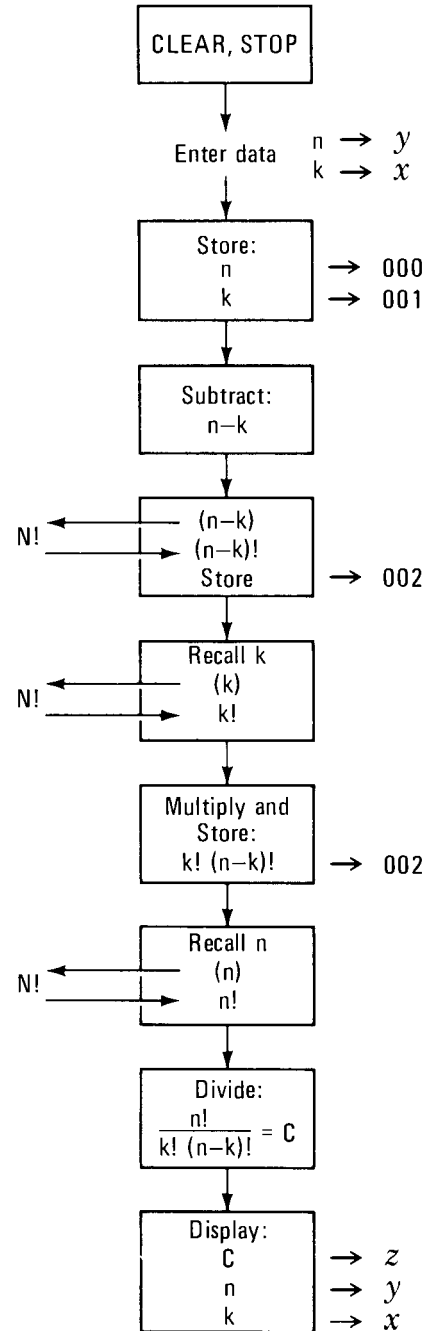
(continued)

PROGRAM KEYS - SUBROUTINES

Load the program starting at address 0000; the N! subroutine follows immediately after the main program.

To run the program:

PRESS: [END]   [CONTINUE]
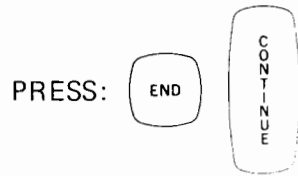
Enter data: n(15) $\rightarrow$ y
             k(5) $\rightarrow$ x

PRESS: [CONTINUE]
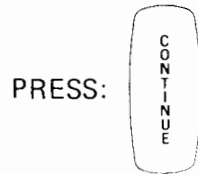
Before the final answer appears, the display will flash three times, showing the results of the factorial subroutine for each value of N (n−k, k and n respectively).

FLASHING DISPLAY: zero $\rightarrow$ z
                    N $\rightarrow$ y
                    N! $\rightarrow$ x

FINAL DISPLAY: C   *3003.* $\rightarrow$ z
                n   *15.* $\rightarrow$ y
                k   *5.* $\rightarrow$ x

(Note that this program contains several examples of short-form addressing, including short-form data register addresses.)

PROGRAM KEYS - SUBROUTINES

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z
0000	CLEAR	20	0	0	0
			enter		
0001	STOP	41	k	n	
0002	$y \rightarrow ( )$	40			
0003	0	00		n	
0004	$x \rightarrow ( )$	23			
0005	1	01	k		
0006	—	34	k	n−k	
0007	↓	25	n−k		
0008	GO TO	44			
0009	SUB/RETURN	77	Calculate (n−k)!		
0010	3	03			
0011	7	07			
0012	$x \rightarrow ( )$	23	(n−k)!	n−k	0
0013	2	02			
0014	$x \leftarrow ( )$	67			
0015	1	01	k		
0016	GO TO	44			
0017	SUB/RETURN	77	Calculate k!		
0018	3	03			
0019	7	07			
0020	$x \rightarrow ( )$	23	k!	k	0
0021	×	36			
0022	2	02			
0023	$x \leftarrow ( )$	67			
0024	0	00	n		
0025	GO TO	44			
0026	SUB/RETURN	77	Calculate n!		
0027	3	03			
0028	7	07			
0029	$x \leftarrow ( )$	67	n!	n	0
0030	÷	35			
0031	2	02	C	n	0
0032	$x \rightleftharpoons y$	30	n	C	0
0033	↑	27		n	C
0034	$x \leftarrow ( )$	67			
0035	1	01	k		
			final display		
0036	STOP	41	k	n	C
(continued)					

DATA STORAGE REGISTERS	
000	n
001	k
002	(n−k)!, k!(n−k)!

SUB
RETURN

(continued)

PROGRAM KEYS - SUBROUTINES

STEP	KEY	KEY CODE	DISPLAY X	Y	Z
0037	$x\to(\ )$	23	N		
0038	a	13			
0039	↑	27			
0040	1	01			
0041	$x\to(\ )$	23			
0042	b	14			
0043	IF $x>y$	53			
0044	0	00			
0045	0	00			
0046	5	05			
0047	5	05			
0048	$y\to(\ )$	40			
0049	×	36			
0050	b	14			
0051	—	34			
0052	GO TO	44			
0053	4	04			
0054	3	03			
0055	a	13			
0056	↑	27			
0057	b	14		flashing display	
0058	PAUSE	57	N!	N	0
0059	SUB/RETURN	77			
0060	ENO	46			

N! SUBROUTINE

A program written as a subroutine may also be used as a 'stand-alone' program provided that two simple rules are observed:

1. Press END to erase any return-addresses currently stored in the calculator.

2. Do not use the SUB/RETURN key when addressing the memory from the keyboard before the program is run. If SUB/RETURN were to be used then the current address (in this case 0000) would become the return-address; program execution would automatically continue at that address when the steps of the subroutine had been executed.

To use the N! subroutine from the previous program as a stand-alone program:

PRESS: (END)

PROGRAM KEYS - SUBROUTINES

PRESS: [GO TO] (0) (0) (3) (7)

Enter a value for N into the X-register and then press CONTINUE.

The program will stop with the value for N in Y and N! in X. The STATUS light will also be on, indicating that there was no return-address for the RETURN at step 0059. In this case the light may be ignored because the lack of any return-address was intentional.

ADVANCED BRANCHING TECHNIQUES

When testing with the conditional branching keys (Page 5-35), there can be only two possible results to the test; either the tested condition is 'not met' (NO), or it is 'met' (YES).

The 'not met' condition results in the four steps following the 'IF' key being skipped and program execution resuming at the fifth step. There is never an exception to the 'skip-four-steps' rule.

The 'met' condition results in the program going to the next step after the 'IF'. The purpose of the four steps after an IF key is to contain the four digits of a branching address, hence the need to skip those steps when the condition is 'not met'.

The conditional branches in the example programs presented earlier in this chapter all use four-digit addresses; however, it is possible to use other types of address and even operations. Here is a general description of the type of key-sequences which can follow an 'IF':

   a. The address which follows an IF key can take any one of four forms (listed below); notice that the 'GO TO' key is required only if the LABEL is used. Numeric addresses can be short-form if properly terminated.

      1. Branch to a numeric address ..... 'n, n, n, n'.

      2. Branch to a label ..... 'GO TO, LABEL, (any)'.

      3. Call a subroutine with a numeric address ..... 'SUB/RETURN, n, n, n, n'.

      4. Call a subroutine with a label address ..... 'GO TO, SUB/RETURN, LABEL, (any)'.

   b. If the steps immediately following the 'IF' are operations (e.g. +, ↑) then the branch is cancelled and the operations are executed.

The following program-sequences, 1 through 9, are parts of imaginary programs. Each sequence illustrates a typical combination of keys following an 'IF' instruction; in each case the subsequent activity for 'condition met' (YES) and 'condition not met' (NO) is explained. Only the 'IF x = y' key is used, however the explanation would be the same for any of the four IF keys.

ADVANCED BRANCHING TECHNIQUES

SEQUENCE 1:

4-digit address.

STEP	KEY
0126	IF $x = y$
0127	0
0128	0
0129	5
0130	6
0131	3

YES: Branches to address 0056 without executing step 0131.

NO: Skips four steps (0127 through 0130) and enters 3 into the X-register.

SEQUENCE 2:

Short-form address terminated by an operation.

STEP	KEY
0126	IF $x = y$
0127	5
0128	6
0129	π
0130	X
0131	3

YES: Branches to address 0056 without executing step 0129.

NO: Skips four steps (0127 through 0130) and enters 3 into the X-register.

Notice that, in Sequence 2, steps 0129 and 0130 will never be executed unless a branch is made to them from elsewhere in the program. If such a branch were not required, then steps 0129 and 0130 would normally contain CONTINUE's (no operation). In this case, the main advantage to be gained from using the short-form address is in time-saving; even though this is only a few millionths of a second it could be significant in some situations.

ADVANCED BRANCHING TECHNIQUES

SEQUENCE 3:

Operations.

STEP	KEY
0126	IF $x=y$
0127	π
0128	$\times$
0129	3
0130	$\div$
0131	a

YES: Executes all steps.

NO: Skips four steps (0127 through 0130) and recalls the contents of a to X.

SEQUENCE 4:

Digit entry.

STEP	KEY
0126	IF $x=y$
0127	CONTINUE
0128	1
0129	0
0130	0
0131	1

YES: CONTINUE acts as a 'no operation' but cancels the branch so that the digits (1001) are entered into the X-register.

NO: Skips four steps (0127 through 0130) and enters 1 (step 0131) into the X-register.

SEQUENCE 5:

Labelled address.

STEP	KEY
0126	IF $x=y$
0127	GO TO
0128	LABEL
0129	π
0130	CONTINUE
0131	$\div$

YES: Branches to the address specified by 'Label π'. In this case the CONTINUE serves no function except to step the program counter past address 0130 when the program is being loaded.

NO: Skips four steps (0127 through 0130) and executes the 'divide'.

ADVANCED BRANCHING TECHNIQUES

SEQUENCE 6:

Subroutine-call using a numeric address.

STEP	KEY
0126	IF $x = y$
0127	SUB/RETURN
0128	0
0129	2
0130	4
0131	6
0132	÷

YES: Branches to the subroutine starting at address 0246; the 'return' will be made to address 0132.

NO: Skips four steps (0127 through 0130), enters 6 (step 0131) into the X-register and divides.

SEQUENCE 7:

Subroutine-call using a short-form address (similar to Sequence 6).

STEP	KEY
0126	IF $x = y$
0127	SUB/RETURN
0128	2
0129	4
0130	6
0131	÷

YES: Branches to the subroutine starting at address 0246; the 'return' will be made to step 0131.

NO: Skips four steps (0127 through 0130) and executes the 'divide'.

ADVANCED BRANCHING TECHNIQUES

SEQUENCE 8:

Subroutine-call using a label.

STEP	KEY
0126	IF $x = y$
0127	GO TO
0128	SUB / RETURN
0129	LABEL
0130	π
0131	$\times$

YES: Branches to the subroutine whose starting address is specified by 'Label, π'; the return is made to step 0131.

NO: Skips four steps (0127 through 0130) and executes the 'multiply'.

SEQUENCE 9:

To conditionally return from a subroutine.

STEP	KEY
0126	IF $x = y$
0127	CONTINUE
0128	SUB / RETURN
0129	CONTINUE
0130	CONTINUE
0131	$x \leftarrow ( )$

YES: The CONTINUE in step 0127 cancels the branch and the RETURN in step 0128 is executed.

NO: Skips four steps (0127 through 0130) and executes step 0131.

GENERAL INFORMATION

The Option 004 9810A Calculator contains a printer which is capable of: 'hard-copying' data from the calculator's X-register, logging keyboard operations, and listing the steps of any program stored in the calculator's memory. Each of these printer operations is described in the section entitled PRINTER OPERATION.

If your calculator does not contain Option 004, your nearest -hp- Sales and Service Office can install the printer for you. When ordering the printer for your calculator, order the -hp- 11219A Printer.

INTRODUCTION

One of the many products available for the Model 10 is the -hp- 11211A Printer Alpha Read Only Memory (ROM). With this accessory the entire calculator keyboard is re-defined to enable printing of alphanumeric characters by either keyboard or programmed instructions. The Printer Alpha ROM also complements keylog and list operations by adding abbreviated titles and mnemonic symbols to the information normally printed during either operation (see the foldout in the Appendix).

The Printer Alpha ROM may be purchased at any time and plugged into the calculator through the top cover. For complete details on this and other easy to install ROM's for your Model 10 contact any -hp- Sales and Service Office.

ALPHA PRINTING CAPABILITY

Printer paper may be ordered from the nearest -hp- Sales and Service Office. See the inside rear cover of this manual for office locations. Printer paper is available in 6 roll quantities, -hp- Part No. 9281-0401.

ORDERING PAPER

Printer paper is loaded by using the following procedure (refer to Figure 6-1).

LOADING PAPER

1. Lift the flap attached to the calculator's top cover.
2. Remove and discard the paper core of the previous roll. If the remaining roll is small and a new roll is to be installed, remove the old roll by:
   a. Unrolling it until the core and paper are above the bail;
   b. Grasp the roll firmly and pull up and forward; the paper guide will tear the paper off cleanly.

(continued)

GENERAL INFORMATION

3. Remove the first layer of paper from a new roll.

4. Insert the new roll so that the free paper end is positioned as shown in Figure 6-1. Be sure the bail drops back into place.

5. Press PAPER until the paper advances through the printer mechanism and appears beyond the printer window.

6. Lower the flap.
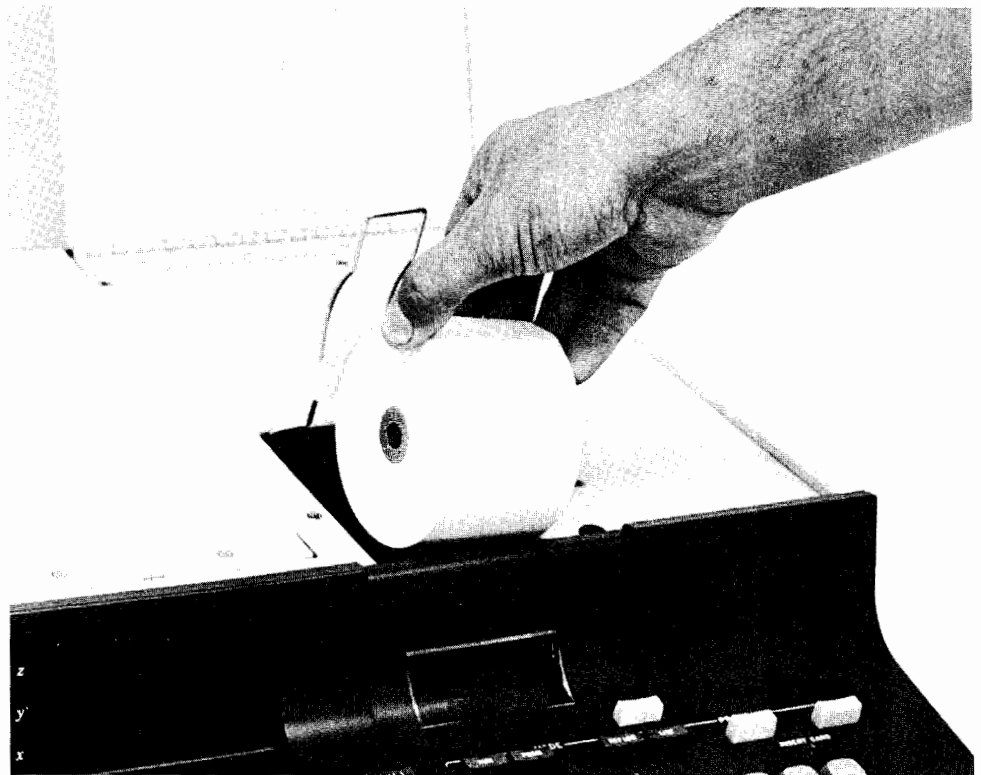
Figure 6-1. Loading Paper

Step 2 in the preceding procedure may be followed in order to remove a partially used roll of printer paper.

NOTE

The printer will not operate if the printer paper roll is empty or if there is no paper loaded in the printer. If the paper roll empties during program operation, the program will be halted at a PRINT/-SPACE Instruction. Provided no print instructions are given from the keyboard or a program, the disabled printer will not affect other calculator operations.

GENERAL INFORMATION

The printer window may be removed, to facilitate cleaning or paper loading, by carefully sliding it up and forward (see Figure 6-2).

Figure 6-2. Printer Window Removal

The following procedure may be used to verify the printer's electrical performance.

NOTE

If your Calculator contains a PRINTER ALPHA ROM, you should also perform the Electrical Inspection in the Printer Alpha ROM Operating Manual.

GENERAL INFORMATION

SWITCH: LINE
           OFF ◗ ON

PRESS   [RUN]   [FLOAT]   (STOP)   (END)   [PRGM]

If the light below the KEYLOG key is not on,

PRESS: [KEY LOG]

NOTE
When the KEYLOG MODE is on press keys slowly enough to enable the printer to log each keystroke.

To load the program:

PRESS: (CLEAR)   (9)   (↑)   (↑)   (1)

PRESS: (0)   (x⇄y)   (÷)   (y→( ))   (b)

PRESS: (↓)   (PRINT SPACE)   (↑)   (b)   (+)   (↓)   (PRINT SPACE)

PRESS: (IF x<y)   (0)   (0)   (1)   (2)

PRESS: (CLEAR x)   (PRINT SPACE)   (PRINT SPACE)   (CHG SIGN)   (ENTER EXP)   (CHG SIGN)

PRESS: (9)   (9)   (PRINT SPACE)   (PRINT SPACE)   (PRINT SPACE)   (PRINT SPACE)   (PRINT SPACE)

PRESS: (PRINT SPACE)   (PRINT SPACE)   (PRINT SPACE)   (PRINT SPACE)   (PRINT SPACE)   (END)

To run or rerun the program:

PRESS: [RUN]   (END)   (CONTINUE)

GENERAL INFORMATION

HARD COPY:

```
                     46
     1.111111111 00
     2.22222222 00
     3.33333333 00
     4.44444444 00
     5.55555555 00
     6.66666666 00
     7.77777777 00
     8.88888888 00
     9.99999999 00
     0.00000000 00

   -1.00000000-99*
```

To obtain a listing of the program previously loaded:

PRESS: (END) [LIST]

HARD COPY:

```
                     46

     0000----------20
     0001----------11
     0002----------27
     0003----------27
     0004----------01
     0005----------00
     0006----------30
     0007----------35
     0008----------40
     0009----------14
     0010----------25
     0011----------45
     0012----------27
     0013----------14
     0014----------33
     0015----------25
     0016----------45
     0017----------52
     0018----------00
     0019----------00
```

(continued)

GENERAL INFORMATION

```
0020-------01
0021-------02
0022-------37
0023-------45
0024-------45
0025-------32
0026-------26
0027-------32
0028-------11
0029-------11
0030-------45
0031-------45
0032-------45
0033-------45
0034-------45
0035-------45
0036-------45
0037-------45
0038-------45
0039-------45
0040-------46
```

To verify keylog operation, compare the hardcopy printed during program entry (printed before the program was run) with the program listing. The printed lists of program steps should contain identical information.

The printouts should be evenly spaced and completely legible. If any characters are not completely printed the program may be rerun and both printouts compared with the example. Should the printer still fail to operate properly, contact your nearest -hp- Sales and Service Office for assistance.

PRINTER OPERATION

This section describes how to use your printer. The printer is capable of performing three important functions: printing data, keylogging, and listing program steps. The examples following each description should be performed in order to obtain a good understanding of printer operation.

Any data in the X-register can be printed by pressing PRINT/SPACE. Since only one instruction is required to print the data, successive PRINT/SPACE instructions will cause the printer to space one line for each added instruction.

PRINTING DATA

PRINT
SPACE

The PRINT/SPACE instruction also terminates the number in the X-register. If the number was not previously terminated, (i.e. just entered from the keyboard) the hardcopy will contain an asterisk (*).

DATA ENTRY SYMBOL
.:.

EXAMPLE:

Print data during keyboard operation.

PRESS: RUN   FIX( )   **3**

Enter data and print.

PRESS: **6**   **2**   **5**   PRINT/SPACE   ↑

PRESS: PRINT/SPACE   PRINT/SPACE   **√x**   PRINT/SPACE

HARD COPY:

     625.000*
     625.000

     25.000

Notice that the data entry symbol accompanies only an unterminated number. Also, notice that a successive PRINT/SPACE instruction causes a line space (space between printed lines).

The PRINT/SPACE instruction is also easily programmed. See the section titled PROGRAMMING WITH THE PRINTER for further information.

**KEYLOG
MODE**

`KEY LOG`

PRINTER OPERATION

The keylog mode is indicated by the light below the KEYLOG key. Pressing KEYLOG will either activate or release the keylog mode.

By specifying KEYLOG (on) during the RUN mode, the printer will print a 'log' (or record) of nearly all programmable operations performed from the keyboard. The printer logs each keystroke by printing the key's keycode. During program operation CONTINUE, STOP, and PAUSE instructions from the keyboard will not be logged. Also, the keystroke following FMT or LBL will not be logged.

NOTE

During KEYLOG operation, the keyboard is momentarily "locked out" while the printer is operating (i.e. the calculator will not receive a keyboard instruction). This requires keyboard entries to be performed slower than when the printer is not in KEYLOG mode.

While programming, the keylog mode will cause the printer to list program steps (logged by keycode) as they are entered from the keyboard.

NOTE

Keycodes for the block of half-keys will be logged even though, without a ROM installed, these keys are not functional.

The following example demonstrates use of the keylog mode and print operations while calculating from the keyboard.

EXAMPLE:

Find the area of a circle if: $A = \pi r^2$, where $r = 7.5$

PRESS: `RUN` `FIX ( )` `3` keylog (on) `CLEAR`

To enter a value for r and compute A:

PRESS: `7` `.` `5` `x²` `↑`

PRESS: `π` `×`

PRINTER OPERATION

DISPLAY: 176.715 $\rightarrow$ y

 3.142 $\rightarrow$ x

To print the resultant data:

PRESS: ↓ | PRINT / SPACE |

HARD COPY:

```
        20
        07
        21
        05
        12
        27
        56
        36
        25
        45
    176.715
```

Notice that every keyboard operation was logged after the KEYLOG mode was specified. The keylog mode is also useful when programming. See the section titled PROGRAMMING WITH THE PRINTER for a further discussion.

PROGRAM LISTING

LIST

By pressing LIST, any program stored in the Model 10 program memory is printed, step-by-step, by the printer. By setting the calculator program counter to the beginning address of the program to be listed and then by pressing LIST, program steps will be listed from that address until an END instruction (keycode 46) is printed.

The program counter may be set by pressing GO TO, followed by the program step address. Thus, pressing

| RUN | | GO TO | | 0 | | 0 | | 3 | 6 | | LIST |

will cause the printer to list program steps beginning at step 0036 and continuing until either an END instruction is printed or STOP is pressed. Also, pressing END and LIST will cause the printer to list from the

**PROGRAM
LISTING**
LIST
(continued)

**PAPER
ADVANCE**
PAPER

PRINTER OPERATION

beginning of the program memory (step 0000). Short-form addressing may also be used to address the program counter. For a discussion of short-form addressing see the chapter titled PROGRAMMING THE MODEL 10.

For an example of program listing, turn to the printer electrical inspection on Page 6-5. If no programming operations have been performed since the electrical inspection was done, a listing of this program can be printed by pressing RUN, END and LIST.

NOTE

Should STOP be pressed to halt a listing operation,
LIST must be pressed again to resume listing with
the next program step.

Printer paper may be advanced by pressing PAPER. This is not a programmable operation.

PROGRAMMING WITH THE PRINTER

This section contains instructions for programming PRINT/SPACE operations. Also included are hints on using the keylog and list features while programming or running a program.

During a program, data may be printed by merely inserting one PRINT/-SPACE instruction in the program. Since the printer is restricted to printing only X-register contents; additional program steps may be required to position the data in the X-register before the PRINT/SPACE instruction is executed.

PROGRAMMING PRINT/SPACE INSTRUCTIONS

The following examples, which could be a part of a program, show possible methods to program the copying of X- and Y-register data.

EXAMPLE 1.

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z
0050	---	---	120.00	75.00	-----
0051	PRINT / SPACE	45	120.00	75.00	-----
0052	↓	25	75.00	-----	-----
0053	PRINT / SPACE	45	75.00	-----	-----

HARD COPY:

```
120.00
75.00
```

EXAMPLE 2.

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z
0050	---	---	120.00	75.00	-----
0051	PRINT / SPACE	45	120.00	75.00	-----
0052	PRINT / SPACE	45	120.00	75.00	-----
0053	$x \gtrless y$	30	75.00	120.00	-----
0054	PRINT / SPACE	45	75.00	120.00	-----

HARD COPY:

```
120.00

75.00
```

Notice that the method in Example 2 is useful if the X-register data in step 0050 must not be lost. Also notice that an extra PRINT/SPACE instruction is added in order to provide a space between copied data.

PROGRAMMING WITH THE PRINTER

**PROGRAMMING
PRINT/SPACE
INSTRUCTIONS
(continued)**

> **NOTE**
>
> If a program containing PRINT/SPACE instructions is run in a calculator without a printer each instruction will halt the program; this enables the operator to copy data which would have been printed. Pressing CONTINUE will resume program operation.

**USE OF
KEYLOG MODE**

The keylog mode is useful when writing short programs, since, when the calculator is in both keylog and program modes, a log of program steps will be printed as they are entered from the keyboard. This enables the programmer to list a short program as he enters it. If branching instructions must be written, he can refer to the log which was just printed in order to find the branching address. The keylog mode is not programmable.

**USE OF
PROGRAM
LISTING**

A fast check of program loading is made possible by obtaining a program listing. The listing feature, although not programmable, is useful when either loading a program or debugging a program while it is in the calculator. See the section titled PRINTER OPERATION for directions on how to obtain a program listing.

PROGRAMMING WITH THE PRINTER

The following program, which is also used in the PROGRAMMING THE MODEL 10 chapter of this manual, will be used to demonstrate how to add PRINT/SPACE instructions to an existing program.

This program will solve the equation: $(A \times B) \div (A + B)$, for values of A and B which are entered into the Y- and X-registers, respectively.

ADDING PRINT/SPACE INSTRUCTIONS

STEP	KEY	KEY CODE	DISPLAY			STORAGE	
			X	Y	Z	a	b
0000	$y \rightarrow (\ )$	40	B	A	·	·	·
0001	a	13	B	A	·	A	·
0002	$x \rightarrow (\ )$	23	B	A	·	A	·
0003	b	14	B	A	·	A	B
0004	$\times$	36	B	A x B	·	A	B
0005	$x \leftarrow (\ )$	67	B	A x B	·	A	B
0006	$+$	33	B	A x B	·	A	B
0007	a	13	A + B	A x B	·	A	B
0008	$\div$	35	A + B	$\frac{A \times B}{A + B}$	·	A	B
0009	a	13	A	$\frac{A \times B}{A + B}$	·	A	B
0010	$\uparrow$	27	A	A	$\frac{A \times B}{A + B}$	A	B
0011	b	14	B	A	$\frac{A \times B}{A + B}$	A	B
0012	END	46	— final display —				

The above program can be modified so that hard copy is printed of both entered variables (A, B) and the resultant $[(A \times B) \div (A + B)]$.

PROGRAMMING WITH THE PRINTER

The modified program below shows one method of adding the required PRINT/SPACE instructions. Notice that each instruction is inserted after the required data is in the X-register. Also, notice that an extra PRINT/-SPACE instruction is added (step 0006) in order to leave a space on the hard copy between the entered data and the resultant data.

STEP	KEY	KEY CODE	X	Y	Z	a (STORAGE)	b (STORAGE)
0000	PRINT / SPACE	45	A	---	---	---	---
0001	$x\rightarrow(\ )$	40	A	---	---	---	---
0002	a	13	A	---	---	A	---
0003	↑	27	A	A	---	A	---
0004	STOP	41	B (enter)	A	---	A	---
0005	PRINT / SPACE	45	B	A	---	A	---
0006	PRINT / SPACE	45	B	A	---	A	---
0007	$x\rightarrow(\ )$	40	B	A	---	A	---
0008	b	14	B	A	---	A	B
0009	×	36	B	A × B	---	A	B
0010	$x\leftarrow(\ )$	67	B	A × B	---	A	B
0011	+	33	B	A × B	---	A	B
0012	a	13	A + B	A × B	---	A	B
0013	÷	35	A + B	$\frac{A \times B}{A + B}$	---	A	B
0014	↓	25	$\frac{A \times B}{A + B}$	---	---	A	B
0015	PRINT / SPACE	45	$\frac{A \times B}{A + B}$	---	---	A	B
0016	↑	27	$\frac{A \times B}{A + B}$	$\frac{A \times B}{A + B}$	---	A	B
0017	a	13	A	$\frac{A \times B}{A + B}$	---	A	B
0018	↑	27	A	A	$\frac{A \times B}{A + B}$	A	B
0019	b	14	B	A	$\frac{A \times B}{A + B}$	A	B
0020	END	46	B	A	$\frac{A \times B}{A + B}$	A	B

The program can be run to ensure correct insertion of the PRINT/SPACE instructions.

To load the modified program:

PRESS: [RUN] (END) [PRGM]

PRESS: (keys of the modified program beginning with step 0000).

PROGRAMMING WITH THE PRINTER

To enter values for A and B (A=40.0; B=20.0) and run the program:

PRESS: [RUN] [FIX ( )] (5) (CLEAR) (END)

PRESS: (4) (0) (CONTINUE)

PRESS: (2) (0) (CONTINUE)

DISPLAY: 13.33333 → z
40.00000 → y
20.00000 → x

HARD COPY:

```
40.00000*
20.00000*

13.33333
```

The above hard copy contains both entered variables and the result in the same order as they appear during program operation. Also notice the data entry symbol (*) indicates the unterminated entered data.

The programmed 'space' instruction (step 0006) between variables and result provides a more readable printout. The space instruction may also be programmed at either end of the program's printout in order to separate groups of data, or to advance the printed data above the printer window to facilitate hard copy removal. To advance the last printed line above the print window requires at least 8 successive PRINT/SPACE instructions after the last data printing operation.

LINE SPACE INSTRUCTIONS

If the modified program does not operate as expected, the first step in debugging would be to check program loading. A program listing is a fast, accurate check to determine if the program was correctly loaded.

APPENDIX

CHARACTERISTICS

TEMPERATURE RANGE: 0°C (32°F) to 45°C (113°F)

VAPOR PRESSURE: Not to exceed 1.03 pounds per square inch.

RANGE OF CALCULATION: $(\pm)10^{-98}$ to $(\pm)9.99999999999 \times 10^{98}$

ERROR SUMMARY: Summary of maximum errors of the functions in the basic calculator. Errors of functions in the plug-in ROM'S are listed in the individual ROM manuals.

a. Errors are expressed as absolute errors:

Absolute error $= | f(X) - \hat{f}(X) |$

where $f(X)$ = exact value
$\hat{f}(X)$ = calculated value
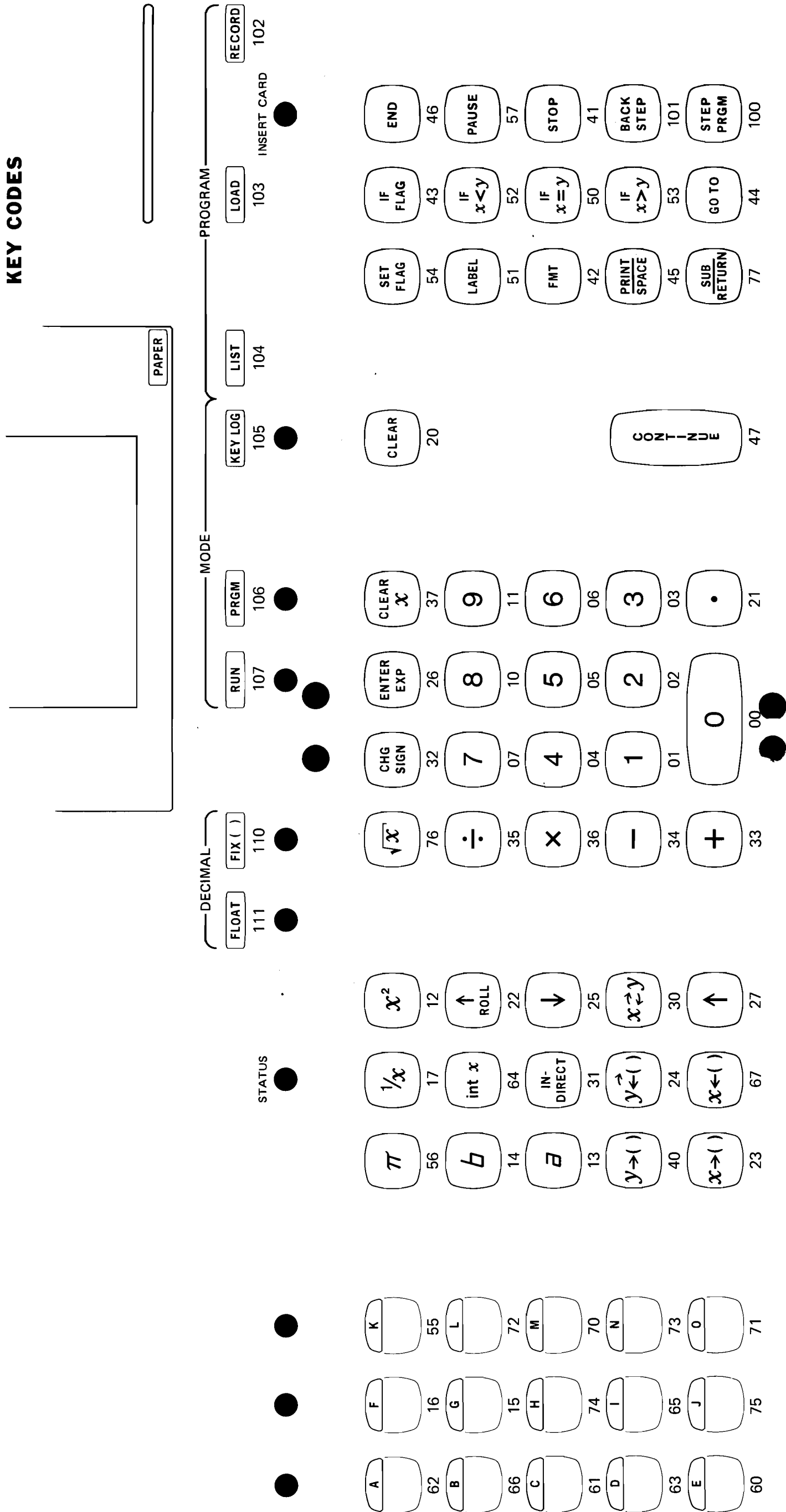
b. One 'visible count' is one count in the tenth digit.

1. ADD, SUBTRACT $\pm5 \times 10^{-11} \times 10^{(\text{larger addend exponent})}$

2. MULTIPLY, X^2 $\pm1/2$ visible count

3. DIVIDE, 1/X $\pm1/2$ visible count

4. SQUARE ROOT $\pm1/10$ visible count

KEY CODE	MNEMONIC	KEY	KEY CODE	MNEMONIC	KEY	KEY CODE	MNEMONIC	KEY	KEY CODE	MNEMONIC	KEY	KEY CODE	MNEMONIC	KEY
00	0	0	15	G	G	32	CHS	CHG SIGN	47	CNT	CONTINUE	64	INT	int x
01	1	1	16	F	F	33	+	$+$	50	X=Y	IF $x=y$	65	I	I
02	2	2	17	1/X	$1/x$	34	−	$-$	51	LBL	LABEL	66	B	B
03	3	3	20	CLR	CLEAR	35	DIV	$\div$	52	X<Y	IF $x<y$	67	XFR	$x\leftarrow(\ )$
04	4	4	21	▪	•	36	X	$\times$	53	X>Y	IF $x>y$	70	M	M
05	5	5	22	RUP	ROLL ↑	37	CLX	CLEAR x	54	SFL	SET FLAG	71	O	O
06	6	6	23	XTO	$x\rightarrow(\ )$	40	YTO	$y\rightarrow(\ )$	55	K	K	72	L	L
07	7	7	24	YE	$y\leftarrow(\ )$	41	STP	STOP	56	π	π	73	N	N
10	8	8	25	DN	↓	42	FMT	FMT	57	PSE	PAUSE	74	H	H
11	9	9	26	EEX	ENTER EXP	43	IFG	IF FLAG	60	E	E	75	J	J
12	XSQ	x^2	27	UP	↑	44	GTO	GO TO	61	C	C	76	√	$\sqrt{x}$
13	a	a	30	XEY	$x\rightleftarrows y$	45	PNT	PRINT SPACE	62	A	A	77	S/R	SUB RETURN
14	b	b	31	IND	INDIRECT	46	END	END	63	D	D			

The mnemonics are printed by the Option 004 Printer if the -hp- 11211A Printer Alpha ROM is installed.

APPENDIX

KEY CODES

PROGRAM

LOAD 103 | INSERT CARD | RECORD 102

END 46	IF FLAG 43	SET FLAG 54
PAUSE 57	IF x<y 52	LABEL 51
STOP 41	IF x=y 50	FMT 42
BACK STEP 101	IF x>y 53	PRINT/SPACE 45
STEP PRGM 100	GO TO 44	SUB RETURN 77

MODE

KEY LOG 105 | PRGM 106 | RUN 107 | LIST 104 | PAPER

CLEAR 20

CONTINUE 47

CLEAR x 37	ENTER EXP 26	CHG SIGN 32	$\sqrt{x}$ 76
9 11	8 10	7 07	$\div$ 35
6 06	5 05	4 04	$\times$ 36
3 03	2 02	1 01	$-$ 34
. 21	0 00		$+$ 33

DECIMAL

FLOAT 111 | FIX ( ) 110

x^2 12	$1/x$ 17	π 56
$\uparrow$ ROLL 22	int x 64	b 14
$\rightarrow$ 25	IN-DIRECT 31	a 13
$x \rightleftarrows y$ 30	$y \leftarrow ( )$ 24	$y \rightarrow ( )$ 40
$\leftarrow$ 27	$x \leftarrow ( )$ 67	$x \rightarrow ( )$ 23

STATUS

K 55	F 16	A 62
L 72	G 15	B 66
M 70	H 74	C 61
N 73	I 65	D 63
O 71	J 75	E 60

Refer to Page 5-11 for a description of the key codes.

Keys with 3-digit codes are not programmable.