HEWLETT-PACKARD CALCULATOR

Model 9100A

Operating and Programming

HEWLETT hp PACKARD

WARRANTY AND ASSISTANCE

The Hewlett-Packard 9100A Calculator is warranted against defects in material and workmanship. This warranty applies for ninety (90) days from date of delivery. We will repair or replace components which prove to be defective during the warranty period. No other warranty is expressed or implied. We are not liable for consequential damages.

For any assistance contact your nearest Hewlett-Packard Sales and Service Office. Addresses are provided on the last pages of this manual.

P. O. Box 301, Loveland, Colorado, 80537 U. S. A.

OPERATING AND PROGRAMMING

HEWLETT-PACKARD 9100A

TABLE OF CONTENTS

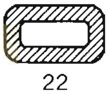
TABLE OF CONTENTS

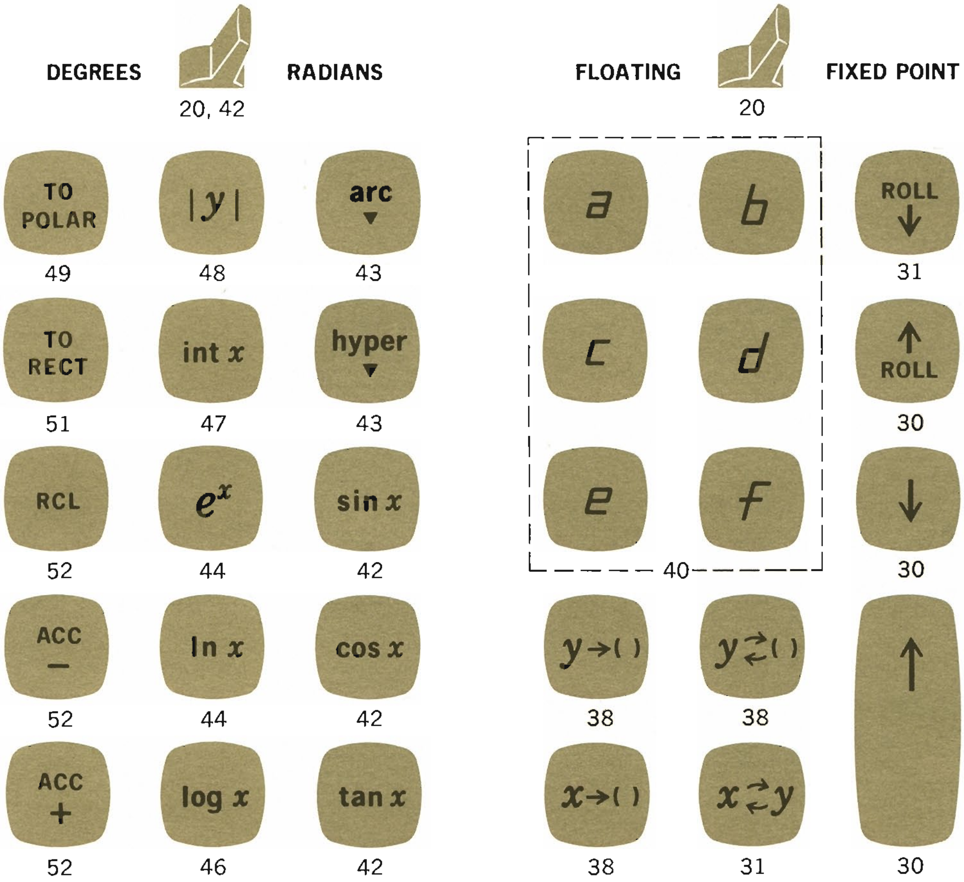
TABLE OF CONTENTS

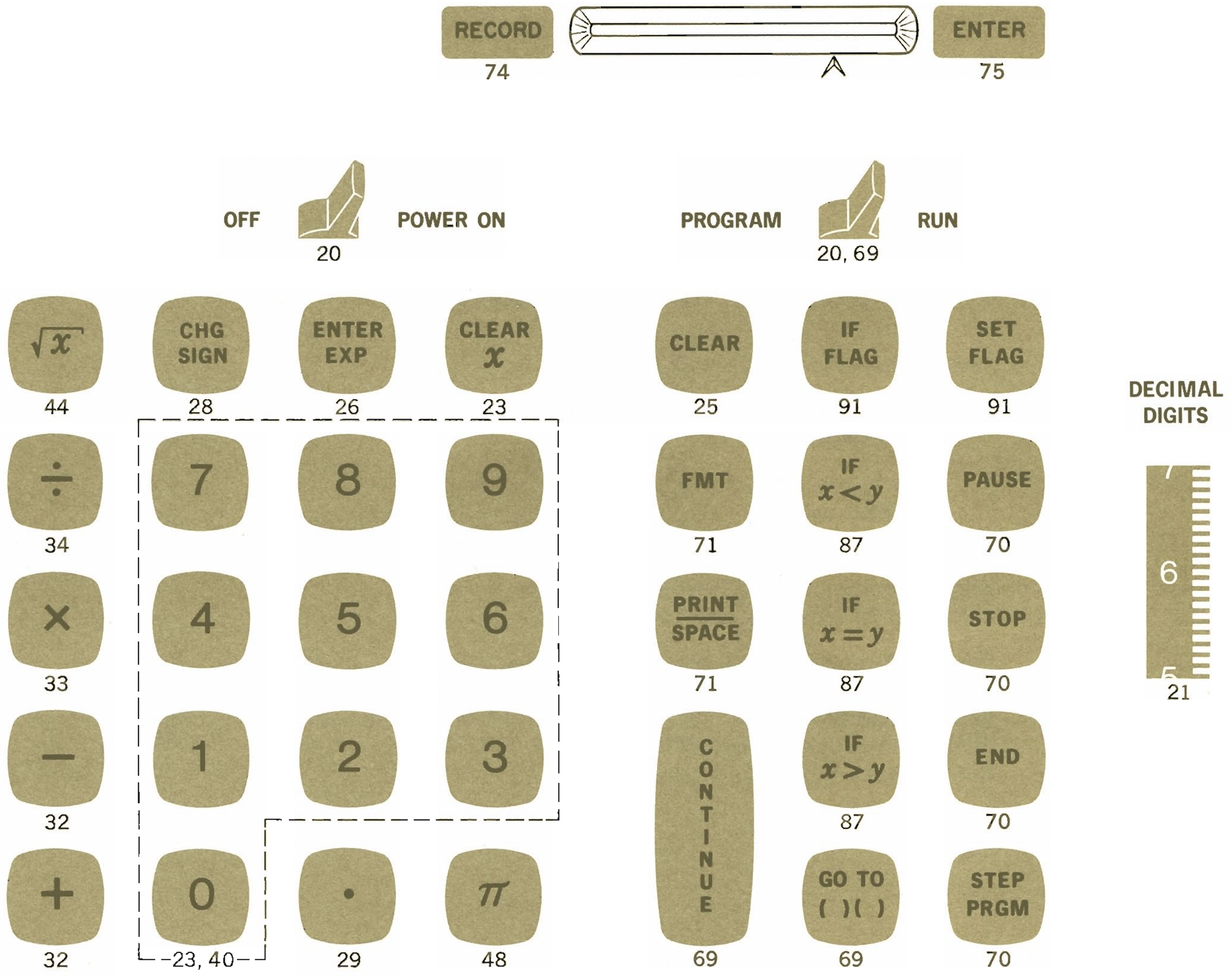
TABLE OF CONTENTS

APPENDIX

PAGE INDEX OF KEYS AND SWITCHES







Numbers below the switches and keys indicate the page(s) on which the bold-print description of the switch or key appears.

HOW TO USE THIS MANUAL

This manual may be used in two ways;

1. as a textbook from which to learn how to use and program your calculator, and

2. as a 'quick-reference', to determine what a particular key or switch will do and the purpose for which it may be used.

As a textbook, the manual progresses in a learning sequence which presupposes no previous experience with this type of calculator or with programming.

For 'quick-reference' the key index on Page vi is used to locate the required key or switch in the text. Each key is briefly described in bold print. This is followed by a more detailed explanation with examples which illustrate what the key does and the purposes for which it may be used.

This section contains general information about the Model 9100A, its accessories, servicing information, etc. Also included is the turn-on procedure and a method for checking the performance of your calculator.

DESCRIPTION

Your Hewlett-Packard 9100A Calculator is an easy-to-use, yet scientific, desk-top calculator. It is capable of performing operations ranging from the very simple to the very complex, such as are encountered in educational, scientific and engineering problems. Trigonometric, logarithmic and mathematical functions are performed with single key strokes.

The 9100A is programmable: its computer-like memory enables the calculator to store instructions and data for repetitive and iterative solutions; conditional branching, which allows the calculator to automatically make decisions while performing a program, provides complete programming capability. Programming the calculator is simple; the keyboard operations become the program instructions and no special programming language is required.

Programs can be recorded on the magnetic cards supplied with the calculator. Recorded programs can be reinserted into the calculator at a later date, thus eliminating repetition of the time-consuming step-by-step entry. A pull-out instruction card is located at the front of your calculator, under the keyboard; the operation of each key is briefly explained on this card.

CAUTION

PLEASE DO NOT APPLY OPERATING POWER TO YOUR 9100A CALCULATOR UNTIL YOU HAVE READ THE TURN-ON INSTRUCTIONS ON PAGE 5 OF THIS MANUAL.

ACCESSORIES EQUIPMENT SUPPLIED

The accessories and equipment supplied with each Model 9100A are listed in Table 1.

TABLE 1

ACCESSORIES／EQUIPMENT SUPPLIED

PART NO.	QUANTITY	DESCRIPTION
09100-90001	2	Operating and Programming Manual
09100-90002	1	Program Library
09100-90003	1	Program Pad
09100-90004	1	Magnetic Program Card containing the Diagnostic Program*
4040-0350	1	Dust Cover
5060-5919	1	Magnetic Program Card Container with Ten Program Cards
8120-0078	1	Power Cord

*Located in the pocket of the rear cover of the Program Library.

Bulk quantities of Magnetic Program Cards are available; prices are obtainable on request.

PROGRAM PADS

Extra program pads may be purchased. Two types of pad are available (an example of each is shown in the Appendix at the back of this manual):

LONG FORM (Part No. 09100-90003) - this form is recommended for the beginning programmer.

SHORT FORM (Part No. 09100-90023) - recommended for the experienced programmer.

Either form of pad may be purchased in packets of five pads as follows:

Packet of 5 long form pads - Part No. 09100-90000
Packet of 5 short form pads - Part No. 09100-90020

PROGRAM LIBRARY

The Program Library (listed in Table 1) furnished with your 9100A Calculator includes programmed solutions to many practical problems in a wide range of business, scientific and engineering fields. It serves both as an illustration of programming techniques and as a source of ready-to-use programs.

'KEYBOARD' PUBLICATION

Calculator owners also receive the Hewlett-Packard 'KEYBOARD', a periodic publication which provides updating information for programs and a forum for the exchange of programs by 9100A users.

Please complete the mailing card (located in the pocket at the back of the Program Library) and return it to us; this will ensure that your name and address are included in our 'KEYBOARD' mailing list.

9100A OPTIONS

Calculators may also be purchased with the pull-out instruction card printed in a language other than English. There is no price difference between the Standard and Optional calculators.

9100A, Standard:	card printed in English
9100A, Option 001:	card printed in French
9100A, Option 002:	card printed in German
9100A, Option 003:	card printed in Italian
9100A, Option 004:	card printed in Spanish

PERIPHERAL EQUIPMENT

The following peripheral equipments (prices available on request) will greatly enhance the versatility of your 9100A Calculator. (These are also compatible with the 9100B Calculator).

MODEL 9120A PRINTER: Attaches to the top of the calculator and can be added at any time. Prints any combination of the three displays which appear on the screen of your calculator; also lists programs contained in the calculator memory.

MODEL 9125A X-Y RECORDER: Provides permanent, accurate graphic solutions of problems solved by the Calculator; controlled either manually or by programming the calculator.

MODEL 9150A MONITOR SCOPE: Large 19" CRT screen provides a calculator display easily seen from long distances; especially suitable for educational and lecture-room demonstrations (this peripheral requires that a modification be made to the standard calculator).

MODEL 9160A OPTICAL CARD READER: Allows entry into the calculator of programs penciled onto a special card; a useful educational tool enabling fast checking of students' programs.

Other peripheral equipments will be available in the future.

INTERFACE MANUAL

The '9100A/B Input/Output Manual' describes the calculator's output signals and required input signals to enable you to design your own interfacing equipment. Manuals may be purchased from your local Sales and Service Offices (listed at the back of this manual).

SERVICE CONTRACTS

Service contracts are available for the 9100A Calculator. For further information contact your local Hewlett-Packard Sales and Service office; office locations are listed at the back of this manual.

INITIAL INSPECTION

The calculator was carefully inspected, both mechanically and electrically, before shipment. It should be physically free of mars or scratches and in perfect electrical order upon receipt. Carefully inspect your calculator for physical damage caused in transit and check for the accessories listed in Table 1 (Page 2). Also, check the electrical performance of the calculator as described on Page 5 ; do not, however, make this check until you have completed the TURN-ON procedure given on Page 5.

If the calculator is damaged or if an electrical deficiency is indicated, file a claim with the carrier or refer to the warranty on the inside front cover of this manual.

CAUTION

USE A SOFT CLOTH TO CLEAN THE DISPLAY WINDOW. ANY ABRASIVE MATERIAL WILL SCRATCH THE SURFACE.

POWER REQUIREMENTS

The Model 9100A Calculator requires either 115 or 230 V ac, ±10%, 50 to 60 Hertz or 400 Hertz; power requirements are less than 70 watts. A slide switch, located on the rear of the calculator, selects either 115 V or 230 V operation.

NOTE

The Model 9100A Calculator should not be operated in a temperature environment outside the range of 0°C. to 55°C. (32°F. to 131°F.).

GROUNDING REQUIREMENTS

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

CAUTION

DO NOT APPLY OPERATING POWER TO THE 9100A CALCULATOR UNLESS THE LINE VOLTAGE SWITCH ON THE REAR PANEL IS IN THE PROPER POSITION. OTHERWISE, DAMAGE TO THE POWER TRANSFORMER MAY RESULT.

TURN-ON PROCEDURE

With the calculator disconnected from the ac power source, slide the line voltage switch, located on the rear panel, to the position where the line voltage to be used (115 or 230) appears on the switch. Connect the calculator to the ac power source. Switch the OFF - POWER ON switch, located above the keyboard, to the POWER ON position; the three register designators, to the right of the display window, will light, indicating that power has been applied to the calculator.

ELECTRICAL INSPECTION

A magnetic program card, containing the Diagnostic Program, is in the pocket on the inside rear cover of the Program Library; this program tests the electrical performance of the calculator. (The steps of the Diagnostic Program are included in the Miscellaneous Section of the Program Library.) To enter the program into the calculator:

set the switches (located above the keyboard) to these positions:

RADIANS FIXED POINT POWER ON RUN

Revolve the DECIMAL DIGITS wheel (to the right of the keyboard) until 9 appears uppermost:

NOTE

If there is no display after a warm-up of 20 seconds:
If there is a flashing display:
If no key works:

PRESS: STOP STOP

and continue with the procedure.

Press these keys in the order shown (left to right):

PRESS: GO TO ( )( ) 0 0

or, alternatively,

PRESS: END

Insert the Diagnostic card, with either the A or B arrow pointing

ELECTRICAL INSPECTION
CONTINUED

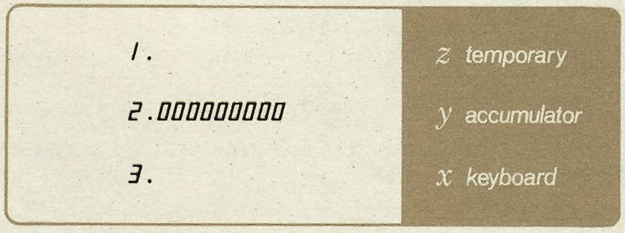
down and the PRINTED SIDE TOWARD THE KEYBOARD, into the card reader slot located between the 'RECORD' and 'ENTER' keys.

Be sure the card is fully inserted with the printed side facing the keyboard.

PRESS: ENTER (hold the key pressed until the card is partially ejected from the card reader; if it does not eject then the card is not fully inserted). Remove the card.

PRESS: CONTINUE to run the program.

Correct operation of the 9100A Calculator is indicated by the display, shown below, momentarily appearing on the screen. Once is sufficient to test the calculator; however, the program will continue to be run, with the display flashing on the screen every few seconds, until the STOP key is pressed. To restart the program press the END and CONTINUE keys.



```
1.                z temporary
2.000000000       y accumulator
3.                x keyboard
```

The calculator is not operating correctly:

if there is no display at all after about 5 seconds,
if the display appears but remains fixed,
if there is a display, fixed or flashing, different from the display shown above.

If one of these should occur, first make sure that the four switches (in particular, the DEGREES-RADIANS switch), located above the keyboard, and the DECIMAL DIGITS wheel are correctly set; then carefully repeat the entire procedure to ensure that no error was made the first time.

If the calculator still will not operate then refer to the warranty, on the inside front cover of this manual, for assistance from your nearest Sales and Service Office.

INTRODUCTION

This section contains general information about the Model 9100A which you will need in order to operate the calculator effectively. Terms are explained, wherever they first appear. The examples given in this section are not intended to teach the keyboard, but to illustrate the points being made in the text; by performing the examples, however, you will develop a 'feel' for your calculator.

CHARACTERISTICS

Your 9100A is an extremely powerful instrument with a wide range of capabilities. It may be efficiently used to perform calculations ranging from the simple adding-machine type of calculation up to highly sophisticated scientific computations. The calculator is particularly valuable in solving problems which are complex but which involve only a moderate amount of data, the type of calculation which previously required a computer; in this respect, because of the wide range of numbers which can be handled at any one time (from 1 x 10^{-98} to 9.999 999 999 x 10^{99}), the Calculator can outperform many computers.

Despite its tremendous capability the calculator remains very simple to operate. An unskilled operator can be taught, in minutes, how to use the calculator as an adding-machine. Program operation is simple enough that the same operator can be easily shown how to enter a program into the calculator and then run the program.

ROM

The 9100A uses two memory systems. One is a unique Hewlett-Packard Read-Only-Memory (ROM). The ROM contains all the subroutines ('wired-in' programs) necessary to execute the instructions and calculate the many functions which can be called from the keyboard; these subroutines are fixed and cannot be changed in any way by the person operating the calculator.

MAGNETIC-CORE MEMORY

The second memory is a magnetic-core memory; this adds to the calculator the capability of storing data and of being programmed to automatically process that data.

MAGNETIC-CORE MEMORY

CONTINUED

The memory consists of 19 'registers'. In the diagrammatic sketch of Figure 1 these are the horizontal rows designated 0 through *f* and x, y, and z. Each register is divided into 14 locations known as 'characters'; these are designated 0 through *d*.

As shown in Figure 1, the first fourteen registers (0 through *d*) may be used to store either data or program steps. The next two registers, *e* and *f*, are used for data storage only and the remaining three registers, x, y, and z, contain the information which is displayed on the calculator screen.

When program steps are stored in the memory, each character in the 0 through *d* registers can contain one program step (one key pressed). This makes a maximum of 196 (14 x 14) program steps stored at any one time.

When a data number is stored, all 14 characters of any one register are required. This means that each time a data number is stored in the memory, the maximum possible number of program steps (196) is reduced by 14 (one complete register). Data and program steps cannot both be stored at the same time in any one register.

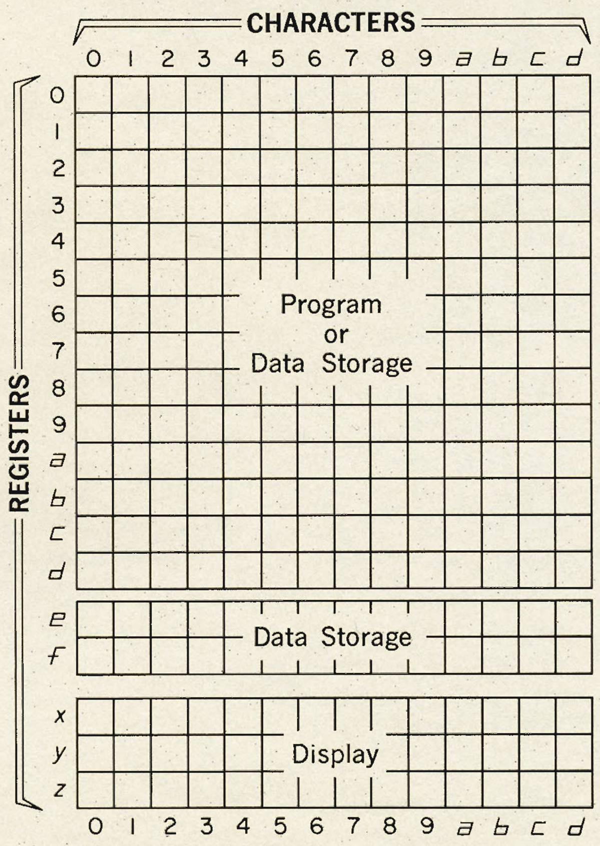


Figure 1. Magnetic-Core Memory

Regardless of the number of digits which a particular data number has, the calculator stores the data number as a 12-digit number with a 2-digit exponent, one digit per character, hence the need for a complete register. (See also FLOATING AND FIXED POINT on Page 11 and GUARD DIGITS on Page 12.)

DISPLAY

x *keyboard* - this register displays numbers as they are entered from the keyboard, one digit at a time.

> **NOTE**
>
> If the calculator is currently running a program, press STOP to stop the program.

EXAMPLE:

Enter 6.34 in the X register.

Set the switches above the keyboard to the following positions:

SWITCH: **FIXED POINT**

SWITCH: **RUN**

SET DECIMAL DIGITS: 2

Press the keys in the order shown (left to right).

PRESS: CLEAR 6 • 3 4

DISPLAY: 6.34 → x

("6.34 appears in the X register")

DISPLAY
CONTINUED

y accumulator - the result of an arithmetic operation on two numbers, one in the X register (the operator), and one in the Y register (the operand), appears in the Y register.

EXAMPLE:

6.34 ÷ 2 = 3.17
(with 6.34 → x)

PRESS: ↑ 2 ÷

DISPLAY: 3.17 → y
2. → x

z temporary - used for 'temporary' storage; the number is stored in the Z register while arithmetic operations are performed on two other numbers in the X and Y registers, then the number in Z can be returned to Y and included once again in the calculation. Use of the Z register saves using the program and data storage registers.

EXAMPLE:

$\frac{20}{3 + 2} = 4$ (20 will be stored in Z while 3 and 2 are added)

PRESS: CLEAR 2 0 ↑

DISPLAY: 20. → y

PRESS: 3 ↑

DISPLAY: 20. → z, 3. → y

PRESS: 2

DISPLAY: 2. → x

X and Y can now be added without affecting Z;

PRESS: +

DISPLAY: 5. → y

Z is now returned to Y for the division to be performed.

PRESS: ↓

DISPLAY: 20. → y , 5. → x

PRESS: ÷

DISPLAY: 4.00 → y

FLOATING AND FIXED POINT

A powerful feature of your 9100A Calculator is the tremendous range of numbers (1×10^{-98} to $9.999\ 999\ 999 \times 10^{99}$) which it accurately handles without any special attention. You do not have to worry about locating the decimal point on the machine for the greatest accuracy of calculation; no matter which 'format' is chosen, FLOATING or FIXED POINT, and no matter where the DECIMAL DIGITS wheel is set, the calculator always calculates with the same high degree of accuracy. The choice of 'format' affects only the display, not the calculation (which is always in floating point).

The calculator stores all numbers and performs all arithmetic operations in 'floating point'. A FLOATING POINT number is nothing more than a number written in a convenient shorthand. The number consists of two parts, the 'principle value' and the 'exponent'. The 'principle value' consists of the digits of the number with the decimal point placed after the first digit (e.g. 1.2345). The 'exponent', which is written as a positive or negative power of 10, represents the number of places, and the direction, that the decimal point should be moved to express the number as a FIXED POINT number. The following examples illustrate the difference between 'floating' and 'fixed point'.

	FIXED POINT		**FLOATING POINT**		
			(PRINCIPLE VALUE)		(EXPONENT)
Example (a)	1234.5	=	1.2345	x	10^3
Example (b)	.0012345	=	1.2345	x	10^{-3}
Example (c)	1.2345	=	1.2345	x	10^0

FLOATING AND FIXED POINT

CONTINUED

The exponent appears as a two-digit number to the right of the display; if the exponent is negative, then the minus sign appears.

Using the example (b) shown above:

SWITCH: **FLOATING**

SWITCH: **RUN**

PRESS: 1 2 3 4 5

PRESS: ENTER EXP CHG SIGN 3

DISPLAY: 1.234 5 -03 → x

exponent

GUARD DIGITS

Although up to 10 digits (plus the 2 exponent digits) are displayed, all numbers are stored and used in the calculator with 12 significant digits, plus the exponent. The two extra digits are the 'guard digits' (these should not be confused with the exponent digits). The purpose of these guard digits is to maintain greater than 10-place accuracy during calculation and to round the least significant digit displayed when 'fixed point' display is selected. In 'floating point' the last displayed digit is not rounded.

The following example illustrates the use of the guard digits in maintaining accuracy.

EXAMPLE:

divide 6.000 000 001 by 3 and then multiply the result by 3.

SWITCH: **FIXED POINT**

SWITCH: **RUN**

SET DECIMAL DIGITS: 9

(Press these keys in the order shown, left to right, top line then second line, etc.)

PRESS: CLEAR 6 •

PRESS: 0 0 0

PRESS: 0 0 0

PRESS: 0 0 1

DISPLAY: 6.000000001 → x

PRESS: ↑ 3 ÷

DISPLAY: 2.000000000 → y

if there were no guard digits then, when the X (multiply) key was pressed, the display would show 6.000000000 and not the 6.000000001 which was originally inserted;

PRESS: ×

DISPLAY: 6.000000001 → y

accuracy has not been lost.

If you wish to view the contents of the guard digits at any time, this is easily done.

EXAMPLE:

if the π key is pressed, pi is inserted with 12 digits, (although only the first ten digits are displayed); to view the last two digits, which are contained in the guard digits, perform the following:

SWITCH: **FLOATING**

SWITCH **RUN**

GUARD DIGITS
CONTINUED

PRESS: π ↑

DISPLAY: $3.141\ 592\ 653\ \ \underline{00}$ → y

exponent

By subtracting the first two digits (3.1) from π, the contents of the guard digits can be displayed:

PRESS: 3 • 1 —

DISPLAY: $4.159\ 265\ 3\underline{60}\ \ -02$ → y

guard digits

OVERFLOW AND UNDERFLOW

In 'fixed point' mode, if the number in any one display register 'overflows' (i.e. becomes too large for a particular setting of the DECIMAL DIGITS wheel) then the display of that register automatically changes to 'floating point'.

EXAMPLE:

SWITCH: **FIXED POINT**

SWITCH: **RUN**

SET DECIMAL DIGITS: 7

PRESS: CLEAR 1 2 3 •

PRESS: 4 5 6 7

PRESS: 8 9 8

DISPLAY: 123.4567898 → x

Notice that with the DECIMAL DIGITS wheel set to 7 there are 7 digits to the right of the decimal point and 3 to the left.

SET DECIMAL DIGITS: 8

DISPLAY: 1.234 567 898 02 → x

When the DECIMAL DIGITS wheel is switched to 8, the calculator can display only two digits to the left; the number becomes too large for the register and 'overflow' occurs, the X register reverts automatically to 'floating point' and no digits are lost.

If the number 'underflows' (i.e. becomes too small for the DECIMAL DIGITS setting) then the display does not revert to floating point. Zeros will appear in the display but the number is still contained in the calculator in floating point. As an example of 'underflow' enter 4 x 10^{-6} (.000004) into the calculator:

SWITCH: **FIXED POINT**

SET DECIMAL DIGITS: 6

PRESS: CLEAR x · 0 0 0

PRESS: 0 0 4

DISPLAY: .000004 → x

SET DECIMAL DIGITS: 5

DISPLAY: .00000 → x

the number underflows and zeros appear in the X register.

CONTINUED

OVERFLOW AND UNDERFLOW

CONTINUED

SWITCH: **FLOATING**

DISPLAY: 4. -06 → x

accuracy is not lost even though the display did not revert to floating point.

In either case, 'overflow' or 'underflow', the digits are not lost because the calculator always operates in 'floating point' regardless of the display mode.

RANGE

The range of the 9100A is from 9.999 999 999 x 10^{99} to 1 x 10^{-98}. Even though 9.999 999 999 x 10^{-99} can be entered and stored in the calculator the lowest effective range is still 1 x 10^{-98}; the following example demonstrates this.

EXAMPLE:

$(2 \times 10^{-50}) \times (5 \times 10^{-50}) = 10 \times 10^{-100} = 1 \times 10^{-99}$

SWITCH: **FLOATING**

SWITCH: **RUN**

PRESS: CLEAR 2 ENTER EXP CHG SIGN

PRESS: 5 0 ↑

PRESS: 5 ENTER EXP CHG SIGN 5 0

PRESS: × (the Y register 'overflows' and displays zero; in this case when 'overflow' occurs the information is lost.)

ILLEGAL OPERATIONS

If a mathematically illegal operation, such as division by 0, is performed, then a light, located at the left of the display, is 'set' (lights up). If the calculator is running a program when the illegal operation occurs the light will remain lit, but the program will continue to run. The light can be 'reset' (switched off) by pressing any key on the keyboard. Illegal operations are listed on Page 22.

EXAMPLE:

division by 0 (zero)

SWITCH: **RUN**

PRESS: CLEAR 1 ↑ 0 ÷

the light 'sets'.

Notice that the operation is performed even though it is 'illegal'; the result, 9. 999 999 999 x 10^{99}, which is infinity as far as the calculator is concerned, appears in the Y register.

PRESS: any key

the light 'resets'.

NOTE

There are (apparent) exceptions to this when a key is pressed which then performs another illegal operation; although the light does in fact 'reset' it will be immediately 'set' by the new illegal operation.

INTRODUCTION

This section describes the function of all switches and keys, except those used for programming only, which are described under programming. Each key is fully described, in a logical 'learn-the-keyboard' sequence, in two parts - a brief explanation in bold print, which may be used as a quick reference, followed by more detailed information with examples. The keyboard presentation on Page vi is an index of the pages on which the 'bold print' explanations appear.

STEP-SAVING

Whenever possible, you should look for the shortest method of working a problem in order to reduce the number of steps required. This will save time when making calculations and will prove invaluable when writing programs, where the number of steps may be limited. Several step-saving hints are included with the examples in this manual.

KEYING INSTRUCTIONS

In the examples, the keying instructions will be given in either one of two formats, depending on which is the more suitable presentation for the particular example:

Format (a)

PRESS: CLEAR 2 ENTER EXP CHG SIGN

PRESS: 5 0 ↑

in which the keys are pressed in the order shown (from left to right) top line, then second line, etc.

Format (b)

STEP	KEY
1	CLEAR
2	2
3	ENTER EXP
4	CHG SIGN

STEP	KEY
5	5
6	0
7	↑

in which the keys are pressed in STEP sequence 1, 2, 3, 4, etc. Use of the Format (b) will also assist present 'non-programmers' to make the transition to program writing, as this is a simplified way of writing a program.

SWITCHING INSTRUCTIONS

Switching instructions remain as before; for example

SWITCH: **RUN**

means "set the PROGRAM - RUN switch to the RUN position."

> **NOTE**
>
> If a switch is not shown, it may be left in any position; it will, however, be assumed throughout this section that the PROGRAM - RUN switch is always in the RUN position.

The setting of the DECIMAL DIGITS wheel will appear as (for example):

SET: DECIMAL DIGITS to 5

OTHER DEFINITIONS

Statements such as "9.00 appears in the Y register" will be shown as:

DISPLAY: 9.00 → *y*

'Enter' implies 'insertion' unless stated otherwise.

EXAMPLE:

"Enter a program" means "insert a program into the calculator" (irrespective of the method used to enter the program).

SWITCHES

OFF POWER ON

Applies ac line power to the calculator.

Lights the register designators, located to the right of the display; these also act as a pilot light to indicate that the calculator is switched on.

> **NOTE**
>
> See Page 4 for power requirements and turn-on instructions.

DEGREES RADIANS

Selects the type of unit, degrees or radians, to be used and displayed in calculations involving the trigonometric functions; angles must be entered, and will be displayed, in the units selected.

DEGREES (RADIANS) are not automatically converted to RADIANS (DEGREES) by resetting the switch; to make this conversion refer to Page 42.

PROGRAM RUN

Selects the mode of calculator operation.

PROGRAM MODE: used when entering a program from the keyboard and when editing a program (explained in the programming section of this manual).

RUN MODE: used when performing calculations, recording a program on a magnetic card, entering a program from a magnetic card, running a program and addressing the program counter (the programming functions are explained in the programming section of this manual).

FLOATING FIXED POINT

Selects the display mode (See also Page 11).

FLOATING DECIMAL POINT: numbers of up to (and including) ten digits are displayed, plus the two digit exponent (which indicates the power of ten multiplier). Non-significant zeros are blanked.

EXAMPLE:

NUMBER

$12{,}345.67898 = 1.234567898 \times 10^4$

DISPLAY

1.234 567 898 04

SWITCHES

FIXED DECIMAL POINT: the number is displayed as it is commonly written, with no exponent and the decimal point correctly placed. The last significant displayed digit is automatically rounded.

If the number to be displayed overflows to the left of the decimal point (i.e. the sum of the exponent and the DECIMAL DIGITS wheel setting exceeds 9) then the display for the overflowed register automatically reverts to floating decimal point.

When underflow occurs the display does not revert to floating decimal point.

DECIMAL DIGITS

Positions the decimal point on the display. The wheel setting designates the maximum number of digits which can appear to the right of the decimal point.

EXAMPLE:

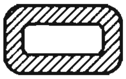
(See KEYING and SWITCHING INSTRUCTIONS on Page 18.)

SWITCH: FIXED POINT

STEP	KEY	STEP	KEY
1	CLEAR	7	•
2	1	8	6
3	2	9	7
4	3	10	8
5	4	11	9
6	5	12	8

DECIMAL WHEEL SETTING	DISPLAY
SET: DECIMAL DIGITS to 5	12345.67898 → x
SET: DECIMAL DIGITS to 3	__12345.679 → x Leading Zeros Blanked Last digit Rounded
SET: DECIMAL DIGITS to 6	1.234 567 898 04 → x Numbers too large for this setting - automatically reverts to floating point.

SWITCHES



ILLEGAL OPERATIONS LIGHT, to the left of the display, indicates that a mathematically illegal operation has been performed, either from the keyboard or from a program.

An illegal operation during a program will not stop the program; however, the light will remain set. Press any key on the keyboard to reset the light (see Page 17 for examples and apparent exceptions to this).

Illegal operations are:

Division by zero
$\sqrt{x}$ where $x < 0$
ln x where $x \leq 0$; log x where $x \leq 0$
arc sin x where $|x| > 1$; arc cos x where $|x| > 1$
arc cosh x where $x < 1$; arc tanh x where $|x| > 1$

ENTRY KEYS

0 THROUGH 9

The digit 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.

Used in conjunction with the following keys to provide access to the registers in the magnetic-core memory:

GO TO ( )( ), $x\rightarrow( )$, $y\rightarrow( )$, and $y\rightleftarrows( )$ (this use is explained in the pages describing the listed keys).

EXAMPLE:

enter 12345.06789

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

PRESS: CLEAR 1 2 3

PRESS: 4 5 . 0

PRESS: 6 7 8 9

DISPLAY: 12345.06789 → x

CLEAR x

CLEAR x Clears the X register only (0. → X).
Clears the ARC and HYPER conditions.
It is not necessary to press CLEAR X before each entry.

'CLEAR X' is required if the number in X has not been 'terminated' but is to be erased without affecting any other register. A number is terminated when an operation (e.g. ↑, +, etc.) has been performed after entry of the number; in this case the next number entered will automatically replace the terminated number and 'CLEAR X' is not required.

A number is not terminated if the last key used was a digit, decimal point, 'CHG SIGN' or 'ENTER EXP'; in this case, if 'CLEAR X' is not used, the next number entered will not replace the unterminated number but become a part of it.

NOTE

Many of the examples throughout this text include the 'CLEAR X' key only because the preceeding examples have left an unterminated number in the X register.

ENTRY KEYS

CLEAR x

CONTINUED

The example following serves three functions:

- illustrates the use of the 'CLEAR X' key;
- illustrates the difference between terminated and unterminated numbers;
- provides a valuable 'time-saving' hint.

A long list of numbers is to be added; as each number is added, the accumulative total appears in the Y register and the last number entered appears in the X register. Halfway through the list, the person operating the calculator is distracted; when he returns to the list he cannot remember whether or not the number in X has been added to the total in Y. The example shows how the 'CLEAR X' key can be used to determine this without repeating the complete list.

EXAMPLE:

Add -, - , 7, 32, 4, - -, etc.

operator interrupted here

(assume that the numbers before 7 total 456)

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 2.

STEP	KEY	
1	CLEAR	
2	4	
3	5	
4	6	
5	↑	456. → y (accumulative total)

this sets up the situation described; the next numbers to be added are 7 and 32:

STEP	KEY		
6	7	7. → x ,	
7	+	7. → x ,	463. → y
8	3		
9	2	32. → x ,	463. → y

the operator is interrupted and now he cannot recall whether or not the + key was pressed.
To determine this:

STEP	KEY		
10	1	321. → x ,	463. → y

ENTRY KEYS

1 has not replaced 32; obviously 32 was not 'terminated' which indicates that the + key was not pressed. To correct and continue the list:

STEP	KEY	
11	CLEAR x	0. → x, 463. → y
12	3	
13	2	32. → x, 463. → y
14	+	32. → x, 495. → y
15	4	
16	+	4. → x, 499. → y
etc.	--	

To illustrate the case where the + key has been pressed.

STEP	KEY	
1-through-9 of the previous example		32. → x, 463. → y
10	+	32. → x, 495. → y

to determine if the + key has been used:

STEP	KEY	
11	1	1. → x, 495. → y

1 has replaced 32; therefore 32 was 'terminated' which indicates that the + key must have been used. To correct and continue the list:

STEP	KEY	
12	CLEAR x	0. → x, 495. → y
13	4	
etc.	--	

CLEAR

'CLEAR' does <u>not</u> clear the entire contents of the memory.

CLEAR

Clears the display registers (0. → x, y, z).
Clears the e and f registers in the magnetic-core memory (this is explained in the pages dealing with the ACC+ and ACC– keys).
Clears the SET FLAG condition (explained under programming).
Clears the ARC and HYPER conditions.

ENTRY KEYS

CLEAR

CONTINUED

It is not always necessary to use the CLEAR key before performing a new calculation; except in the case of an unterminated entry (explained under CLEAR X key) and the special case explained under ACC+ and ACC–, new information automatically replaces the old.

CLEAR is, however, often a useful step, especially in a program, as old information in the x, y, z, *e*, and *f* registers could give an erroneous result to a calculation.

ENTER EXP

ENTER EXP Clears the exponent in the X register and causes the next digit entries (0 to 9) and CHG SIGN to affect only the exponent. The exponent is entered serially, each new digit entered becomes the least significant digit of the exponent.

Pressing the decimal point key after the ENTER EXP key has been pressed clears the ENTER EXP condition (see description of decimal point key).

Pressing ENTER EXP after any keyboard operation, except digit entry will enter 1 in the X register (thus, for example, it is not possible to recall a number from storage and change its exponent by means of the ENTER EXPONENT key).

EXAMPLES:

SWITCH: **FLOATING**

SET: DECIMAL DIGITS to 3

(a) Enter 1 x 10^0(1.0)

PRESS: CLEAR x ENTER EXP

DISPLAY: 1.000 000 000 00 → x

(note that it is not necessary to use the 1 or decimal point keys.)

repeat with **FIXED POINT**

DISPLAY: 1.000 → x

(b) Enter 1 x 10^3 (1,000)

SWITCH: **FLOATING**

PRESS: CLEAR x ENTER EXP 3

DISPLAY: 1.000 000 000 03 → x

(note that the 1 and decimal point keys are not used)

repeat with: **FIXED POINT**

DISPLAY: 1000.000 → x

(c) Enter 1.2678 x 10^2 (126.78)

SWITCH: **FLOATING**

PRESS: CLEAR x 1 2 6

PRESS: 7 8 ENTER EXP 2

alternatively,

PRESS: CLEAR x 1 2 6

PRESS: • 7 8

DISPLAY: 1.267 8 02 → x

Numbers smaller than 1 x 10^{-9} must be entered using the ENTER EXP key.

EXAMPLE:

Enter 2 x 10^{-10} (.000 000 000 2)

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 9

(a) Incorrect entry:

PRESS: CLEAR x • 0 0 0

PRESS: 0 0 0

CONTINUED

ENTRY KEYS

ENTER EXP

CONTINUED

PRESS: 0 0 0 2

DISPLAY: 2. → x

SWITCH: **FLOATING**

DISPLAY: 2. -00 → x

the required number has not been entered.

(b) Correct entry:

PRESS: CLEAR x 2

PRESS: ENTER EXP CHG SIGN 1 0

DISPLAY: 2. -10 → x

CHG SIGN

CHG SIGN **Changes the sign of the contents of the X register; the number in X may be terminated or unterminated (explained under CLEAR X key on Page 23).**

If ENTER EXP was pressed, changes the sign of the exponent only.

EXAMPLE:

Enter -4.9×10^{-3} (–.0049)

SWITCH: **FLOATING**

SET: DECIMAL DIGITS to 4

PRESS: CLEAR x CHG SIGN 4 9

PRESS: ENTER EXP CHG SIGN 3

DISPLAY: -4.9 -03 → x

repeat with: **FIXED POINT**

DISPLAY: -.0049 → x

ENTRY KEYS

• **Enters the decimal point.**
It is not necessary to use the decimal point when entering integers or when using the ENTER EXP key.
Clears the ENTER EXP condition; then a digit key or CHG SIGN will affect the number but not the exponent.

The ability of the decimal point key to override the ENTER EXP condition may be used to correct an error in entry without using CLEAR X.

EXAMPLE:

−4.5 x 10^{-2} is required
but 4. x 10^{-2} is entered in error.

SWITCH: **FLOATING**

PRESS: CLEAR x 4 ENTER EXP 2 CHG SIGN

DISPLAY: 4.   -02 → x

to correct this without using CLEAR X key:

PRESS: • 5 CHG SIGN

DISPLAY: -4.5   -02 → x

SWITCH: **FIXED POINT**

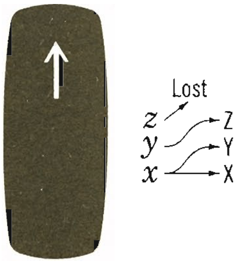
SET: DECIMAL DIGITS to 3

repeat the above example

DISPLAY: -.045 → x

CONTROL KEYS

The five control keys (↑, ↓, ROLL ↑, ROLL ↓, $x \rightleftarrows y$) are used to reposition the contents of the display registers.



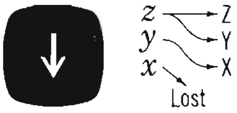
↑ Duplicates the contents of X in the Y register.
Shifts the contents of Y to the Z register.
The contents of the Z register are lost.

EXAMPLE:

SWITCH: **FIXED POINT**

PRESS: CLEAR x 4 ↑ 3 ↑

DISPLAY: 4. → z
3. → y
3. → x



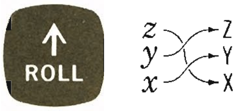
↓ Duplicates the contents of Z in the Y register.
Shifts the contents of Y to the X register.
The contents of the X register are lost.

EXAMPLE:

SWITCH: **FIXED POINT**

PRESS: 6 ↑ 5 ↑ ↓

DISPLAY: 6. → z
6. → y
5. → x



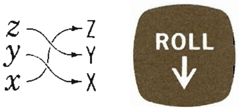
↑ ROLL 'Rolls' the display up without losing information:
shifts the contents of X to the Y register;
shifts the contents of Y to the Z register;
shifts the contents of Z to the X register.

Step-saving hint: ROLL ↓ has the same effect as ROLL ↑, ROLL ↑.

EXAMPLE:

See Roll ↓ key.

CONTROL KEYS



ROLL ↓ 'Rolls' the display down without losing information:
shifts the contents of Z to the Y register;
shifts the contents of Y to the X register;
shifts the contents of X to the Z register.

Step-saving hint: ROLL ↑ has the same effect as ROLL ↓, ROLL ↓.

EXAMPLE:

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 4

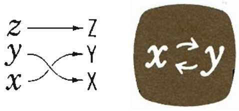
PRESS: CLEAR 9 ↑ROLL π

PRESS: ↑ROLL ↑ROLL

DISPLAY: 3.1416 → z
0. → y
9. → x

steps can be saved by using the ROLL ↓ key

PRESS: CLEAR π ROLL↓ 9



x⇄y Exchanges the contents of the X and Y registers.
The contents of Z remain unchanged.

EXAMPLE:

$$\frac{20}{3+2}=4$$

this example was used on Page 10 to illustrate the use of the Z register for temporary storage. Discounting the CLEAR instruction (which is not always necessary) nine steps were required. By using the *x⇄y* key only eight steps are required.

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 2

PRESS: 3 ↑ROLL 2 +

PRESS: 2 0 x⇄y ÷

DISPLAY: 4.00 → y

ARITHMETIC KEYS

The four arithmetic keys +, −, ×, ÷, are used to perform operations on the numbers in the X and Y registers. The result of the calculation appears in the Y register and the number in the X register remains unchanged. The number in the Z register is not affected.

A calculation using all four keys and illustrating some step-saving hints is included after the ÷ key has been explained.

+ 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

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

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

PRESS: 8 ↑ 4 +

DISPLAY: original contents of y → *z*
12. → *y*
4. → *x*

− 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.

EXAMPLES:

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

ARITHMETIC KEYS

(a) 17 − 9 = 8

PRESS: 1 7 ↑ 9 −

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

(b) 47 − 19 − 8 + 5 = 25

(contents of Z to remain unchanged)

PRESS: 4 7 $x \rightleftarrows y$

PRESS: 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:

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

(a) [(3 x 4) − 6] x 8 = 48

PRESS: 3 ↑ 4 ×

CONTINUED

ARITHMETIC KEYS

× CONTINUED

PRESS: 6 − 8 ×

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

(b) $5^2 = 25$

PRESS: 5 ↑ ×

DISPLAY: 25. → y

(c) $5^4 = 625$

PRESS: 5 ↑ × × ×

DISPLAY: 625. → y

(see also the second example under e^x key, Page 45).

÷

÷ **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:

$36 \div 9 = 4$

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

PRESS: 3 6 ↑ 9 ÷

DISPLAY: original contents of Y → z
4.00000 → y
9. → x

ARITHMETIC KEYS

The following example includes the use of all four arithmetic 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, add them and store the result;

(b) solve the denominator in the same way;

(c) recall the numerator and divide.

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 1

		DISPLAY REGISTERS			
STEP	KEY	X	Y	Z	NOTES
1	CLEAR	0	0	0	--
2	3	3	0	0	--
3	↑	3	3	0	--
4	4	4	3	0	--
5	×	4	12	0	(3 x 4) → 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) → y
11	↓	−1	12	12	recall (3 x 4) to Y
12	+	−1	11	12	(3 x 4) + (8 – 9) → 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 x 2) → y
18	6	6	16	11	--
19	−	6	10	11	(8 x 2) – 6 → y
20	ROLL ↓	10	11	6	recall (3 x 4) + (8 – 9) to Y
21	÷	10	1.1	6	$\frac{(3 \times 4) + (8 - 9)}{(8 \times 2) - 6} = 1.1 \rightarrow y$

ARITHMETIC KEYS

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

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 ↑.

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).

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

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

ARITHMETIC KEYS

The following example illustrates an accumulative process using the arithmetic keys; the ↑ and ↓ keys allow temporary storage and recall of the accumulative total.

The sum of the products, $n_1n_2 + n_3n_4 + n_5n_6 + \cdots$ 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.

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 0

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	×	$(n_5\ n_6) \rightarrow y$
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.

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

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

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

More complex expressions, such as $\frac{n_1 + n_2}{n_3} + \frac{n_4 + n_5}{n_6} +$ etc. can also be solved using this general program form; however, in these cases additional steps are required.

STORAGE AND RECALL KEYS

The Storage and Recall keys [*x*→( ), *y*→( ), *y*⇄( )] are used as prefixes to the numeric keys (0 through 9) and the alphabetic keys (*a* through *f*) to provide access to the 16 storage registers (see Figure 1, Page 8).

It is not necessary to clear a storage register before storing a number as the new number automatically replaces the old number.

RCL, ACC+, and ACC– are special storage and recall keys; these are included under VECTOR KEYS (starting on Page 49).

x→( )

x→( ) Stores the contents of the X register into the storage register selected by the next key pressed:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e* or *f*.

The contents of the X register remain unchanged.

EXAMPLE:

Store π in the *c* register.

PRESS: π *x*→( ) *c*

y→( )

y→( ) Stores the contents of the Y register into the storage register selected by the next key pressed:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e* or *f*.

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 manner as the *x*→( ) key stores the contents of X.

y⇄( )

y⇄( ) Used for both storage and recall. Exchanges the contents of the Y register with the contents of the storage register selected by the next key pressed:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e* or *f*.

This is the only key that can be used to recall the contents of the numeric storage registers (0 through 9).

STORAGE AND RECALL KEYS

EXAMPLE:

Enter π into the Y register and exchange with the contents of the 6 - register.

PRESS: π ↑

PRESS: y⇄( ) 6

DISPLAY: original contents of 6 → y

(if the display in y appears nonsensical this is because the 6 - register originally contained program steps.)

To recall π from the 6 - register:

PRESS: y⇄( ) 6

DISPLAY: π → y

the 6 - register now contains its original contents.

The y⇄( ) key is particularly useful for moving data from one memory register to another (this technique is used in the 'possible combinations' program, starting on Page 97).

EXAMPLE:

PRESS: y⇄( ) a

PRESS: y⇄( ) b

PRESS: y⇄( ) c

the contents of y → a
the contents of a → b
the contents of b → c
the contents of c → y

STORAGE AND RECALL KEYS

a

THROUGH

f

The alphabetic keys, a through f, are used for both storage and recall.

STORAGE: See $x\to(\ )$, $y\to(\ )$, $y\rightleftarrows(\ )$ keys.

RECALL: Press a, b, c, d, e or f to recall the contents of the selected register to the X register; the contents of the recalled register remain unchanged.

The alphabetic keys are also used, in conjunction with the GO TO ( ) ( ) key, to address the program counter (as described in the programming section of this manual).

The e and f registers are a special case; they may be used as either

1) storage register (as described above) or
2) accumulator registers (this use is described under the RCL, ACC+, and ACC– keys, Page 52).

The CLEAR instruction clears both the e and f registers but does not clear any other storage register.

0

THROUGH

9

The numeric keys, 0 through 9, when used for storage or recall must be used after one of the following keys:

$x\to(\ )$, $y\to(\ )$, $y\rightleftarrows(\ )$.

[See also GO TO ( ) ( ) key in the programming section of this manual.]

Unlike the alphabetic keys, if a numeric key only is pressed, then the number in the storage register will not be recalled; instead, the digit pressed will appear in the X register.

The following is a general example illustrating use of the memory.

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 constant will first be stored in the calculator memory; this makes it unnecessary to re-enter k each time it is required.

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

STORAGE AND RECALL KEYS

PRESS: CLEAR x (only if necessary)

Enter k and store k in the *a* register (an alphabetic register is used in preference to a numerical, as the latter requires two keystrokes for recall):

PRESS: 1 • 6 8 4

PRESS: $x \rightarrow ( )$ *a*

Enter n_1, recall k and multiply:

PRESS: 3 ↑ *a* ×

DISPLAY: (kn_1) 5.052 → y

Enter n_2, recall k and multiply:

PRESS: 1 1 • 2 ↑

PRESS: *a* ×

DISPLAY: (kn_2) 18.8608 → y

Enter n_3 · · · · · etc.

FUNCTION KEYS

DEGREES RADIANS

This switch selects the units, degrees or radians, to be used and displayed in calculations involving the trigonometric functions: angles must be entered in the units selected.

The trigonometric functions of angles from 0° up to 360° (or greater) can be calculated; however, the inverse trigonometric functions are calculated only for the principle values of the functions:

$$\begin{aligned} \theta &= \sin^{-1} x; & -90^\circ &\leq \theta \leq +90^\circ; & (-\pi/2 &\leq \theta + \pi/2) \\ \theta &= \cos^{-1} x; & 0^\circ &\leq \theta \leq 180^\circ; & (0 &\leq \theta + \pi\ ) \\ \theta &= \tan^{-1} x; & -90^\circ &\leq \theta \leq +90^\circ; & (-\pi/2 &\leq \theta + \pi/2) \end{aligned}$$

As an example:

cos 150° = cos 210° = cos 510° = (etc.) = −.866; but, arc cos −.866 = 150°

To convert DEGREES (RADIANS) to RADIANS (DEGREES)

1. SWITCH: DEGREES (RADIANS)
2. ENTER: DEGREES (RADIANS)
3. PRESS: (one of) sin x, cos x, tan x
4. SWITCH: RADIANS (DEGREES)
5. PRESS: arc, (as in step 3) sin x, cos x, tan x;

the result in RADIANS (DEGREES) appears in the X register; however, the principle value of the function must be taken into account if this method is used.

If the following method is used the principle value of the function need not be considered:

To convert DEGREES (RADIANS) to RADIANS (DEGREES):

multiply degrees by $\frac{\pi}{180^\circ}$ to convert to radians;

multiply radians by $\frac{180^\circ}{\pi}$ to convert to degrees.

sin x

sin x Replaces the contents of the X register with the Sine of the contents of X (see also ARC and HYPER keys).

cos x

cos x Replaces the contents of the X register with the Cosine of the contents of X (see also ARC and HYPER keys).

tan x

tan x Replaces the contents of the X register with the Tangent of the contents of X (see also ARC and HYPER keys).

FUNCTION KEYS

arc ▼ Used as a prefix to the hyperbolic key and trigonometric keys to calculate the inverse functions. The answer appears in the X register.

For example:

arc ▼ sin x gives $\sin^{-1} x$,

arc ▼ hyper ▼ sin x gives $\sinh^{-1} x$.

hyper ▼ Used as a prefix to the trigonometric keys to calculate the hyperbolic functions; ARC must preceed HYPER to calculate the inverse hyperbolic functions. The answer appears in the X register. For example:

hyper ▼ cos x gives cosh x

EXAMPLE:

$\sinh^{-1} 1 = .88137$

SWITCH: **DEGREES**

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

(correct sequence)

PRESS: 1 arc ▼ hyper ▼ sin x

DISPLAY: ($\sinh^{-1} 1$) .88137 → x

(incorrect sequence)

PRESS: 1 hyper ▼ arc ▼ sin x

DISPLAY: ($\sin^{-1} 1$) 90.00000 → x

note that when ARC is pressed after HYPER, the HYPER condition is cleared.

FUNCTION KEYS

$\sqrt{x}$

$\sqrt{x}$ Replaces the contents of the X register with the square root of the contents of X.

EXAMPLE:

$\sqrt{9} = 3$

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 2

PRESS: 9 $\sqrt{x}$

DISPLAY: 3.00 → x

NOTE

To solve for $\sqrt{x}$ when $x = a^2 + b^2$ see EXAMPLE c) on Page 51.

$\ln x$

$\ln x$ Replaces the contents of the X register with the natural logarithm (logarithm to the base e) of the contents of X.

See e^x for examples.

e^x

e^x Replaces the contents of the X register with e raised to the power defined by the contents of X (i.e. the anti-logarithm, to the base e, of the contents of X).

EXAMPLE:

enter e into the X register.

SWITCH: **FLOATING**

PRESS: 1 e^x

DISPLAY: 2.718 281 828 00 → x

(NOTE: the guard digits contain - - 48)

FUNCTION KEYS

EXAMPLE:

$19^{1.6} = 1.111\ 746\ 475 \times 10^{2}$

SWITCH: **FLOATING**

PRESS: 1 9 ln x ↑

PRESS: 1 · 6 ×

PRESS: ↓ e^x

DISPLAY: 1.111 746 475 02 → x

EXAMPLE:

$\sqrt[5]{23} = 1.872\ 171\ 230$

SWITCH: **FLOATING**

PRESS: 2 3 ln x ↑

PRESS: 5 ÷

PRESS: ↓ e^x

DISPLAY: 1.872 171 230 00 → x

FUNCTION KEYS

log x

log x Replaces the contents of the X register with the logarithm, to the base 10, of the contents of X.

To calculate an antilogarithm to the base 10, use the following formula:

$x = 10^y$ (where $y = \log_{10} x$)

10^y can be calculated using the natural logarithm,

$\ln 10^y = y \cdot \ln 10$

taking the antilog ($\ln^{-1} x = e^x$),

$x = e^{(y \cdot \ln 10)}$

EXAMPLE:

take the antilog $_{10}$ of .60206
($\log_{10}^{-1} .60206 = 4$)

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

take the natural log of 10:

PRESS: 1 0 ln x ↑

DISPLAY: ln10 → y

multiply ln 10 by y (y = .60206):

PRESS: • 6 0 2

PRESS: 0 6 ×

DISPLAY: $y \cdot \ln 10$ → y

take $\ln^{-1} (y \cdot \ln 10)$

PRESS: ↓ e^x

DISPLAY: ($\log_{10}^{-1} .60206 =$) 4.00000 → x

FUNCTION KEYS

int x

int x Eliminates the decimal part of the contents of the X register without affecting the sign or the integer part.

EXAMPLE:

integer −5.9 = −5

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 4

PRESS: CHG SIGN 5 • 9

DISPLAY: -5.9 → x

PRESS: int x

DISPLAY: -5. → x

SET: DECIMAL DIGITS to 0

Repeat the above procedure; notice that when −5.9 is entered the number is rounded, the display appearing as −6. As the calculation is performed in floating point (even though the display is fixed point), when 'int x' is pressed, the result (−5.) is correct.

EXAMPLE:

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

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

enter 5.72

PRESS: 5 • 7 2 ↑

separate the principle part (degrees) from the decimal part (minutes):

PRESS: int x —

CONTINUED

FUNCTION KEYS

int x

CONTINUED

store degrees in the Z register:

PRESS: ROLL ↓

convert the decimal part to minutes by multiplying by 60:

PRESS: $x \rightleftarrows y$ 6 0 ×

DISPLAY: (°) 5. → z

(') 43.2 → y

|y|

|y| **Absolute value of y. Sets the contents of the Y register positive without affecting the sign of the exponent.**

EXAMPLE:

The absolute value of $-5 \times 10^{-2} = 5 \times 10^{-2}$

SWITCH: **FLOATING**

PRESS: CLEAR x CHG SIGN 5

PRESS: ENTER EXP CHG SIGN 2 ↑

DISPLAY: -5. -02 → y

PRESS: |y|

DISPLAY: 5. -02 → y

This instruction is used (mostly in programming) to check if a number falls within a given (+ or −) tolerance. The diagnostic program, contained in the Miscellaneous section of your Program Library, contains several examples of this.

The program on Page 96 of this manual also includes the |y| instruction.

π

π **Clears the X register and enters pi.**
Up to and including ten digits are displayed, but two more significant digits (- -60) are included in the guard digits (see Page 12).

VECTOR KEYS

The vector keys (TO POLAR, TO RECT, ACC+, ACC–, RCL) provide complete capability for performing complex or vector arithmetic.

NOTE

Although ACC+, ACC– and RCL are included with VECTOR KEYS, there is no valid reason for reserving these instructions exclusively for use with 'TO RECT' and 'TO POLAR'. Several of the example programs (in the Programming section of this manual) include one or more of these keys even though 'TO RECT' and 'TO POLAR' are not used.

Conversion from rectangular to polar coordinates will calculate the angle, θ, in the range:

$-180° < \theta < +180°$

$-\pi$ radians $< \theta \leq +\pi$ radians

TO POLAR

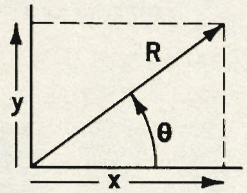
TO POLAR Converts rectangular coordinates, consisting of an x component in the X register and a y component in the Y register, to polar coordinates:

Angle (θ) = $\text{Tan}^{-1} y/x \rightarrow y$

Radius (R) = $\sqrt{x^2 + y^2} \rightarrow x$

EXAMPLES:

a) x = 4, y = 3, convert to polar form.
(R = 5, θ = 36.87°)



SWITCH: DEGREES

SWITCH: FIXED POINT

SET: DECIMAL DIGITS to 3

CONTINUED

TO POLAR

CONTINUED

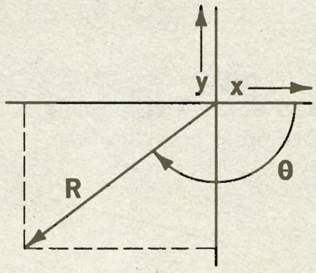
VECTOR KEYS

PRESS: 3 ↑ 4 TO POLAR

DISPLAY: (θ in degrees) 36.870 → *y*
(R) 5.000 → *x*

SWITCH: RADIANS
PRESS: repeat the preceeding example.
DISPLAY: (θ in radians) .644 → *y*
(R) 5.000 → *x*

b) x = –4, y = –3, convert to polar form.
(R = 5, θ = –143.13°)



SWITCH: DEGREES

SWITCH: FIXED POINT
SET: DECIMAL DIGITS to 3

PRESS: 3 CHG SIGN ↑ 4 CHG SIGN

PRESS: TO POLAR

DISPLAY: (θ in degrees) -143.130 → *y*
(R) 5.000 → *x*

VECTOR KEYS

c) $\sqrt{(3.4)^2 + (8.5)^2} = 9.155$

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 3

(DEGREES - RADIANS switch can be in either position)

PRESS: 3 • 4 ↑

PRESS: 8 • 5

PRESS: TO POLAR

DISPLAY: $[\sqrt{(3.4)^2 + (8.5)^2} =]$ 9.155 → x

an angle (in either degrees or radians) appears in the Y register, this may be ignored.

TO RECT

Converts polar coordinates, consisting of radius (R) in the X register at an angle (θ) in the Y register, to rectangular coordinates:

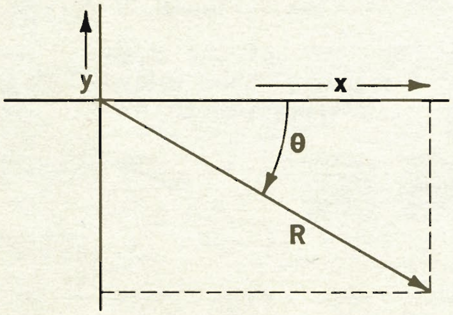
TO RECT

y component = R Sin θ → y

x component = R Cos θ → x

EXAMPLE:

R = 8, θ = –30° (330°), convert to rectangular form.



CONTINUED

VECTOR KEYS

TO RECT

CONTINUED

SWITCH: DEGREES

SWITCH: FIXED POINT

SET: DECIMAL DIGITS to 3

PRESS: 3 0 CHG SIGN ↑ 8

PRESS: TO RECT

alternatively

PRESS: 3 3 0 ↑ 8

PRESS: TO RECT

DISPLAY: (y) -4.000 → y

(x) 6.928 → x

ACC +

ACC −

RCL

The ACC+, ACC− and RCL keys are special storage and recall keys for use with the accumulator registers, e and f. These keys provide capability of vector addition and subtraction with single keystrokes. (See also the NOTE on Page 49).

The CLEAR key clears both the e and f registers. To clear the e and f registers without using CLEAR (which also clears the display registers) press either

RCL ACC −

or

CLEAR x x→( ) e x→( ) f

VECTOR KEYS

ACCUMULATE +: adds the contents of the X register to the original contents of f and, simultaneously, adds the contents of the Y register to the original contents of e. The sums are entered into the f and e registers respectively; the contents of X and Y remain unchanged.

$$e + y \rightarrow e$$

$$f + x \rightarrow f$$

ACCUMULATE −: subtracts the contents of the X register from the original contents of f and, simultaneously, subtracts the contents of the Y register from the original contents of e . The differences are entered into the f and e registers respectively; the contents of X and Y remain unchanged.

$$e - y \rightarrow e$$

$$f - x \rightarrow f$$

RCL: recalls the contents of f to the X register and the contents of e to the Y register. The contents of f and e remain unchanged.

$$e \rightarrow y$$

$$f \rightarrow x$$

EXAMPLES:

(a) Vector addition

$$(2x + 3y) + (4x + 5y) - (3x - 6y) = 3x + 14y$$

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 3

STEP	KEY	NOTES
1	CLEAR	must be used to clear e and f
2	3	Y is entered before X
3	↑	--
4	2	--
5	ACC +	$0 + 3 = 3 \rightarrow e$ $0 + 2 = 2 \rightarrow f$
6	5	--
7	↑	--
8	4	--
9	ACC +	$3 + 5 = 8 \rightarrow e$ $2 + 4 = 6 \rightarrow f$

CONTINUED

VECTOR KEYS

ACC +

ACC −

RCL

CONTINUED

STEP	KEY	NOTES
10	6	--
11	CHG SIGN	--
12	↑	--
13	3	--
14	ACC −	8 − (−6) = 14 → *e* 6 − 3 = 3 → *f*
15	RCL	*e* → *y* *f* → *x*

DISPLAY: (y) 14. → *y*

(x) 3. → *x*

NOTE

In the following example notice in particular the powerful combination of vector keys and logarithmic keys (ln x and e^x) which allow numbers to be added and numbers to be multiplied simultaneously.

(b) Multiplication of complex numbers

$(j = i = \sqrt{-1})$

$(3 + j4)\,(-2 + j3) = -18 + j1$

this problem is best solved in the calculator by the following method as this method can be easily adapted to solve problems with many complex terms:

1. convert the quantities in parentheses to polar form (R, $\underline{/\theta}$),
2. multiply the R quantities and add the angles,
3. convert the result back to rectangular form.

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to **3**

NOTE

The setting of the DEGREES-RADIANS switch will not affect the final result of this calculation (provided that the switch is not changed during the calculation). However, each time TO POLAR is used, the displayed number in the Y register will be in either degrees or radians, depending on the position of the switch.

VECTOR KEYS

clear the *e* and *f* registers:

PRESS: CLEAR

enter $(3 + j4)$ and convert to polar form (R_1 at $\angle\theta_1$):

PRESS: 4 ↑ 3 TO POLAR

take the log of R_1 and store $\ln R_1$ in *f* and $\angle\theta_1$ in *e*:

PRESS: ln x ACC +

enter $(-2 + j3)$ and convert to polar form (R_2 at $\angle\theta_2$):

PRESS: 3 ↑ 2 CHG SIGN TO POLAR

take the log of R_2 and add $\ln R_2$ and $\angle\theta_2$ to *f* and *e* respectively:

PRESS: ln x ACC +

take the antilog of $(\ln R_1 + \ln R_2)$

PRESS: RCL e^x

CONTINUED

VECTOR KEYS

ACC +

ACC −

RCL

CONTINUED

convert the result (R in X, $\underline{/\theta}$ in Y) back to rectangular form:

PRESS: TO RECT

DISPLAY: (j) 1.000 → y

-18.000 → x

NOTE

The preceeding example can be changed to division of complex numbers, i.e. $\frac{3 + j4}{-2 + j3}$, by changing the ACC+ key, at its second appearance only, to ACC−.

INTRODUCTION TO PROGRAMMING

This section describes the programming keys and explains how to program the 9100A Calculator. The magnetic-core memory, described briefly on Page 8 of this manual, is covered in more detail to assist programmers in writing more efficient programs.

Programming the 9100A is simple, the keyboard operations are the program instructions and no special language has to be learned; one key pressed is one program step. It is, nevertheless, essential that the programmer knows exactly what each key (including those described in the previous section) does, and that he is precise in giving instructions to the calculator. However, the design of the calculator is such, that, if an error is made in a program, it is a simple matter to correct the program while it is in the machine.

REGISTERS

Figure 2, below, is a three-dimensional representation of the programmable part of the magnetic-core memory.

As previously described, there are sixteen registers:

0 through *d* - these fourteen registers are used for either program steps or data storage;
e, *f* - used for data storage only.

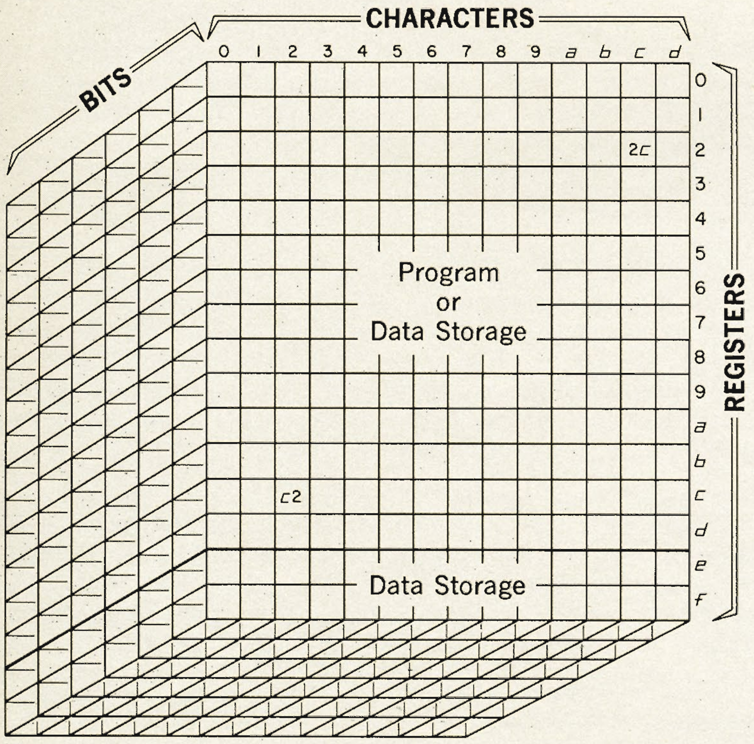


Figure 2. Programmable Memory

INTRODUCTION TO PROGRAMMING

CHARACTERS

The program registers each contain fourteen locations called 'characters'; each character can contain either one program step (one keystroke), which gives a maximum capacity of 196 program steps, or one digit of a data number.

Each piece of data consists of a twelve-digit number (ten displayed digits and two guard digits) and a two-digit exponent so that, if data is stored in the program registers, a complete register is required for each number. The maximum program size of 196 steps is reduced by 14 steps for each program register used for data storage.

Data and program steps cannot both be stored in any one register at the same time (but, see 'ENTERING A CONSTANT' on Page 104).

MEMORY ADDRESS

Each 'character' has a unique, two-part address, the register number followed by the character number; thus, for example, the third character in register c is addressed as 'c2' (Figure 2 Page 57) and '2c' is the address of character c in register 2.

BITS

Each character has a depth of six locations; the locations are known as 'bits'. It is these six 'bits' which contain, in a coded form, either one program step or, alternatively, one digit of a data number.

OCTAL CODE

The code for each key is shown in the appendix at the back of this manual and also on the pull-out card located at the lower front of the calculator. The code is in 'octal' form (based on 8 instead of the normal 10 so that there are no 8's or 9's); each key of the calculator is represented by a two-digit number and can, therefore, be used in a program. STEP PRGM (octal 51) is not usable as a program step.

Any one of these two-digit instruction codes can be stored in each character (this is not a contradiction of the earlier statement, that only one data digit can be stored per character, as each data digit requires a two-digit code number). All six bits of one character are required to store one two-digit instruction code.

It is not necessary to use the octal code in order to program the calculator until it becomes necessary to edit or correct a program (see Page 79). It is, therefore, more convenient to consider the programmable memory as consisting of only two dimensions, a 14 x 14 array of registers and characters.

INTRODUCTION TO PROGRAMMING

PROGRAM ADDRESSING

When a program is written it is first assigned a starting address [because of the convenience of the END instruction, described on Page 70, it is usual (although not essential) to start at address 00]. As each step is entered into the calculator, by pressing the appropriate keys, the program is automatically stored sequentially, beginning at the starting address. With the first program step at 00 (for example) storage is used in the following sequence:

00, 01 - - - - - 09, 0a , 0b , 0c , 0d , 10, - - - - - 19,
1a , 1b , 1c , 1d , 20 - - - - up to - - - - - dc, dd.

When address dd is filled the 'program counter' automatically resets to address 00, so that, if another key is then pressed, the original program step in 00 will be erased and replaced by the new program step.

When a program is running, the same sequential steps are used except where the program includes branching instructions (see 'BRANCHING', below).

PROGRAM COUNTER

The 'Program Counter' is best considered as those electronic circuits which automatically step the calculator sequentially through the memory. The programmer can preset the counter to start at any address; this operation is known as 'addressing the program counter' (see GO TO key on Page 69).

BRANCHING

The programmer can also instruct the counter to 'branch' during a program (i.e. go to another address and continue stepping from there). Branching can take two forms; 'conditional' and 'unconditional'.

If the branch is 'conditional', the calculator makes the decision whether to branch or not; as a simplified example, the calculator may ask, "Is one number larger than another number?" If the answer is NO, the calculator branches to another address and performs the steps from there; if the answer is YES, the calculator goes to the next step in the program. Conditional branching is fully described, starting on Page 87.

An 'unconditional' branch gives the calculator no option; it must branch to the address indicated in the program (see GO TO key on Page 69).

INTRODUCTION TO PROGRAMMING

BLANKED AND DISPLAY MODES

The display is blanked while a program is being run; when the program stops, the calculator returns to the 'display mode'.

Many programs are short enough that the display will do no more than blink before reappearing; a long program, such as the diagnostic program, may leave the screen blanked for several seconds or even longer. It is possible to write programs which put the calculator into a 'loop' (it continually branches in a circle and will never return to the display mode unless instructed to do so); when this occurs the keyboard is 'locked out' and no key will work until the program is stopped, by pressing the STOP key (also see the PAUSE key on Page 70).

An error written into a program may result in an incorrect answer or it may put the calculator into a loop so that it remains blanked. When this occurs press the STOP key and the display will return.

PROGRAM WRITING

WHAT IS A PROGRAM?

A program is nothing more than a sequence of instructions telling the calculator what it must do in order to solve a particular calculation. In the previous sections 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 view a particular result; (in most cases) if the sequence was not followed exactly, the result was not correct. In a similar manner, the calculator must now be given a sequence of correct instructions.

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 Y and C in X; choose simple numbers, say A = 6, B = 5 and C = 3.

The display should be:

(A) 6. → z
(B) 5. → y
(C) 3. → x

Now press whatever keys are necessary to solve $\frac{A \times B}{C} \left[\frac{6 \times 5}{3}\right]$. Write down each operation as it is performed; also write down the effect on each of the three display registers (the space below is included for this purpose; write the first operation in step 00 and use 'END' as the last step). Remember that A, B and C are entered manually before the program is run; do not include entering A, B and C as program steps.

STEP	KEY	DISPLAY		
		X	Y	Z
Enter the No's.		C(=3)	B(=5)	A(=6)
00				
01				
02				
03				
04				
05				
06				
07				
08				
etc.				

PROGRAM WRITING

WHAT IS A PROGRAM?

CONTINUED

You have now written a usable program. If you wish to enter your program into the calculator here is the procedure.

SWITCH: **RUN**

PRESS: END SETS PROGRAM TO 00

SWITCH: **PROGRAM**

PRESS: the keys in the program sequence. NO STEP #

SWITCH: **RUN**

PRESS: END

to run (and rerun) the program:

enter A, B and C into Z, Y and X respectively.

PRESS: CONTINUE

Here are two (of several possible) programs to solve $\frac{A \times B}{C}$

(1)

STEP	KEY	DISPLAY X	Y	Z
Enter the numbers		C	B	A
00	ROLL ↓	B	A	C
01	×	B	A x B	C
02	↓	A x B	C	C
03	$x \rightleftarrows y$	C	A x B	C
04	÷	C	$\frac{A \times B}{C}$	C
05	END	C	final display $\frac{A \times B}{C}$	C

(2)

STEP	KEY	DISPLAY X	Y	Z
Enter the numbers		C	B	A
00	÷	C	B/C	A
01	↓	B/C	A	A
02	×	B/C	$\frac{A \times B}{C}$	A
03	END	B/C	final display $\frac{A \times B}{C}$	A

PROGRAM WRITING

WRITING A PROGRAM

Program writing may be divided, generally, into three main steps:

a) define the problem;
b) decide how the problem is to be solved;
c) write the steps sequentially for the calculator.

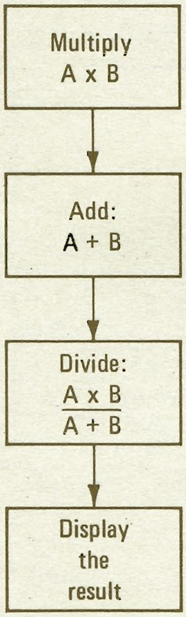
Example of program writing:

a) 'define the problem':

Write a program to solve $\frac{A \times B}{A + B}$ for any value 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':

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

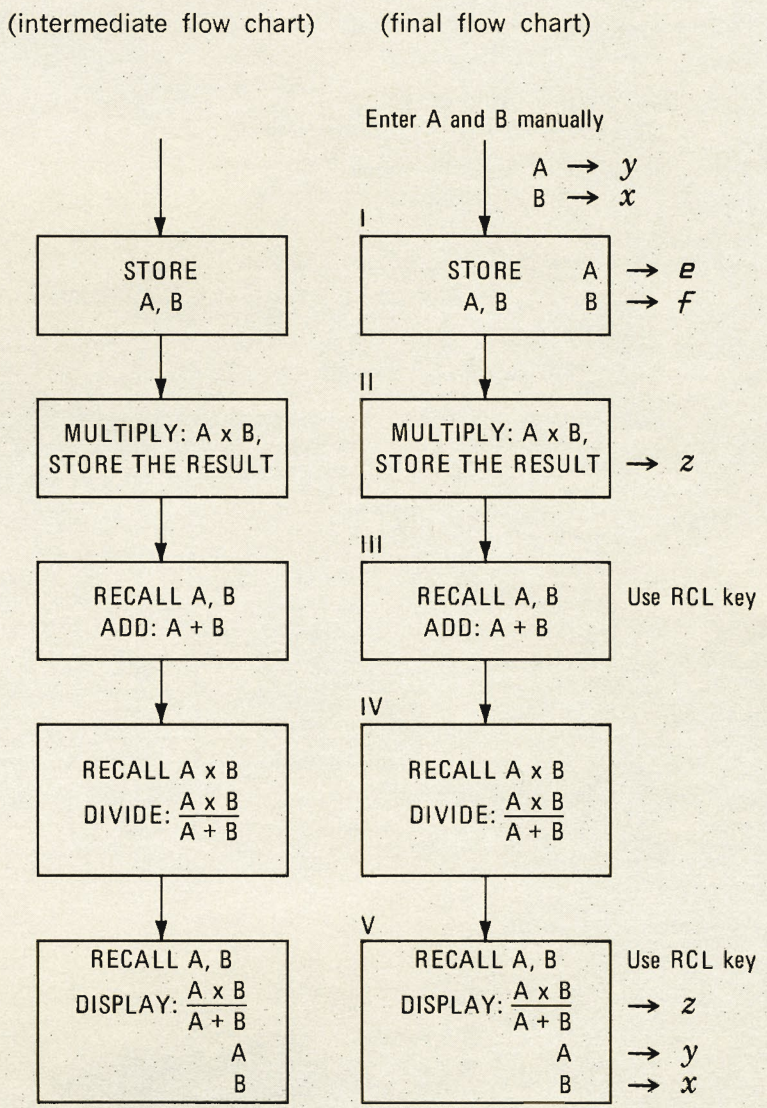


Next, draw the 'flow chart' in greater detail and then add specific notes, such as where numbers are to be stored, etc. It may be necessary if the problem is complex, to draw several versions of the flow chart.

PROGRAM WRITING

WRITING A PROGRAM

CONTINUED


(intermediate flow chart)
(final flow chart)
Enter A and B manually
A → y
B → x
STORE A, B
I
STORE A, B
A → e
B → f
MULTIPLY: A x B, STORE THE RESULT
II
MULTIPLY: A x B, STORE THE RESULT
→ z
RECALL A, B ADD: A + B
III
RECALL A, B ADD: A + B
Use RCL key
RECALL A x B DIVIDE: (A x B)/(A + B)
IV
RECALL A x B DIVIDE: (A x B)/(A + B)
RECALL A, B DISPLAY: (A x B)/(A + B) A B
V
RECALL A, B DISPLAY: (A x B)/(A + B) A B
Use RCL key
→ z
→ y
→ x

c) 'write the steps sequentially for the calculator':

Write the steps exactly as the keys would be pressed if the operations were to be performed manually; at each step write down the effect of the operation on the display and storage registers (the program pad, supplied with each instrument, may be used for this). Do not include any manual operations, such as digit entry, as part of the program.

PROGRAM WRITING

(Starting with I in the final flow chart)
A and B are stored in the memory:

STEP (address)	KEY	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f	STORAGE e
00	$x \rightarrow (\ )$	B	A	-	-	-
01	f	B	A	-	B	-
02	$y \rightarrow (\ )$	B	A	-	B	-
03	e	B	A	-	B	A

(II of the flow chart) multiply A and B and store the result in the Z register:

STEP (address)	KEY	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f	STORAGE e
04	×	B	A x B	-	B	A
05	↑	B	B	A x B	B	A

(III of the flowchart) recall A and B and add them:

STEP (address)	KEY	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f	STORAGE e
06	RCL	B	A	A x B	B	A
07	+	B	A + B	A x B	B	A

(IV of the flow chart) recall A x B and divide:

STEP (address)	KEY	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f	STORAGE e
08	↓	A + B	A x B	A x B	B	A
09	÷	A + B	$\frac{A \times B}{A + B}$	A x B	B	A

CONTINUED

WRITING A PROGRAM

CONTINUED

(V of the flow chart) display the result and A and B:

STEP (address)	KEY	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f	STORAGE e
0a	↑	A + B	A + B	$\frac{A \times B}{A + B}$	B	A
0b	RCL	B	A	$\frac{A \times B}{A + B}$	B	A
*0c	END	B	A	$\frac{A \times B}{A + B}$	B	A

***NOTE**

The END or STOP must be given, otherwise the program counter would continue to the next address and perform whatever step happened to be in the memory. END is preferable to STOP as END automatically resets the program counter to 00 ready for the next value of A and B to be entered; if the program is to be recorded on a magnetic card the END must be used (see END key on Page 70).

Here is the complete program; this will be used later in this section to illustrate program operation (Page 72). The key codes have been added.

STEP (address)	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f	STORAGE e
00	$x\rightarrow( )$	23	B	A	-	-	-
01	f	15	B	A	-	B	-
02	$y\rightarrow( )$	40	B	A	-	B	-
03	e	12	B	A	-	B	A
04	×	36	B	A x B	-	B	A
05	↑	27	B	B	A x B	B	A
06	RCL	61	B	A	A x B	B	A
07	+	33	B	A + B	A x B	B	A
08	↓	25	A + B	A x B	A x B	B	A
09	÷	35	A + B	$\frac{A \times B}{A + B}$	A x B	B	A
0a	↑	27	A + B	A + B	$\frac{A \times B}{A + B}$	B	A
0b	RCL	61	B	A	$\frac{A \times B}{A + B}$	B	A
				final display			
0c	END	46	B	A	$\frac{A \times B}{A + B}$	B	A

PROGRAM WRITING

When a program is completed it is sometimes helpful to step through the program manually to check that the display registers do, indeed, contain the numbers in the program.

EXAMPLE:

manually execute the above program.

SWITCH: **FIXED POINT**

SWITCH: **RUN**

SET: DECIMAL DIGITS to 5

PRESS: any digit for A

PRESS: ↑

PRESS: any digit for B

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

Step-saving has been stressed (and will continue to be stressed) throughout this manual; however, it is sometimes advisable to waste steps to make the program easier to operate. This is especially true when the program is written for an unskilled operator to use; in this case it is desirable to minimize the number of keys which the operator must press to run the program. It is also desirable to simplify interpretation of the final display. The program on Page 66 illustrates this.

1) The program could have been started as:

STEP		KEY	DISPLAY			STORAGE	
(address)	KEY	CODE	X	Y	Z	f	e
00	CLEAR	20	0	0	0	0	0
01	STOP	41	0	0	0	0	0
Manually enter A and B		---	B	A	0	0	0
02	ACC +	60	B	A	0	B	A

the result in the display and storage registers at step 02 is the same as in step 03 of the original program; one step has been saved. However, by using this alternative form it is necessary to press the CONTINUE key twice to run the program; the first time to start the program and the second time after the data is

CONTINUED

PROGRAM WRITING

WRITING A PROGRAM

CONTINUED

entered. With the original program CONTINUE has to be pressed only once, after data entry (see Page 69 for use of CONTINUE).

2) The calculation is complete at step 09; steps 0*a* and 0*b* are included only to allow the operator to verify that the correct values for A and B were used.

3) If steps 0*a* and 0*b* are not included, then it becomes unnecessary to store B in the memory; steps 00 and 01 can be deleted and step 06 changed to '*e*' (step 05 moves B to the Y register, then step 06 will now recall A to the X register).

PROGRAM KEYS

Selects the mode of operation (see also 'Program Operation', Page 72 and following).

PROGRAM MODE: used when entering a program from the keyboard and when editing a program.

RUN MODE: used when performing calculations manually, recording a program on a magnetic card, entering a program from a magnetic card, running a program and addressing the program counter.

GO TO ( )( ) Used to (manually) address the program counter and (in a program) as an unconditional branch to the address designated by the two instructions (which must be alphameric) immediately following the GO TO instruction.

If the GO TO and two alphameric instructions (the address) are given from the keyboard, then the program counter will reset to the address designated and wait for the next key to be pressed.

If the GO TO and two alphameric instructions are included in a program, the counter will branch unconditionally (see Page 59) to the address designated, perform the instruction located there and then continue to run the program sequentially from that address.

Three program steps are required to address the memory:

CORRECT

STEP	KEY
29	GO TO ( )( )
2a	5
2b	c

INCORRECT

STEP	KEY
29	GO TO ( )( )
2a	5c
2b	

During the display mode, if the GO TO key is pressed the screen will blank and remain blanked until two more keys (any keys) have been pressed.

CONTINUE Used to start program execution at the present address. Usually a manual operation but may be used as a 'no operation' program step (see Correcting a Program, Page 82).

If the program has halted at a STOP or PAUSE instruction, press CONTINUE to continue running the program (from the next program step).

If the program has halted at an END instruction, press CONTINUE to run the program (starting at address 00).

PROGRAM KEYS

STOP

STOP Halts the program and causes a return to the display mode (see Page 60). Use the STOP instruction in a program where data has to be entered manually.

When the program is halted, any keyboard operation can be manually executed without affecting the stored program. If, however, the keying sequence used stores data in a register which contains program steps, the program steps in that register will be destroyed.

END

END Halts program execution, causing a return to the display mode; also, resets the program counter to 00 so that pressing CONTINUE begins program execution at 00.

If the program is to be recorded on a magnetic card 'END' must be the last (and only the last) step of the program (see Page 74).

The END instruction is also the equivalent of GO TO ( ) ( ), 0, 0; END may, therefore, be used in place of those keys when setting the program counter to address 00 (from the keyboard).

STEP PRGM

STEP PRGM This instruction single-steps the program.
This is the only instruction on the calculator which cannot be included as a program step.

In PROGRAM mode, STEP PRGM single-steps the program, displaying in the X register, the program step address followed by the octal code (Page 58) for the instruction in that address.

In RUN mode, STEP PRGM single-steps the program, displaying the contents of the X, Y, and Z registers after each step. Data must be entered at the proper program steps to check operation of the program.

The use of STEP PRGM is fully detailed under EDITING A PROGRAM on Page 79.

PAUSE

PAUSE Causes a brief return (approximately 1/8 of a second) to the display mode before continuing to the next step of the program. For a longer display, use successive PAUSE instructions. If the PAUSE key is held pressed, during the running of a program, a stop will occur at the next PAUSE instruction in the program (press CONTINUE to continue running the program).

PROGRAM KEYS

The pause instruction is extremely useful in a program which runs for a long time because of branching instructions. The pauses enable the operator to observe, for example, the partial result of the calculation while the program is running. PAUSE also allows the program to be conditionally stopped at a specified point, if the operator so wishes, by holding the PAUSE key depressed (if the STOP instruction were to be included in the program for this purpose, the operator would be forced to remain with the calculator to press CONTINUE after each STOP had occured). Without the PAUSE instruction the display will remain blanked for the entire calculation; the operator has no way of knowing, during this time, whether the program is running correctly or not; in this situation the PAUSE can provide a valuable confidence factor.

PRINT SPACE

PRINT SPACE Used as a PRINT command for peripheral equipment. For complete information refer to the Operating Manual for the particular peripheral. If no printer is being used, the PRINT instruction will act as the STOP instruction; this will require CONTINUE to be pressed to continue running the program.

> **NOTE**
>
> Some 9100A Calculators do not include the word 'SPACE' on the PRINT key; however, operation of this key has not been changed.

FMT

FMT The FORMAT instruction, followed by the next instruction, is used to control peripheral equipment; this allows the calculator to be used as a controller in small systems. For complete information, see the Operating Manual for the particular peripheral.

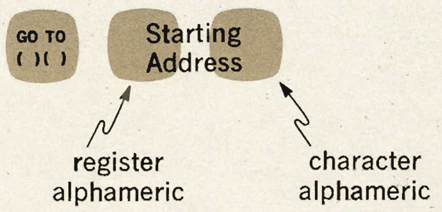
If the FMT is included in a program and no peripheral device is being used, the program will stop after the FMT. If CONTINUE is then pressed, the program will begin running again, starting at the next instruction after the FMT. However, the operation following the FMT is intended to control peripheral equipment and is not part of the main program as such; this means that an undesired operation will be executed as part of the main program. To avoid this, when no peripheral is being used, replace all instructions used to control the peripheral device with CONTINUE's (see 'Correcting a Program', Page 82, for this procedure).

PROGRAM OPERATION

Each of the operations listed below are described and the sample program (shown on Page 66) is used as an illustrative example.

Entering a program from the keyboard.
Recording a program on a magnetic card.
Entering a program from a magnetic card.
Recording program steps and data.
Recording multi-programs on one card.
Editing a program.
Correcting a program.

In this section these keying symbols will be used to address the program counter:



ENTERING A PROGRAM FROM THE KEYBOARD

1. Address the program counter:

SWITCH: **RUN**

PRESS: GO TO ( )( ) Starting Address

> **NOTE**
> When the starting address is 00, END may be used in place of GO TO, 0, 0 to address the program counter.

2. Enter the program:

SWITCH: **PROGRAM**

PRESS: keys in the program sequence.

As each step is entered, the X register displays the address and the octal code for the instruction.

EXAMPLE:

DISPLAY: 0.0---------23 → x

memory location 00 contains (octal) 23, which is the code for the **x→( )** key.

PROGRAM OPERATION

3. Run the program:

SWITCH: RUN

PRESS: GO TO ( )( ) Starting Address

or

PRESS: END (if starting address is 00)

a. The program will now run, beginning with the instruction in the starting address, when CONTINUE is pressed.

b. Enter data and press CONTINUE as necessary for program execution.

EXAMPLE:

enter and run the program for $\frac{A \times B}{A + B}$ (Page 66)

Enter the program:

SWITCH: RUN

PRESS: GO TO ( )( ) 0 0

or

PRESS: END

SWITCH: PROGRAM

PRESS: keys in the program sequence

Observe that, as each step is entered, the X register displays the step address and the key code.

Address the program counter:

SWITCH: RUN

PRESS: END

CONTINUED

PROGRAM OPERATION

ENTERING A PROGRAM FROM THE KEYBOARD

CONTINUED

Enter data (A and B):

PRESS: digit keys for A

PRESS: ↑

PRESS: digit keys for B

Run the program:

PRESS: CONTINUE

DISPLAY:

$$\begin{array}{lcl} \dfrac{A \times B}{A + B} & \rightarrow & z \\ A & \rightarrow & y \\ B & \rightarrow & x \end{array}$$

To rerun the program, enter data as before and press CONTINUE. In this program it is not necessary to re-address the program counter each time as the END instruction automatically resets the counter to 00.

RECORDING A PROGRAM ON A MAGNETIC CARD

RECORD

The magnetic card has two parts, A and B. The entire contents of the memory (with the exception of the e, f, X, Y and Z registers) may be recorded on either part of a card. To record (or enter) part A, insert the card into the cardreader slot with the A arrow pointing down. Ensure that the printed side of the card is facing the keyboard. To record (or enter) part B, insert the card in the same manner with the B arrow pointing down.

> **NOTE**
>
> 'END' must be the last (and only the last) instruction in the program; otherwise the program cannot be re-entered correctly into the calculator once it has been recorded. If it is required to record data following the program steps, refer to Page 76 for the procedure.

To record a program:

SWITCH: **RUN**

PROGRAM OPERATION

PRESS: GO TO ( )( ) Starting Address

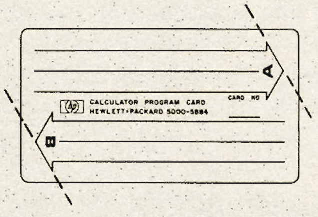
INSERT: magnetic card into the card reader.

PRESS: RECORD (hold the key pressed until the card is partially ejected).

> **NOTE**
> The contents of the X register are destroyed when a program is recorded.

A new program can be recorded over an existing program by repeating the record procedure with the new program.

To permanently save a program recorded as program A or B, cut off the corner of the card along the tip of the A or B arrow.



ENTERING A PROGRAM FROM A MAGNETIC CARD

ENTER

SWITCH: RUN

STOP

PRESS: GO TO ( )( ) Starting Address

INSERT: magnetic card into the card reader.

PRESS: ENTER

> **NOTE**
> The contents of the X, Y and Z registers remain unchanged.

The program is now inserted and ready to run as before.

PROGRAM OPERATION

RECORDING PROGRAM STEPS AND DATA

If both program steps and data are to be recorded on a magnetic card, care must be taken of the position of the 'END' statement.

In order to understand the significance of the 'END' instruction, a brief description of the magnetic card and of the recording and entering processes is in order. (Refer to the diagram of the magnetic-core memory, Figure 2, Page 57, throughout this discussion.)

> **NOTE**
>
> During the 'record' and 'enter' process the calculator cannot distinguish between a program instruction and a data digit.

Either part (A or B) of the magnetic card is long enough to record approximately one and a half times the contents of the programmable memory (registers 0 through *d*); the exact amount recorded will differ between cards and between calculators.

The recording (and entering) process may be started at any point (selected by the operator) in the calculator's memory; however, regardless of the starting position in the memory, the steps are loaded onto (and read from) the magnetic card starting at the top.

During the recording process the calculator steps sequentially through the memory to address *dd*; at this point it automatically goes to address 00 and continues recording until the magnetic card is completely filled. The card now contains approximately one and a half times the contents of the memory (half of which is recorded twice); however, the last step recorded is probably not recorded correctly because the card stops moving at the same time as that step is recorded.

When the program on the card is re-entered into the calculator, the steps are read from the card (and loaded sequentially into the memory) until an END instruction is encountered on the card. After the END has been loaded, the calculator stops reading from the card and resets to address 00. Anything following the END on the card is thus not entered into the calculator memory.

Assume, now, that there is no END statement on the card; also, assume that (as will most often be the case) the recorded program starts at address 00. During the entering process the steps on the card will be loaded sequentially into the memory until address *dd* is filled; at this point, the calculator will automatically reset to 00 and load steps over those steps previously loaded into the memory. As the new steps are actually duplicates of the steps loaded during the first pass through the memory, this will not affect the stored program until the last step on the card is reached. As that step was probably not recorded correctly there will now (probably) be a wrong program

PROGRAM OPERATION

step in the memory; there is no way of predetermining at which address this step will fall. Hence the need for the END statement to stop the card reader during entry; the (probably) wrong step is then never loaded into the memory.

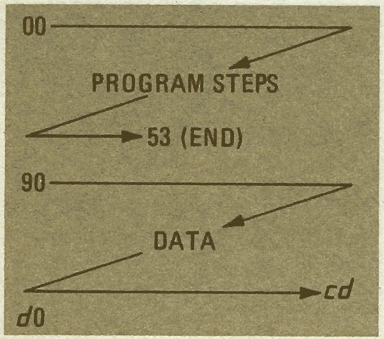
From the foregoing discussion it can be seen that if both program steps and data are to be recorded, then the END must somehow be recorded after the data; otherwise the data cannot be re-entered into the calculator. There are several ways to do this; two of the simpler methods are described here.

METHOD I

Assume that the program to be recorded consists of (as diagrammed below):

1. program steps from address 00 to 53 (END is in address 53);
2. data in the 9, *a*, *b* and *c* registers; one register (in this case *d*) is not available for data storage as the END must now be moved to this register.

The diagram represents this program as it would appear in the calculator's memory.



Before the program and data can be recorded on the magnetic card:

1. Change the instructions in addresses 53, 54 and 55 to GO TO, *d*, 0 (see Correcting a Program on Page 82 for this procedure) - this will cause the program to unconditionally branch to address *d*0 when the program is run;
2. insert END in address *d*0;
3. insert CONTINUE ('no operation') in all of the unused addresses between the program steps and the data (addresses 56 through 8*d*) -this ensures that there is no END in any of these addresses.

The program and data may now be recorded (starting at address 00) and re-entered into the calculator, as previously described.

PROGRAM OPERATION

RECORDING PROGRAM STEPS AND DATA

CONTINUED

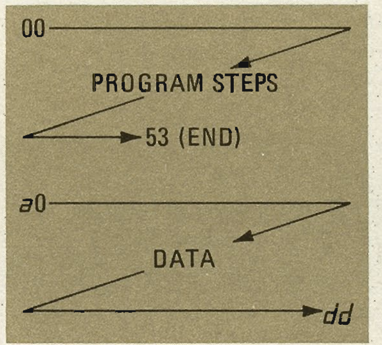
METHOD II

If this second method is used to record program steps and data, it is not necessary to change any program steps; neither is it necessary to reserve a complete register (as in Method I) for the 'END' instruction.

Assume that the program to be recorded consists of (as diagrammed below):

1. program steps from address 00 to 53 (END is in address 53);
2. data in the *a*, *b*, *c* and *d* registers.

The diagram represents this program as it would appear in the calculator's memory.



The program and data may be recorded, without any changes, simply by starting the recording process at (in this example) address *a*0 instead of address 00. Remember that, when the instruction in address *dd* has been recorded, the calculator will automatically continue recording at address 00.

When the recorded program is to be entered into the calculator, again start the process at address *a*0. The data will be loaded into the *a*, *b*, *c* and *d* registers and then the program steps will be loaded into address 00 and following. When the END is loaded, at step 53, the card reader will stop reading program steps and reset to 00. The program is now ready to be run, starting at address 00.

Whenever this method is used be sure to avoid future confusion; mark on the magnetic card that entry is to be started at some address other than 00 (in this case *a*0) but that the program starts at 00.

PROGRAM OPERATION

RECORDING MULTI-PROGRAMS ON ONE CARD

It is also possible to record more than one program on a magnetic card. In this case the END instruction must appear only after the last program.

Once the programs have been recorded on a magnetic card it is not possible to enter only one of the programs into the calculator from that card. This is because, during the record and enter processes, steps are recorded on and read from the card, starting at the top of the card (regardless of the starting address in the calculator's memory). It is not possible to start either the record or entry process part way down the card.

EDITING A PROGRAM

When a program has been entered into the calculator it may be checked step by step, either in the RUN or PROGRAM mode, by use of the STEP PRGM (step program) key.

1. **PROGRAM MODE:** used to verify that the program steps have been entered as written in the program.

    SWITCH: **RUN**

    PRESS: GO TO ( )( ) Starting Address

    SWITCH: **PROGRAM**

    PRESS: STEP PRGM

    Each time the STEP PRGM key is pressed the program counter is incremented by one count. The X register will display the last program step address and the octal code for the program step at that address. Refer to either the pull-out card (at the front of the calculator) or to the fold-out in the Appendix of this manual for the octal codes for each key.

> **NOTE**
> When in the PROGRAM mode, pressing any key, other than STEP PRGM, will cause the instruction of the key pressed to be entered as a program step.

As an example, step through the program previously used (Page 66) and observe that the correct address and (octal) key codes are displayed in the X register. Switch back to RUN mode any time the program counter is to be re-addressed. If an incorrect step is found, refer to Correcting a Program (Page 82).

PROGRAM OPERATION

EDITING A PROGRAM

CONTINUED

If a complete register is found to contain incorrect program steps, then there is probably an error in the program such that, when the program is first run, it loads data into a register containing program steps. Data loaded in this way will frequently appear as a series of (octal) 40's and 60's (in this case the 40 represents a blanked digit zero and the 60 represents a blanked, negative digit zero).

2. **RUN MODE:** used to verify that the program will operate as written.

SWITCH: **RUN**

PRESS: GO TO ( )( ) Starting Address

Press the STEP PRGM key at each step; data must be entered at the correct program steps to check operation and solution of the program. At each step the X, Y and Z registers will display the results of the operation.

EXAMPLE:

edit the $\frac{A \times B}{A + B}$ program (Page 66) in the RUN mode:

(address the program counter)

SWITCH: **RUN**

PRESS: END

(enter data)

PRESS: any digit keys for A

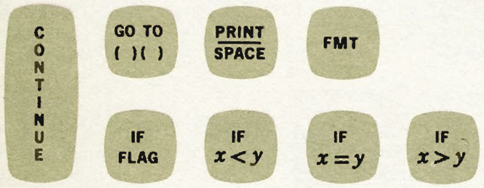
PRESS: ↑

PRESS: any digit keys for B.

Step the program by pressing the STEP PRGM key; at each step verify that the X, Y and Z registers contain the results as predicted in the program.

PROGRAM OPERATION

In the RUN mode (only), several instructions will not act as expected when the STEP PRGM key is being used. These are:



CONTINUE - when STEP PRGM is pressed at this instruction the CONTINUE will override the STEP PRGM condition and automatically continue execution of the program. Replace any CONTINUE with a STOP instruction (see 'Correcting a Program' Page 82) before editing in the RUN mode.

GO TO - when STEP PRGM is pressed at this instruction the calculator will automatically go to the address designated by the next two program steps. The next time STEP PRGM is pressed, the operation in the new address is executed. As an example, suppose the following is part of a program to be edited in the RUN mode:

STEP	KEY
----	----
03	+
----	----
07	GO TO ( )( )
08	0
09	3

when STEP PRGM is pressed (apparently) at step 08 the plus operation in step 03 will be executed.

FMT and PRINT/SPACE - both of these instructions are used to control peripheral equipment. When a program is running, the program stops at these instructions and waits for a 'continue' signal from the peripheral equipment before continuing with the program. In the RUN mode when STEP PRGM is pressed at either of these instructions the 'continue' signal from the peripheral equipment will cause the program to continue running. To avoid this, disconnect any peripherals before editing a program in the RUN mode.

PROGRAM OPERATION

EDITING A PROGRAM

CONTINUED

IF x <y, IF x = y, IF x >y, IF FLAG - these instructions are used for conditional branching (see Pages 87 and 91 for a full description of these keys). All four instructions cause an immediate branch when the STEP PRGM is used in the RUN mode.

EXAMPLES:

a)

STEP	KEY
----	----
42	IF $x = y$
43	↑
44	×
45	*d*
----	----

(IF x =y compares the numbers, in the X and Y registers, for equality.) When STEP PRGM is pressed at step 42:

if x = y, the operation in step 43 will be executed.
if x ≠ y, step 45 will be executed (the contents of the *d* register will be recalled to the X register).

b)

STEP	KEY
----	----
21	×
----	----
34	IF $x = y$
35	2
36	1
37	÷
----	----

When STEP PRGM is pressed at step 34:

if x = y, the branch to step 21 is executed immediately. When STEP PRGM is pressed (apparently) at step 35 the multiply operation in step 21 is executed.
if x ≠ y, the division in step 37 is immediately executed.

CORRECTING A PROGRAM

Any program step can be changed or deleted without re-entering the entire program into the calculator.

Extra steps can also be inserted into a program (as explained on Page 84 and following).

SWITCH: **RUN**

PROGRAM OPERATION

PRESS: GO TO ( )( ) Address of step to be changed

SWITCH: PROGRAM

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

the corrected step will appear in the X register.

SWITCH: RUN

EXAMPLE:

change the $\frac{A \times B}{A + B}$ program (Page 66) to solve $\frac{A \times B}{A - B}$

SWITCH: RUN

PRESS: GO TO ( )( ) 0 7

SWITCH: PROGRAM

PRESS: –

DISPLAY: 0.7---------34 → x

SWITCH: RUN

The program may now be run as before except that the solution will be for $\frac{A \times B}{A - B}$; also the X, Y and Z registers, as shown in the program, will contain A – B in place of A + B.

Steps can also be inserted into a program. If the program is short, it is faster to re-enter the program manually, starting at the first incorrect step; if the program is long, the magnetic card may be used, as illustrated in the following example, or a 'patch-in' method may be used (this will be explained after the next example).

PROGRAM OPERATION

CORRECTING A PROGRAM
CONTINUED

METHOD I

The steps shown below are part of a (purely hypothetical) long program and it is required to insert two more instructions (say, ROLL ↑ and $|y|$) between steps 64 and 65.

STEP	KEY	KEY CODE
----	----	----
62	ACC +	60
63	↑	27
64	RCL	61
65	+	33
66	$\sqrt{x}$	76
67	END	46

In order to make these changes first enter the program into the calculator:

SWITCH: RUN

PRESS: GO TO ( )( ) 6 2

SWITCH: PROGRAM

PRESS: the keys in the program sequence.

To correct the program:

1) Record the program, starting from address 65, on a magnetic card:

SWITCH: RUN

PRESS: GO TO ( )( ) 6 5

INSERT: magnetic card

PRESS: RECORD

2) Insert the two extra instructions in locations 65 and 66:

PRESS: GO TO ( )( ) 6 5

PROGRAM OPERATION

SWITCH: **PROGRAM**

PRESS: ↑ ROLL |y|

3) Re-enter the remaining steps (recorded on the program card) into address 67 and following:

SWITCH: **RUN**

INSERT: program card

PRESS: ENTER

the steps on the card will automatically be moved down the required two steps in the memory. The program sequence of the example will now be:

		KEY
STEP	KEY	CODE
----	----	----
62	ACC +	60
63	↑	27
64	RCL	61
65	ROLL ↑	22
66	|y|	55
67	+	33
68	$\sqrt{x}$	76
69	END	46

4) If any branching instructions are included in the program, change the alphameric instructions, where necessary, to conform to the changed addresses.

METHOD II

Using the previous technique to insert steps can have a major disadvantage (especially if a large part of the program has been moved to new locations in the memory); changing the branching instructions in the program may itself be a source of further errors.

To avoid such errors, a 'patch-in' technique (described below) may be used to insert steps in the program. From the point of view of efficient programming this technique is 'sloppy', however, it is fast and very effective.

PROGRAM OPERATION

CORRECTING A PROGRAM

CONTINUED

Using the same example as previously, it is required to insert two extra instructions (ROLL ↑ and $|y|$) between steps 64 and 65.

STEP	KEY	KEY CODE
----	----	----
62	ACC +	60
63	↑	27
64	RCL	61
65	+	33
66	$\sqrt{x}$	76
67	END	46

To insert steps:

1) Insert (any convenient) branching address in the steps preceeding step 65;

STEP	KEY	KEY CODE
----	----	----
62	GO TO ( )( )	44
63	7	07
64	0	00
65	+	33
66	$\sqrt{x}$	76
67	END	46

2) at the new address, first insert the deleted steps, then the new steps, then branching instructions to return to step 65.

STEP	KEY	KEY CODE
70	ACC +	60
71	↑	27
72	RCL	61
73	ROLL ↑	22
74	$\|y\|$	55
75	GO TO ( )( )	44
76	6	06
77	5	05

PROGRAM KEYS – CONDITIONAL BRANCHING

Four keys located in the right hand block of the keyboard, are used for conditional branching (see also Page 59).

IF x <y, IF x = y, IF x >y and IF FLAG (which operates in conjunction with SET FLAG).

IF $x<y$

IF $x=y$

IF $x>y$

These three instructions compare the numbers contained in the X and Y registers. The indicated condition (e.g. is X equal to Y?) is tested. If the condition is met (YES) the program continues with the next step. If the condition is not met (NO) the program skips the next two steps and continues.

IF $x<y$

IF $x=y$

IF $x>y$

NOTE

The 'IF' instructions compare all twelve digits (i.e. ten displayed digits and two guard digits) and the two-digit exponent, of the numbers in the X and Y registers, when testing for the condition indicated. However, if a number consists of digit nine (9) in the first eleven places and (one of digits) five (5) through nine (9) in the twelfth place, then that number is considered to be equal to the next higher (more positive) power of ten. For example,

9.999 999 999 9(5 thru 9) x $10^2 = 10^3$

similarly,

9.999 999 999 9(5 thru 9) x $10^{-2} = 10^{-1}$

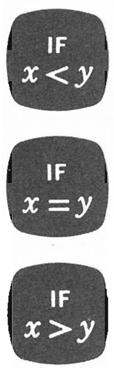
Despite their equality to the next higher power of ten, those numbers which have unequal digits only in the twelfth place are not considered to be equal to each other. Thus for example,

9.999 999 999 95 x 10^2 is not equal to
9.999 999 999 96 x 10^2.

CONDITION MET: When an 'IF' qualifier is met (YES) a 'GO TO' condition is established. If the two steps following the 'IF' instruction contain an address, the program will branch to that address. If the steps following the 'IF' contain operations, and not an address, then the 'GO TO' condition is cleared and the operations are executed (the 'GO TO' is not a required step).

CONDITION NOT MET: When an 'IF' qualifier is not met (NO) the next two steps are skipped and the program continues.

PROGRAM KEYS – CONDITIONAL BRANCHING



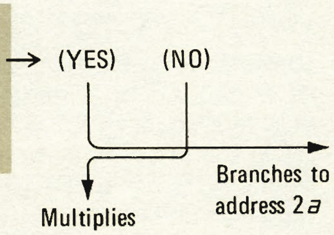
CONTINUED

The following two examples are parts of imaginary programs containing an IF instruction. In each case the course of the program is mapped for 'condition met' (YES) and 'condition not met' (NO).

EXAMPLE (1):

Address in the two steps following the 'IF' instruction:

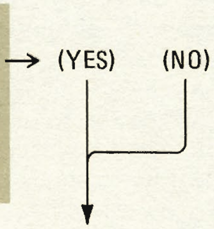
STEP	KEY
23	IF $x = y$
24	2
25	a
26	×



EXAMPLE (2):

Operations in the two steps following the 'IF' instruction:

STEP	KEY
23	IF $x = y$
24	TO POLAR
25	↑
26	×
27	----



The following program illustrates use of an IF key (in this case IF x <y):

This program sums any number (N) until the total (Σ_n) is equal to, or greater than, 100.

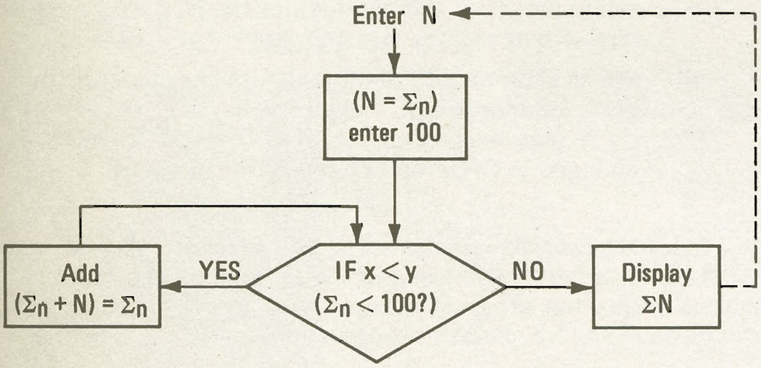
The final display shows N in the Z register, 100 in Y and ΣN (the final Σ_n) in X.

'IF x <y' is used to test if Σ_n (or initially N) is equal to, or greater than, 100.

If N is negative, 100 is never reached and the program sums (–)N to (–) infinity.

PROGRAM KEYS – CONDITIONAL BRANCHING

FLOWCHART:



PROGRAM

to run the program press END, enter N into the X register and press CONTINUE.

		KEY	DISPLAY		
STEP	KEY	CODE	X	Y	Z
00	↑	27	N	N	--
01	↑	27	N	N	(N) = Σ_n
02	ENTER EXP	26	1	N	Σ_n
03	2	02	100	N	Σ_n
04	ROLL ↑	22	Σ_n	100	N
				flashing display	
05	PAUSE	57	Σ_n	100	N
06	IF x<y	52	Σ_n	100	N
07	0	00	Σ_n	100	N
08	a	13	Σ_n	100	N
				final display	
09	END	46	$\Sigma_n = \Sigma N$	100	N
0a	ROLL ↑	22	N	Σ_n	100
0b	+	33	N	$(\Sigma_n + N) = \Sigma_n$	100
0c	ROLL ↓	31	Σ_n	100	N
0d	GO TO ( )( )	44	Σ_n	100	N
10	0	00	Σ_n	100	N
11	5	05	Σ_n	100	N

At step 06, if Σ_n is less than 100 the condition tested is met, the program continues to step 07 and 08 and, as these contain an address, it branches (to step 0a). When Σ_n is not less than 100 (i.e. $\Sigma_n = \Sigma N \geq 100$) the condition is not met; steps 07 and 08 are skipped and the instruction in 09 is executed.

The above program also illustrates the following:

a) CLEAR is not always a necessary instruction;

PROGRAM KEYS – CONDITIONAL BRANCHING

IF $x < y$

IF $x = y$

IF $x > y$

CONTINUED

b) use of PAUSE to view partial results of the program;

c) manipulation of the contents of the X, Y and Z registers so that storage registers are not required;

d) use of END, at step 09, to automatically reset the program counter to 00; if the program is to be recorded on a magnetic card the END instruction must be changed to STOP and END inserted in step 12.

The following program calculates N! (N Factorial). A new value for N can be entered at the end of the program, since END automatically sets the program counter back to 00. This program can calculate N! for values of N up to, and including, 69.

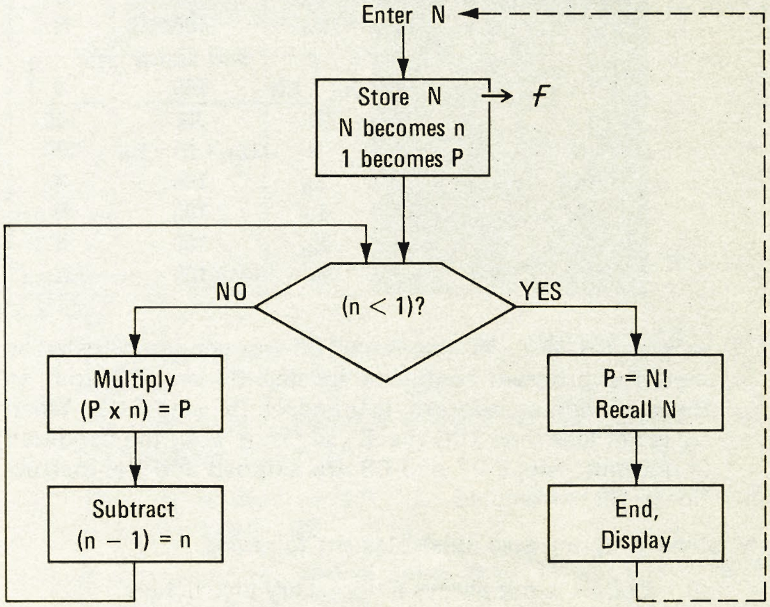
$$N! = N(N-1)(N-2)\cdot\cdot\cdot\cdot\cdot(N-N+2)(1)$$

$$6! = (6)(5)(4)(3)(2)(1) = 720$$

$$0! = 1 \text{ (by definition)}$$

(In the program) N = number entered;
n = the current multiplier term N, (N – 1), (N – 2), etc.
P = the partial N!

FLOW CHART



PROGRAM KEYS – CONDITIONAL BRANCHING

PROGRAM

To run the program: press 'END', enter N into the X register and press 'CONTINUE'.

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f
00	$x \to (\ )$	23	N	-	-	-
01	f	15	N	-	-	N
02	↑	27	N	N = n	-	
03	1	01	1	n	-	
04	↑	27	1	1	n	
05	ROLL ↓	31	1	n	1 = P	
06	IF $x > y$	53	1	n	P	
07	1	01	1	n	P	
08	2	02	1	n	P	
09	ROLL ↓	31	n	P	1	
0a	×	36	n	(P x n) = P	1	
0b	ROLL ↑	22	1	n	P	
0c	−	34	1	(n − 1) = n	P	
0d	GO TO ( )( )	44	1	n	P	
10	0	00	1	n	P	
11	6	06	1	n	P	
12	f	15	N	n = 0	P = N!	
				final display		
13	END	46	N	0	N!	

IF FLAG

A conditional branching instruction which tests a 'YES' or 'NO' condition stored in the calculator. The condition is set to 'YES', by the SET FLAG instruction, either from the keyboard or as a program step.

Branching after the IF FLAG instruction is the same as for the conditional branch instructions previously described (IF x $<$y, IF x $=$ y, and IF x $>$y).

CONDITION MET: FLAG SET ('YES') - the program continues with the next step following the IF FLAG instruction and also clears the flag to the 'NO' condition. If the two steps following the 'IF FLAG' are alphamerics (an address) then the program branches to the indicated address. If these are operations (not alphamerics) then the operations are executed.

CONDITION NOT MET: FLAG NOT SET ('NO') - the program skips the next two steps following the IF FLAG instruction and continues with the third step. (See SET FLAG for example of the use of the IF FLAG instruction).

IF FLAG

SET FLAG

PROGRAM KEYS – CONDITIONAL BRANCHING

IF FLAG

SET FLAG

CONTINUED

SET FLAG **Sets the 'YES' condition which is tested by the IF FLAG instruction. May be set either from the keyboard, when the program is stopped, or as a program step.**

The 'YES' (SET FLAG) condition is cleared (to 'NO) when the IF FLAG instruction is encountered in a program; it is also cleared when a CLEAR instruction is encountered, either as a program step or (when the program is stopped) from the keyboard.

The following program calculates the average value $(\overline{X})$ of N data points, X_i:

$$\frac{X_1 + X_2 + X_3 + \cdots\cdot X_N}{N} = \frac{\Sigma X_i}{N} = \overline{X}$$

This program illustrates the following:

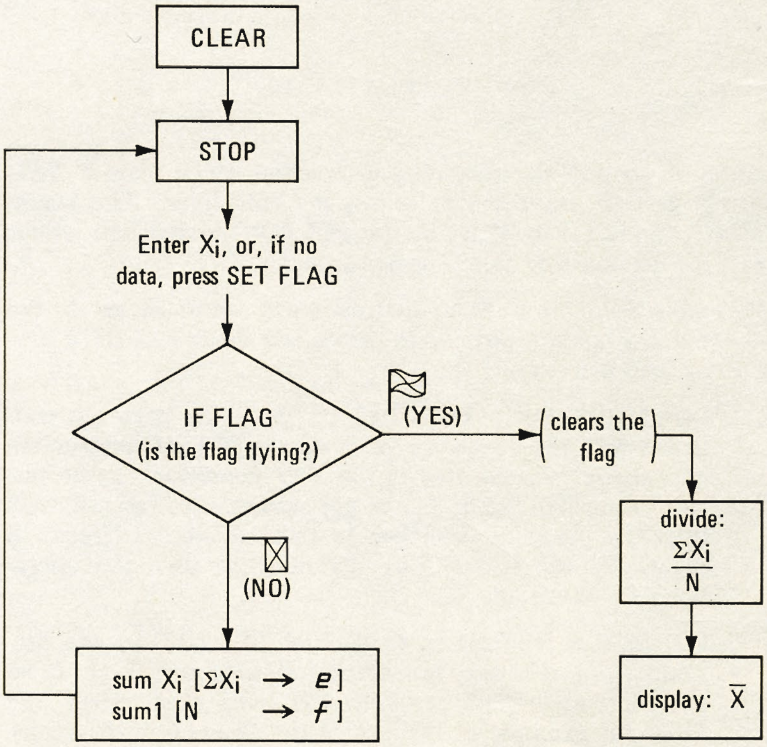
a) use of IF FLAG, with the SET FLAG instruction set from the keyboard;

b) generation of a counter (the number of data points is counted by adding 1 to the counter each time a data point is entered);

c) use of the accumulative registers, *e* and *f*;

d) use of the CLEAR to initially clear *e* and *f* and to clear the SET FLAG (to 'NO').

FLOW CHART


CLEAR
STOP
Enter X_i, or, if no data, press SET FLAG
IF FLAG (is the flag flying?)
(YES)
clears the flag
divide: $\frac{\Sigma X_i}{N}$
display: $\overline{X}$
(NO)
sum X_i [$\Sigma X_i \rightarrow e$]
sum1 [N $\rightarrow f$]

PROGRAM KEYS – CONDITIONAL BRANCHING

To operate the program:

1) PRESS: END CONTINUE

2) enter X_1 in the X register

3) PRESS: CONTINUE

4) enter X_2 in the X register

5) PRESS: CONTINUE

6) repeat steps 4 and 5 for all remaining data points up to, and including, X_N

7) PRESS: SET FLAG CONTINUE

		KEY	DISPLAY			STORAGE	
STEP	KEY	CODE	X	Y	Z	f	e
00 b4	CLEAR	20	0	0	0	0	0
01	STOP	41	[Enter X_i]				
02	IF FLAG	43					
03 c	0	00					
04 1	b	14					
05	↑	27	X_i	X_i	(X_{i-1})		
06	1	01	1	X_i	(X_{i-1})		
07	ACC +	60	1	X_i	(X_{i-1})	N+1 = N	ΣX_i
08	GO TO ( )( )	44					
09	0 b	00					
0a ac	1 5	01					
0b c1	RCL	61	N	ΣX_i	(X_{i-1})	N	ΣX_i
0c	÷	35	N	$\frac{\Sigma X_i}{N}$	(X_{i-1})		
0d c3	END	46	$\boxed{N}$	$\boxed{\bar{X}}$	$\boxed{(X_{i-1})}$	→ final	display

PROGRAM KEYS – CONDITIONAL BRANCHING

IF FLAG

SET FLAG

CONTINUED

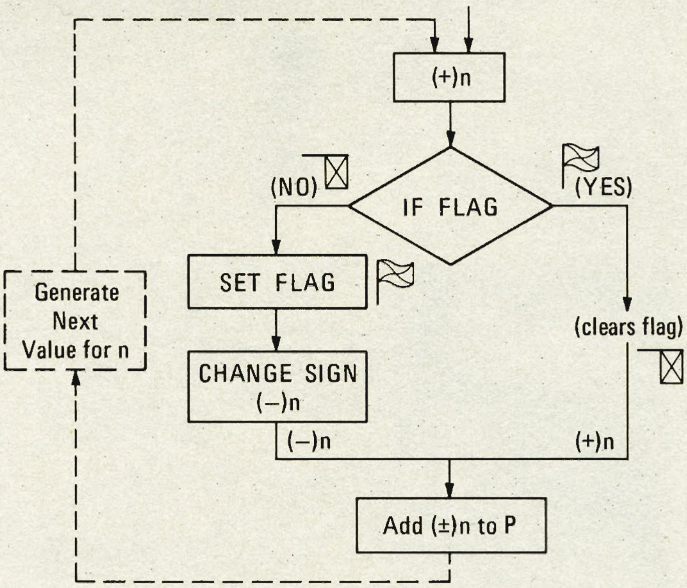
The next program illustrates use of the SET FLAG instruction as a program step. This program calculates the following alternating series:

$$1 - 3 + 5 - 7 + 9 \text{ - - - - - - - - - - - - } \infty$$

> **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 (±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

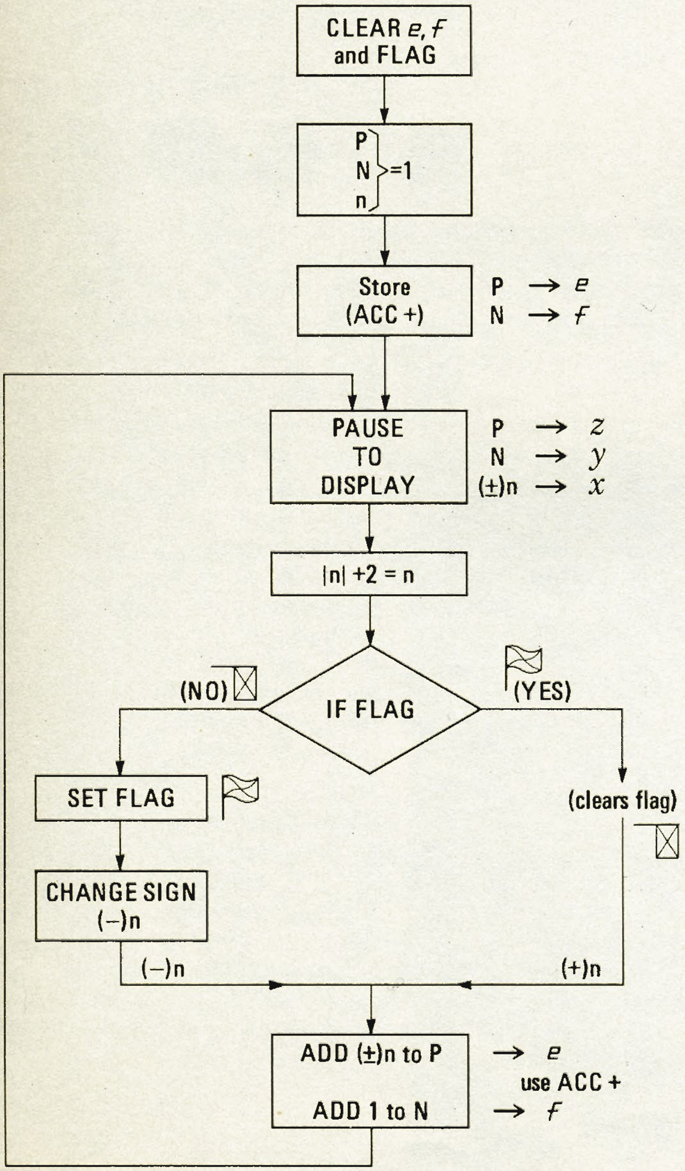
The next time through, the flag is not set:

a) negative (–)n is added to P;

b) the flag is set (by a program step) so that the next time through a (+)n will be added.

FLOW CHART: for series 1 – 3 + 5 – 7 + 9 · · · · · ∞

n = current term of the series
N = number of terms added
P = the accumulative total of the series.



IF FLAG

SET FLAG

CONTINUED

PROGRAM KEYS – CONDITIONAL BRANCHING

To operate the program:

PRESS: END CONTINUE

To stop the program and display P, N and n:

PRESS: PAUSE ; press CONTINUE to continue running the program.

To restart the program:

PRESS: PAUSE (or STOP) END CONTINUE

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE f	STORAGE e
00	CLEAR	20	0	0	0	0	0
01	1	01	1 = P	0	0		
02	↑	27	1 = N	P	0	↓	↓
03	ACC +	60	N	P	0	N	P
04	↑	27	1 = n	N	P		
05	PAUSE	57	flashing display				
06	PAUSE	57	±n	N	P		
07	↑	27	±n	±n	N		
08	|y|	55	±n	|n| = n	N		
09	2	02	2	n	N		
0a	+	33	2	n+2 = n	N		
0b	↓	25	n	N	N		
0c	IF FLAG	43					
0d	1	01					
10	3	03					
11	SET FLAG	54					
12	CHG SIGN	32	-n	N	N		
13	↑	27	±n	±n	N		
14	1	01	1	±n	N	↓	↓
15	ACC +	60	1	±n	N	N+1 = N	P+(±)n =P
16	↑	27	1	1	±n	N	P
17	RCL	61	N	P	±n		
18	ROLL ↑	22	±n	N	P		
19	GO TO ( )( )	44					
1a	0	00					
1b	5	05					
1c	END	46				↓	↓

PROGRAM KEYS – CONDITIONAL BRANCHING

'POSSIBLE COMBINATIONS'

This program is a general program which further illustrates some of the programming techniques previously described. The program calculates possible combinations (C) of n objects taken k at a time; for example:

> If a box contains (n) 15 different 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 is $C_k^n = \frac{n!}{k!\,(n-k)!}$. The program will calculate C for any values of n and k up to, and including, 69.

The program illustrates the following:

1. Use of a sub-program in a loop to calculate the value for n!, k! and (n – k)!. The N! program (Page 91) is used.
2. Use of a counter to count the number of times through the loop; one (1) is subtracted from the counter (x) each time through the loop. When x = 0 the program branches out of the loop and calculates the final value for C.
3. Use of the $y \rightleftarrows ()$ key to move data through the memory, so that, each time through the loop, the proper value for N (n, k and n – k) can be recalled.
4. The original values of n and k, stored in the memory, are lost because the $y \rightleftarrows ()$ key is used; therefore n and k are stored twice in the memory so that they can be recalled for the final display. This is purely an added convenience which is not essential to the program.

NOTE

Compare the counting technique used in this program with the technique used in the N! program where the multiplier (n) also acts as the counter. One (1) is subtracted from n each time through the loop and the question "is n less than one?" is used to determine whether to return through the loop or to branch out of it. Thus the number of passes through the loop depends on the initial value of N entered.

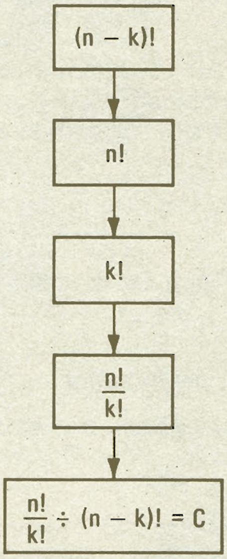
In the present program, the number of passes through the loop (3) is written into the program.

PROGRAM KEYS — CONDITIONAL BRANCHING

'POSSIBLE COMBINATIONS'

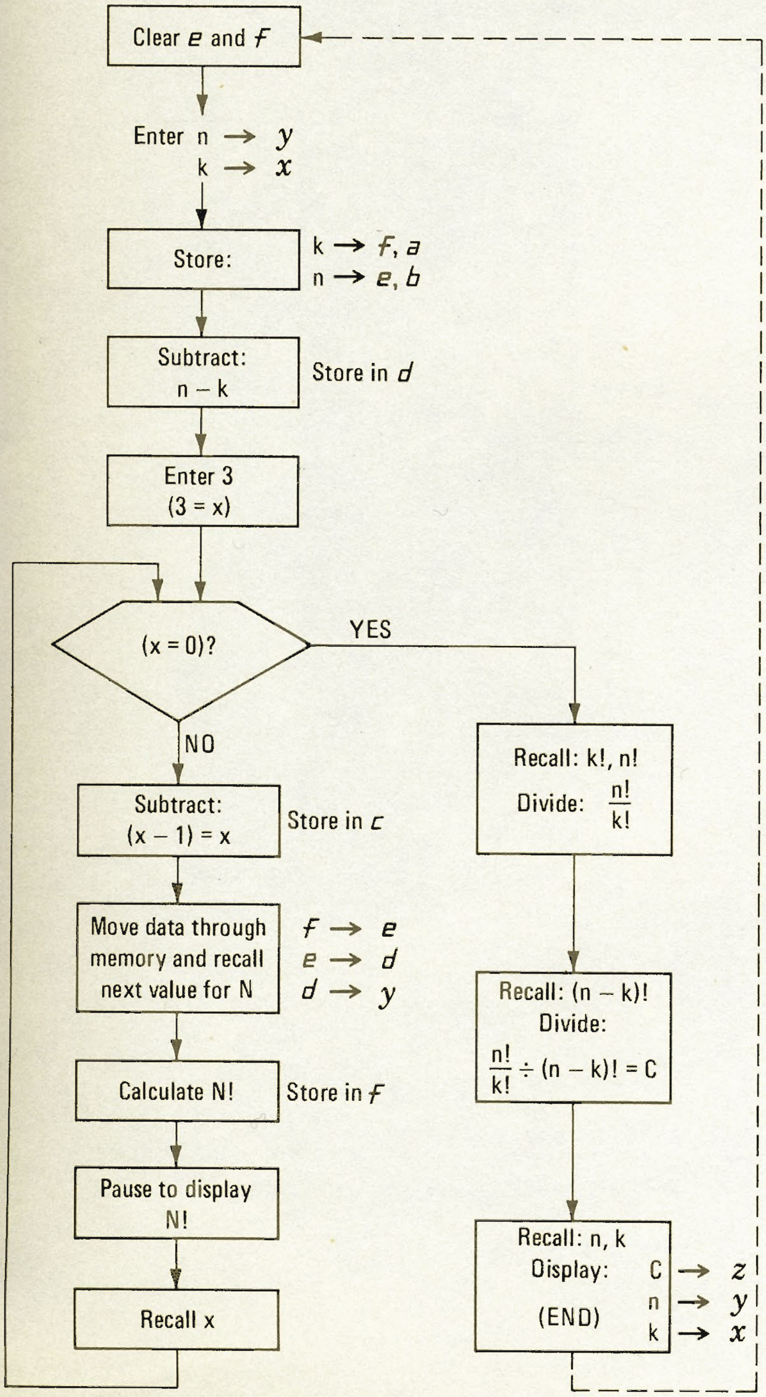
CONTINUED

Here is the simplified flow chart:



PROGRAM KEYS – CONDITIONAL BRANCHING

FINAL FLOW CHART (refer also to the table showing the contents of the storage registers, on Page 101).



PROGRAM KEYS – CONDITIONAL BRANCHING

'POSSIBLE COMBINATIONS'

CONTINUED

To run the program: press END, CONTINUE; then enter n into Y, k into X and press CONTINUE.

			KEY	DISPLAY		
	STEP	KEY	CODE	X	Y	Z
	00	CLEAR	20	0	0	0
				Enter		
	01	STOP	41	k	n	
	02	ACC +	60	k	n	
	03	x→( )	23			
	04	a	13	k	n	
	05	y→( )	40			
	06	b	14	k	n	
	07	—	34		n − k	
	08	y→( )	40			
	09	d	17		n − k	
Start of	0a	3	03	3 = x		
Loop →	0b	↑	27		x	
	0c	0	00	0	x	
	0d	IF x = y	50			
	10	3	03			
	11	6	06			
	12	1	01	1	x	
	13	—	34		(x − 1) = x	
	14	y→( )	40			
	15	c	16		x	
	16	y⇄( )	24			
	17	f	15			
	18	y⇄( )	24			
	19	e	12			

CONTINUED

PROGRAM KEYS – CONDITIONAL BRANCHING

			KEY	DISPLAY		
	STEP	KEY	CODE	X	Y	Z
	1a	y⇄( )	24			
N! SUB-PROGRAM	1b	d	17		N	
	1c	1	01			
	1d	↑	27			
	20	ROLL ↓	31			
	21	IF x > y	53			
	22	2	02			
	23	b	14			
	24	ROLL ↓	31			
	25	×	36			
	26	ROLL ↑	22			
	27	−	34			
	28	GO TO ( )( )	44			
	29	2	02			
	2a	1	01			
	2b	CLEAR x	37	0	0	N!
	2c	PAUSE	57		flashing display	
	2d	↓	25		N!	
	30	y→( )	40			
	31	f	15		N!	
	32	c	16	x		
	33	GO TO ( )( )	44			
End of	34	0	00			
Loop →	35	b	14			
	36	RCL	61	k!	n!	
	37	÷	35		n!/k!	
	38	d	17	(n − k)!	n!/k!	
	39	÷	35		C	
	3a	b	14	n	C	
	3b	↑	27	n	n	C
	3c	a	13	k	n	C
	3d	END	46		final display	

STORAGE REGISTERS

REGISTER	INITIAL DATA STORAGE	AFTER EACH PASS THROUGH LOOP		
		1st PASS	2nd PASS	3rd PASS
f	k	(n − k)!	n!	k!
e	n	k	(n − k)!	n!
d	n − k	n	k	(n − k)!
c	----	x = 2	x = 1	x = 0
b	n	→	→	→
a	k	→	→	→

ADDITIONAL PROGRAMMING

EFFICIENT USE OF THE MEMORY

Here are some guidelines to efficient use of the memory.

> **NOTE**
>
> Program steps and data cannot both be stored at the same time in one register.
>
> The *e* and *f* registers are used only for data storage.
>
> The contents of *e* and *f* (and the X, Y and Z) registers cannot be recorded on a magnetic card.

Generally, start program steps at address 00 and sequentially fill each register; work down the memory as each register is filled.

Store data starting at the bottom of the memory (register *f*) and work upwards through the alphabetic registers (*f* through *a*) before storing data in the numeric registers (9, 8, 7, etc.). The alphabetic registers are used first, as recall from these registers requires only one keystroke. Also, recall from the numeric registers requires two keystrokes and destroys the contents of the register recalled.

When applicable, reserve the *e* and *f* registers for use with the ACC+, ACC− and RCL instructions. Even if these registers are not to be used as accumulators they are extremely useful for storing data; two numbers can be stored simultaneously with one keystroke (ACC+) and recalled with one keystroke (RCL). Be sure that the *e* and *f* registers are cleared before storing data by means of the ACC+ instruction.

'DESTRUCTIVE PROGRAMS'

When program steps do not leave sufficient memory space for data storage, use a 'destructive' routine. In this case, program steps which will be required only once during the program, can be started at (for example) address *a*0. After these steps have been used, the registers containing them can then be used for data storage.

This technique requires recording the program on a magnetic card, the steps are reinserted into the memory each time the program is to be run.

ADDITIONAL PROGRAMMING

CASCADING MAGNETIC CARDS

'Cascading magnetic cards' is a variation of the destructive routine used when programs too large for the memory are required.

In this case the first part of the program is entered from the keyboard and recorded; this is repeated with the next part of the program, and so on, until the complete program is recorded on a series of cards. The first card is re-entered and that part of the program run; this is repeated with the remaining cards until the program is completed.

Care must be taken to ensure that, as each card is entered, data stored in the registers is not destroyed. The END instruction must be used on each card; this will stop the card reader at the required point in the memory and ensure that stored data is not affected.

USE OF PERIPHERAL EQUIPMENT

If peripheral equipment is to be used at a later date, it is advisable, if possible, to make provision for this at the time the program is written. Insert a 'no operation' in each step required to control the peripheral equipment. The CONTINUE instruction may be used as a 'no operation'. This allows the program to be run now; then, when required, extra steps can be easily inserted and it will not be necessary to change the addressing instructions in the program.

For the Model 9120A PRINTER, insert CONTINUE in the program wherever a 'PRINT' instruction will be required:

> PRINT/SPACE causes the printer to print any combination (selected on the printer) of the contents of the X, Y and Z registers;
>
> successive PRINT/SPACE instructions cause the printer to space after printing.

For the Model 9125A X-Y RECORDER, make provision for a sub-program to be written to control the recorder. In this case insert three (3) successive CONTINUE instructions in the program, then when required, these can be replaced by GO TO, ( ), ( ) and the sub-program can be inserted in any available part of the memory.

STEP SAVING

Following are more step-saving hints.

ENTERING A CONSTANT

If a constant is required during a program this can be entered as program steps.

STEP	KEY
22	ROLL ↑
23	2
24	•
25	1
26	3
27	

after step 26, 2.13 will be in the X register. If this number is stored as data, fourteen program steps (one register) are required; in this particular case ten steps have been saved.

Powers of 10 can be entered using the ENTER EXP key:

STEP	KEY
22	↑
23	ENTER EXP
24	4
25	

after step 24, 1 x 10^4 (10,000) will be in the X register; three steps have been saved.

MULTIPLY BY 2

To multiply by 2, use the plus (+) instruction:

STEP	KEY
34	π
35	↑
36	+
37	----

after step 36, 2π will be in the Y register; one step has been saved.

STEP SAVING

$\sqrt{\text{SUM OF TWO SQUARES}}$

To calculate the square root of the sum of two squares

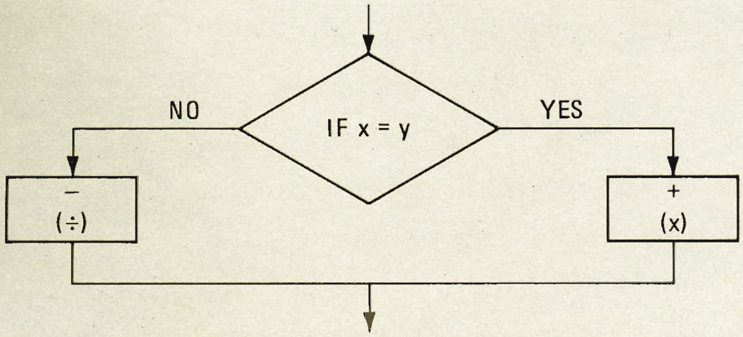
($\sqrt{A^2 + B^2}$):

	KEY
Enter A	( )
	↑
Enter B	( )
	TO POLAR

$\sqrt{A^2 + B^2}$ appears in the X register. (As TO POLAR is the operation which converts rectangular coordinates to polar coordinates, the angle (θ) in either degrees or radians, appears in the Y register.) Discounting the steps used to enter A and B, this calculation requires eight operations if the TO POLAR key is not used; six program steps have been saved by using TO POLAR.

CONDITIONAL OPERATIONS

To conditionally add or subtract (multiply or divide):



STEP	KEY
61	----
62	IF x = y
63	ACC+ (or) + (or) x
64	ACC+ (or) + (or) x
65	ACC− (or) − (or) ÷

if x = y, steps 63 through 65 result in a net addition (multiplication);

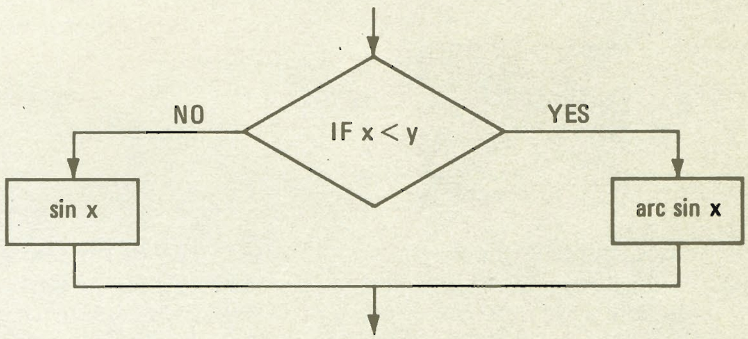
if x ≠ y, steps 63 and 64 are skipped and the subtraction (division) is executed.

STEP SAVING

CONDITIONAL OPERATIONS

CONTINUED

To conditionally calculate sin x or arc sin x ($\sin^{-1} x$):



STEP	KEY
75	IF $x < y$
76	CONTINUE
77	arc ▾
78	sin x
79	——

ADVANCED PROGRAMMING

In order to fit complex programs into the calculator memory it occasionally becomes necessary to revert to unusual programming techniques. Following are two such techniques; 'subroutines' and 'splitting a register'.

SUBROUTINES *

A subroutine may be defined as a sub-program which is to be stored only once in the calculator's memory but which may be used any number of times during the main program.

> **NOTE**
>
> A subroutine, as defined here, consists of program steps entered into the calculator by the programmer; this should not be confused with the subroutines which are an integral part of the ROM (see Page 7).

A subroutine may be 'called' at any desired point in the program; that is, the program branches to the subroutine's starting address, executes the operations of the subroutine and then returns to any specified point in the main program.

To illustrate this, here is a flow chart of the 'possible combinations' program (Page 97) as it might appear if a subroutine were to be used (to calculate N!) in place of the loop used in the original program.

The formula used is:

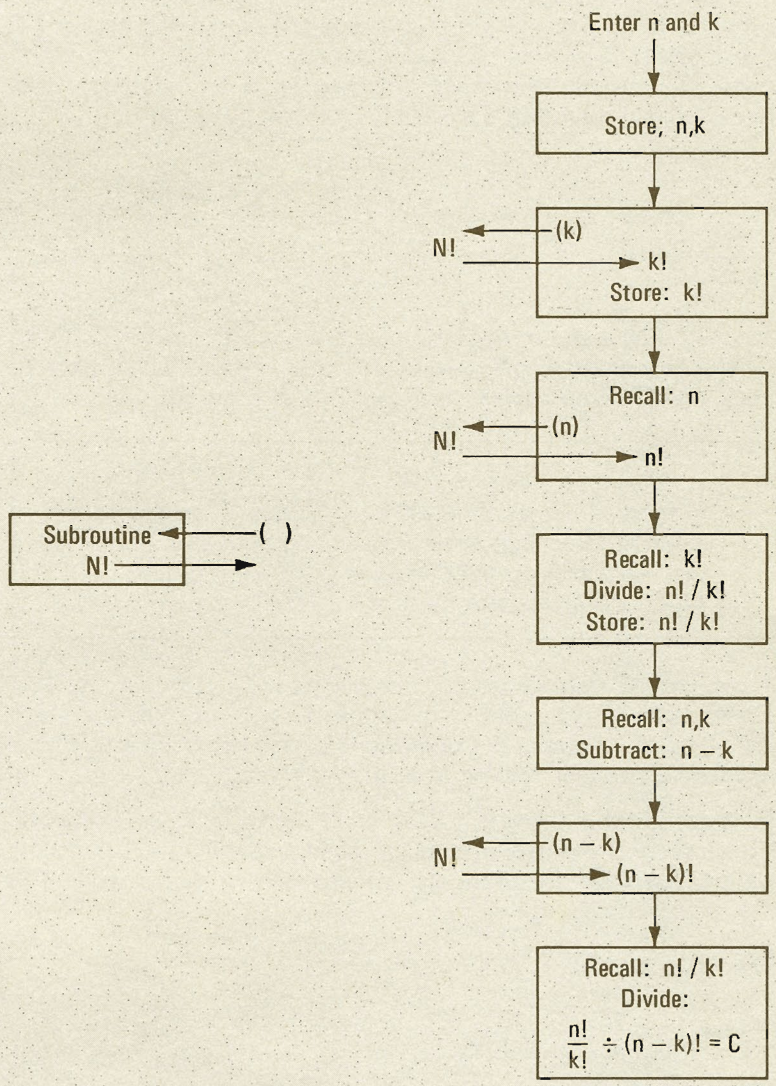
$$C_k^n = \frac{n!}{k!\,(n-k)!}$$

CONTINUED

* The programming technique described here is printed with the permission of Mr. W. W. Berg. Mr. Berg, who is an electrical engineer from Davenport, Iowa, originated the technique as applied to the 9100A Calculator and wrote the program steps which appear here.

ADVANCED PROGRAMMING

SUBROUTINES
CONTINUED



In order to use a subroutine, a return address must be generated each time the subroutine is called; that is, when the instructions of the subroutine have been executed, a 'GO TO' instruction, followed by an address, is required. The program steps shown on Page 109 generate that return address each time the subroutine is called. For convenience, these program steps will be referred to as the 'linkage'.

The linkage requires two complete registers; the 0 and 1 registers may be assigned. The GO TO instruction (referred to in the preceeding paragraph) is inserted in address 0d and the return address will be inserted in addresses 10 and 11 each time the subroutine is called. The steps of the subroutine may be located anywhere in the memory.

ADVANCED PROGRAMMING

In order to understand how the linkage generates the return address, it is first necessary to know how a data number is stored in a register. Data is stored as a 12-digit number (including the two guard digits) and a 2-digit exponent. Starting at character 0 in the register, the digits are stored in memory as follows:

1. the first 10 (displayed) digits of the number are stored in reversed order (the least significant digit in character 0, the most significant digit in character 9);
2. the exponent, least significant digit first, in characters *a* and *b*;
3. the guard digits, least significant digit first, in characters *c* and *d*.

For example, 1.234567890(12) x 10^{34} would be stored, say in register 1, as:

	characters													
	0	1	2	3	4	5	6	7	8	9	a	b	c	d
REGISTER 1	0	9	8	7	6	5	4	3	2	1	4	3	2	1
	number										exponent		guard digits	

> **NOTE**
>
> This can be easily verified by storing a number in a register and then stepping through the register, in PROGRAM mode, using the STEP PRGM key.

What the linkage now does (each time a subroutine is called) is to generate and store a data number in the 1 register so that the required return address appears in addresses 10 and 11. Then, with the 'GO TO' instruction in address 0*d*, the calculator assumes that the two digits in 10 and 11 are a branching address and it branches to that address.

Here are the program steps required for the linkage:

STEP	KEY
00	y⇄( )
01	1
02	ENTER EXP
03	CHG SIGN
04	9
05	×
06	1

STEP	KEY
07	+
08	y→( )
09	1
0a	ROLL ↓
0b	CLEAR x
0c	ROLL ↓
0d	GO TO ( )( )

ADVANCED PROGRAMMING

SUBROUTINES
CONTINUED

The instructions of the subroutine must be as follows (anywhere in the memory):

```
y⇄( )
1
•  }
•  } Body of Subroutine
•  } (any computations)
•  }
GO TO ( )( )
0
0
```

To call the subroutine, from anywhere in the main program, the following instructions must be used:

d_2

d_1

ROLL ↑

GO TO ( )( )

s_1

s_2

d_2 and d_1 are the second and first digits (respectively) of the return address; s_1 and s_2 are the first and second digits of the subroutine's starting address. d_1 and d_2 cannot be alphabetic, they must both be numeric, so the program must be arranged accordingly (insert CONTINUE in any unused program steps).

The subroutine itself must be so arranged that the final computed value appears in the Z register before the branch to address 00 is made. The value will then appear in the X register when the return is made to the main program, at address d_1d_2.

Using the same linkage, more than one subroutine can be called during a program; however, only one subroutine can be called at any one time during the program.

SPLITTING A REGISTER

'Splitting a register' allows two data numbers to be stored in one register, thus saving memory space for program steps. The technique is, however, effective only if there are many data numbers to be stored (the program steps used to 'split a register' may themselves require one, or more, registers).

In order to illustrate the 'splitting' technique, a simple case will be chosen. The form of the data may complicate the use of this technique; this will be discussed after the initial explanation has been given.

ADVANCED PROGRAMMING

Assume the following three conditions:

1. Six data-numbers, all two-digit integers, are to be stored (numbers 10 through 15 will be used as the data);
2. the data is to be stored in the a, b and c registers;
3. the program will then select the stored data from the c register as the next values for the computation. This requires that the y⇄( ) key be used to move the data through the memory (as in the 'possible combinations' program on Page 97) so that the next value to be recalled appears in the c register.

The data is paired before entry (10-11, 12-13, 14-15). As each pair is entered, the second number of the pair is converted to a decimal and added to the first number; the composite pair is then stored as one data number. The steps to do this can be either part of the program or part of the manual operations during digit entry.

(With 10 in the Y register and 11 in the X register) the required steps are:

```
↑
ENTER EXP
2
÷
↓
+
y⇄( )
a (or b.or c)
```

After all data has been entered and stored, the program recalls, in turn, each composite pair and reconverts the pair to the original two numbers.

```
y⇄( )
a (or b, or c)
↓
↑
int x
—
x⇄y
↑
ENTER EXP
2
×
```

the two numbers appear as originally entered, except that they now appear in the Z and Y registers, respectively.

SPLITTING A REGISTER
CONTINUED

Following is a flow chart illustrating how the technique might be used in a program. The flow chart assumes the three conditions previously established. Combining each data-pair is included as part of the program and a counting routine is included to determine when all data has been recalled for use. The y⇄( ) key is used to move the stored data through the memory so that the next values to be used are recalled from the c register.

As previously stated, the form of the data can introduce complications when this programming technique is used: the form of the data must be known and must be relatively uniform.

> **NOTE**
>
> It is because of these 'complications' that this technique has been avoided in the generalized programs of the Program Library.

1. In the preceeding example, the data consists of two-digit integers; if three-digit integers are to be included, then the 'exponent 2' must be changed to 'exponent 3', whereas if one-digit integers are included then no such change is necessary.
2. The second number in each pair can contain decimal digits whereas the first number cannot.
3. The first integer of each pair can contain more than two digits (without requiring 'exponent 2' to be changed) whereas the second integer cannot.
4. If the first number in any pair is to contain a decimal point, then some scaling routine must be used to convert it to an integer.

The following alternative method may also be used to combine and then separate each data pair:

1. Enter the smaller number of the pair in Y and the larger number (which must be an integer) in X.
2. To combine and store the data pair;

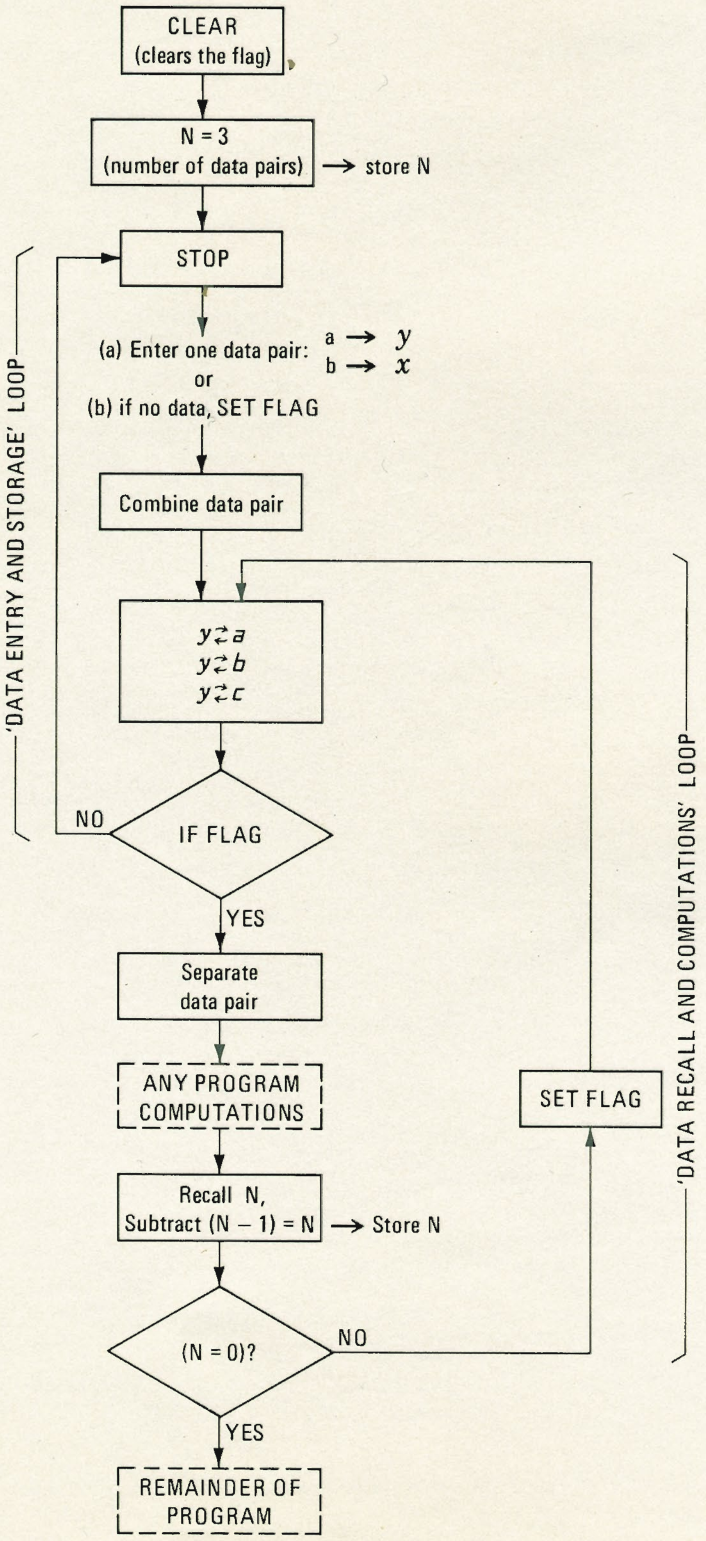
   ÷
   + (composite appears in Y)
   y⇄( )
   a

3. To recall and separate the data pair:

   a (composite appears in x)
   ↑
   int x
   —
   ×

the numbers appear, as originally entered, in the Y and X registers.

ADVANCED PROGRAMMING



APPENDIX

NOTE

a) One visible count is one count in the tenth digit.

b) "No. of decades of circles" refers to the number of integer circles divided by 10

I.	ADD, (SUBTRACT)		$\pm 5 \times 10^{-11} \times 10^{\text{(larger addend exponent)}}$
II.	MULTIPLY		±1/2 visible count
III.	DIVIDE		±1/2 visible count
IV.	SQUARE ROOT		±1/12 visible count
V.	COS θ		
		$\theta < 1$ rad	$\pm 4 \times 10^{-11}$
		$1 \text{ rad} \leq \theta < \pi$	$\pm 9 \times 10^{-11}$
		$\pi \leq \theta < 2\pi$	$\pm 28 \times 10^{-11}$
		$2\pi \leq \theta$	$\pm 18 \times 10^{-10} \times 10^{\text{(no. of decades of circles)}}$
VI.	SIN θ		
		$\theta < 1$ rad	±1 visible count
		$1 \text{ rad} \leq \theta < \pi$	$\pm 12 \times 10^{-11}$
		$\pi \leq \theta < 2\pi$	$\pm 31 \times 10^{-11}$
		$2\pi \leq \theta$	$\pm 18 \times 10^{-10} \times 10^{\text{(no. of decades of circles)}}$
VII.	TAN θ		
		$\theta < 1$ rad	±1 1/2 visible count
		$1 \text{ rad} \leq \theta < \pi$	±1/2 (exp of answer) $\pm 2 \times 10^{-10}$ (visible counts)
		$\pi \leq \theta < 2\pi$	±1 x exp of answer $\pm 3 \times 10^{-10}$ (visible counts)
		$2\pi \leq \theta$	±5 x exp of answer $\times 10^{\text{(no. of decades of circles)}}$ $\pm 31 \times 10^{-10} \times 10^{\text{(no. of decades of circles)}}$ visible counts
VIII.	POLAR TO RECT		same as cos θ and sin θ
IX.	TAN^{-1}(a)		
		radians	±1/10 visible count
		degrees	±1/4 visible count
X.	SIN^{-1} (a)		
		$a < .707$	±1/2 visible count
XI.	COS^{-1} (a)		
		$a < .707$	±1/2 visible count $\pm 10^{-11}$
XII.	SIN^{-1} (a)		
		$a > .707$	
		radians	$\pm 10^{-10}$
		degrees	$\pm 70 \times 10^{-10}$
XIII.	COS^{-1} (a)		
		$a > .707$	
		radians	$\pm 10^{-10}$
		degrees	$\pm 70 \times 10^{-10}$

APPENDIX

RANGE OF ARGUMENTS

Range of arguments for trigonometric, hyperbolic, logarithmic and exponential functions.

sin x, cos x, tan x;	$\|x\| \leq 1 \times 10^{10}$
$\sin^{-1}$ x;	$\|x\| \leq 1$
$\cos^{-1}$ x;	$\|x\| \leq 1$
$\tan^{-1}$ x;	any x
sinh x;	$\|x\| < 230.25$
cosh x;	$\|x\| < 230.25$
tanh x;	$\|x\| < 230.25$
$\sinh^{-1}$ x;	any x
$\cosh^{-1}$ x;	$\|x\| \geq 1$
$\tanh^{-1}$ x;	$\|x\| < 1$
ln x;	$x > 0$
log x;	$x > 0$
e^x;	$-227.95 < x < 230.25$

Range of the 9100A $9.999\ 999\ 999 \times 10^{99}$ to 1×10^{-98}

ERROR SUMMARY

The following is a summary of maximum calculator function errors. They are expressed as absolute errors unless relative error is noted.

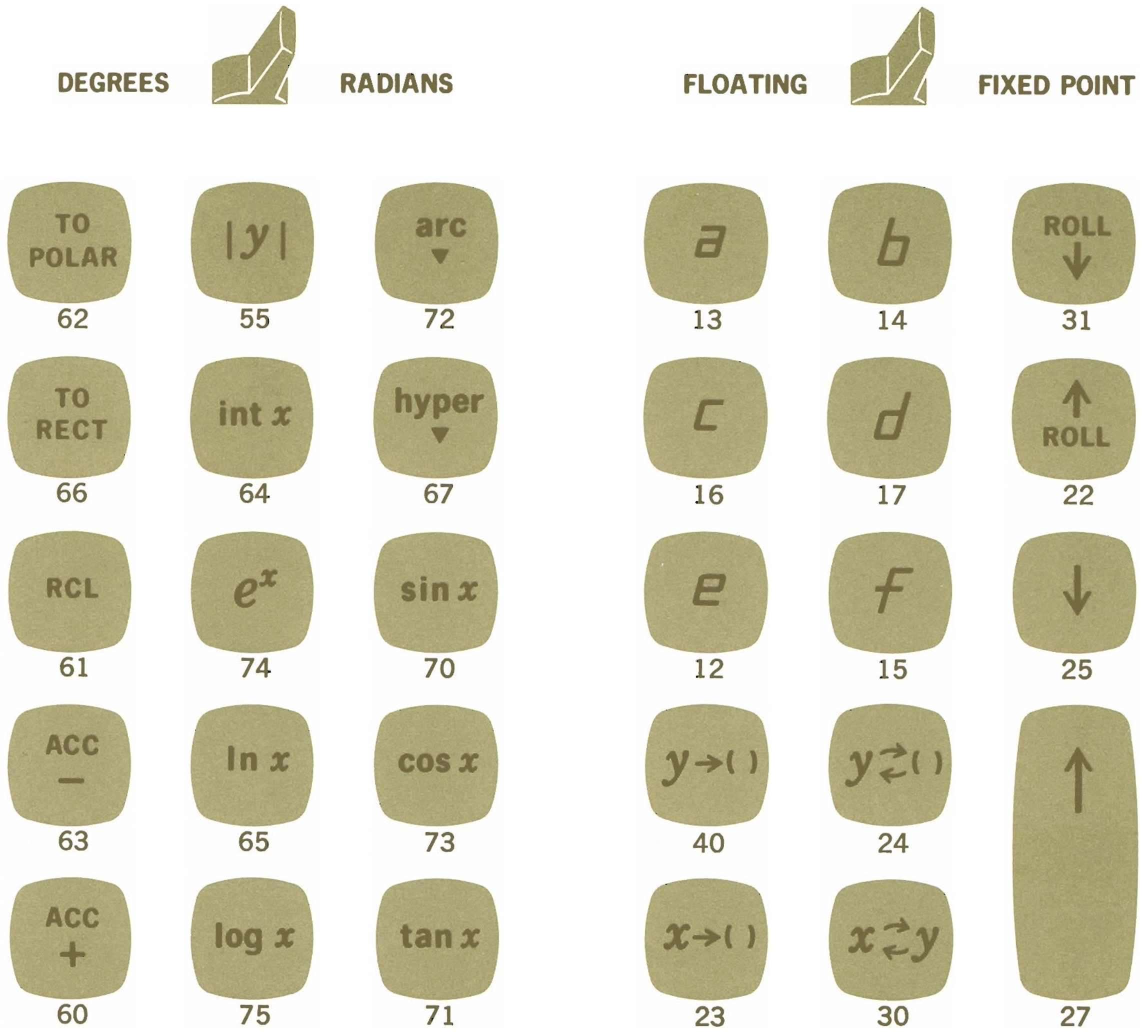
$$\text{Absolute error} = |f(X) - \hat{f}(X)|$$

$$\text{Relative error} = \frac{|f(X) - \hat{f}(X)|}{f(X)}$$

$f(X)$ = exact value

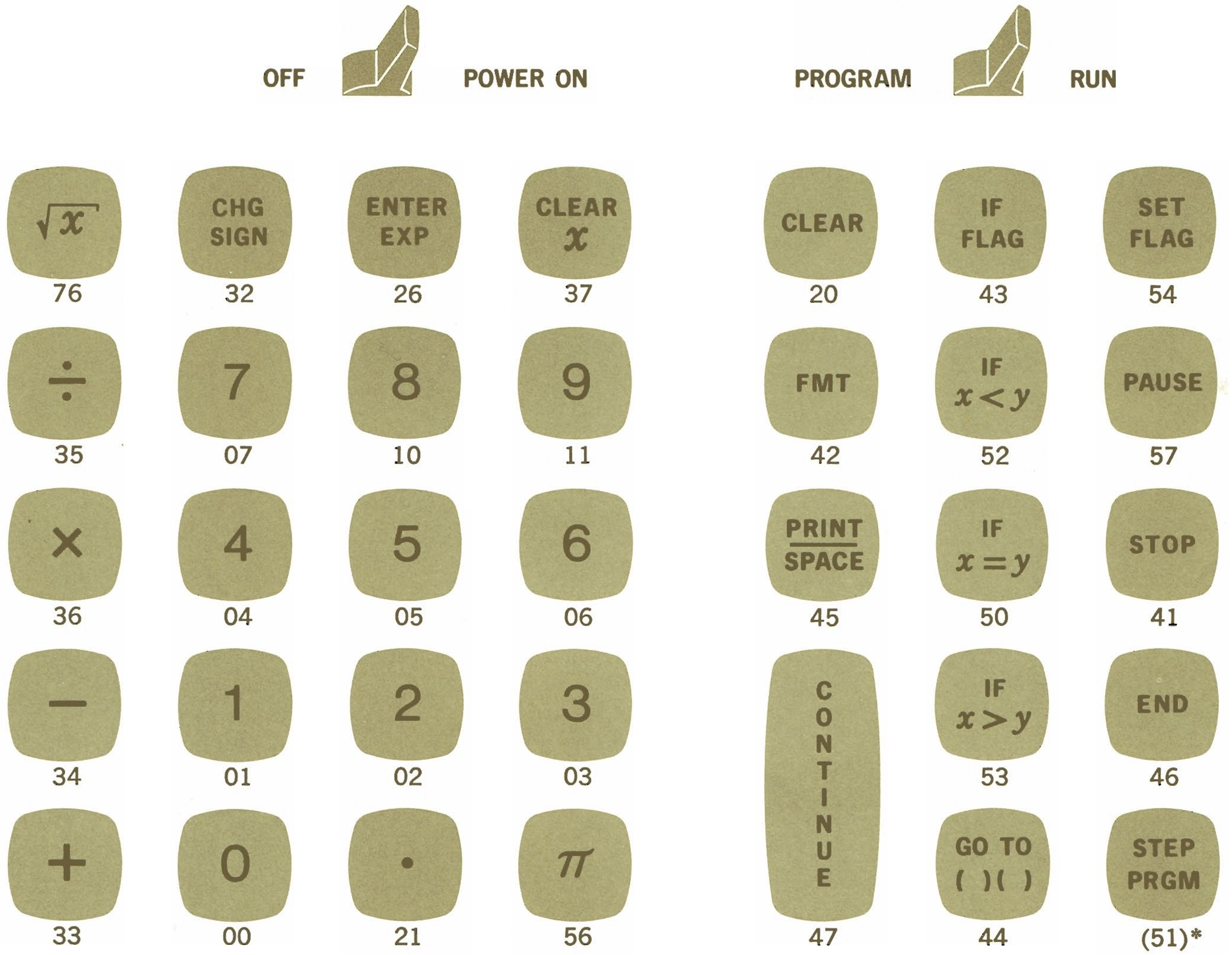
$\hat{f}(X)$ = calculated value

CONTINUED


DEGREES
RADIANS
FLOATING
FIXED POINT
TO POLAR
62
|y|
55
arc ▼
72
a
13
b
14
ROLL ↓
31
TO RECT
66
int x
64
hyper ▼
67
c
16
d
17
↑ ROLL
22
RCL
61
e^x
74
sin x
70
e
12
f
15
↓
25
ACC −
63
ln x
65
cos x
73
y→( )
40
y⇄( )
24
↑
27
ACC +
60
log x
75
tan x
71
x→( )
23
x⇄y
30

APPENDIX

KEY CODES



OFF — POWER ON

PROGRAM — RUN

$\sqrt{x}$ 76	CHG SIGN 32	ENTER EXP 26	CLEAR x 37	CLEAR 20	IF FLAG 43	SET FLAG 54
÷ 35	7 07	8 10	9 11	FMT 42	IF $x<y$ 52	PAUSE 57
× 36	4 04	5 05	6 06	PRINT SPACE 45	IF $x=y$ 50	STOP 41
− 34	1 01	2 02	3 03	CONTINUE 47	IF $x>y$ 53	END 46
+ 33	0 00	· 21	π 56		GO TO ( )( ) 44	STEP PRGM (51)*

The number shown below each key is the (octal) instruction code. These are also shown, in numerical order, on the pull-out card at the front of the calculator.

Refer to Page 58 for an explanation of the code.

*Not usable as a program operation

APPENDIX

Title ______________________________ Date __________ page ______

______________________________ Name ______________________________

Step	Key	Code	Display			Storage					
			x	y	z	f	e	d	c	b	a
0											
1											
2											
3											
4											
5											
6											
7											
8											
9											
a											
b											
c											
d											
0											
1											
2											
3											
4											
5											
6											
7											
8											
9											
a											
b											
c											
d											

PROGRAM PAD - LONG FORM
-hp- Part Number 09100-90003
(See Page 2)

APPENDIX

Step	Key	Code	Display			Step	Key	Code	Display			Step	Key	Code	Display		
			x	y	z				x	y	z				x	y	z
0						0						0					
1						1						1					
2						2						2					
3						3						3					
4						4						4					
5						5						5					
6						6						6					
7						7						7					
8						8						8					
9						9						9					
a						a						a					
b						b						b					
c						c						c					
d						d						d					
0						0						0					
1						1						1					
2						2						2					
3						3						3					
4						4						4					
5						5						5					
6						6						6					
7						7						7					
8						8						8					
9						9						9					
a						a						a					
b						b						b					
c						c						c					
d						d						d					
0						0											
1						1											
2						2											
3						3											
4						4											
5						5											
6						6											
7						7											
8						8											
9						9											
a						a											
b						b											
c						c											
d						d											

Storage	
f	
e	
d	
c	
b	
a	
9	
8	
7	
6	
5	
4	
3	
2	
1	
0	

PROGRAM PAD - SHORT FORM
-hp- Part Number 09100-90023
(See Page 2)

SALES & SERVICE OFFICES

UNITED STATES

ALABAMA
P.O. Box 4207
2003 Byrd Spring Road S.W.
Huntsville 35802
Tel: (205) 881-4591
TWX: 810-726-2204

ARIZONA
3009 North Scottsdale Road
Scottsdale 85251
Tel: (602) 945-7601
TWX: 910-950-1282

5737 East Broadway
Tucson 85716
Tel: (602) 298-2313
TWX: 910-952-1162

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2170

1101 Embarcadero Road
Palo Alto 94303
Tel: (415) 327-6500
TWX: 910-373-1280

2591 Carlsbad Avenue
Sacramento 95821
Tel: (916) 482-1463
TWX: 910-367-2092

1055 Shafter Street
San Diego 92106
Tel: (714) 223-8103
TWX: 910-335-2000

COLORADO
7965 East Prentice
Englewood 80110
Tel: (303) 771-3455
TWX: 910-935-0705

CONNECTICUT
508 Tolland Street
East Hartford 06108
Tel: (203) 289-9394
TWX: 710-425-3416

111 East Avenue
Norwalk 06851
Tel: (203) 853-1251
TWX: 710-468-3750

DELAWARE
3941 Kennett Pike
Wilmington 19807
Tel: (302) 655-6161
TWX: 510-666-2214

FLORIDA
P.O. Box 545
Suite 106
9999 N.E. 2nd Avenue
Miami Shores 33153
Tel: (305) 754-4565
TWX: 810-848-7262

P.O. Box 20007
Herndon Station 32814
621 Commonwealth Avenue
Orlando
Tel: (305) 841-3970
TWX: 810-850-0113

P.O. Box 8128
Madeira Beach 33708
410 150th Avenue
St. Petersburg
Tel: (813) 391-0211
TWX: 810-863-0366

GEORGIA
P.O. Box 28234
450 Interstate North
Atlanta 30328
Tel: (404) 436-6181
TWX: 810-766-4890

ILLINOIS
5500 Howard Street
Skokie 60076
Tel: (312) 677-0400
TWX: 910-223-3613

INDIANA
3839 Meadows Drive
Indianapolis 46205
Tel: (317) 546-4891
TWX: 810-341-3263

LOUISIANA
P.O. Box 856
1942 Williams Boulevard
Kenner 70062
Tel: (504) 721-6201
TWX: 810-955-5524

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-0850

P.O. Box 1648
2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
24315 Northwestern Highway
Southfield 48075
Tel: (313) 353-9100
TWX: 810-232-1532

MINNESOTA
2459 University Avenue
St. Paul 55114
Tel: (612) 645-9461
TWX: 910-563-3734

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

2812 South Brentwood Blvd.
St. Louis 63144
Tel: (314) 962-5000
TWX: 910-760-1670

NEW JERSEY
W. 120 Century Road
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

1060 N. Kings Highway
Cherry Hill 08034
Tel: (609) 667-4000
TWX: 710-892-4945

NEW MEXICO
P.O. Box 8366
Station C
6501 Lomas Boulevard N.E.
Albuquerque 87108
Tel: (505) 255-5586
TWX: 910-989-1665

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2485
TWX: 910-983-0550

NEW YORK
1702 Central Avenue
Albany 12205
Tel: (518) 869-8462
TWX: 710-441-8270

1219 Campville Road
Endicott 13760
Tel: (607) 754-0050
TWX: 510-252-0890

82 Washington Street
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-248-0012

39 Saginaw Drive
Rochester 14623
Tel: (716) 473-9500
TWX: 510-253-5981

1025 Northern Boulevard
Roslyn, Long Island 11576
Tel: (516) 869-8400
TWX: 510-223-0811

5858 East Molloy Road
Syracuse 13211
Tel: (315) 454-2486
TWX: 710-541-0482

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101
TWX: 510-926-1516

OHIO
25575 Center Ridge Road
Cleveland 44145
Tel: (216) 835-0300
TWX: 810-427-9129

3460 South Dixie Drive
Dayton 45439
Tel: (513) 298-0351
TWX: 810-459-1925

OKLAHOMA
2919 United Founders Boule:ard
Oklahoma City 73112
Tel: (405) 848-2801
TWX: 910-830-6862

OREGON
Westhills Mall, Suite 158
4475 S.W. Scholls Ferry Road
Portland 97225
Tel: (503) 292-9171
TWX: 910-464-6103

PENNSYLVANIA
2500 Moss Side Boulevard
Monroeville 15146
Tel: (412) 271-0724
TWX: 710-797-3650

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

RHODE ISLAND
873 Waterman Ave.
East Providence 02914
Tel: (401) 434-5535
TWX: 710-381-7573

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101
TWX: 910-867-4723

6300 Westpark Drive
Houston 77027
Tel: (713) 781-6000
TWX: 910-881-2645

231 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-4171
TWX: 910-871-1170

UTAH
2890 South Main Street
Salt Lake City 84115
Tel: (801) 487-0715
TWX: 910-925-5681

VERMONT (Feb. 70)
P.O. Box 2287
Kennedy Drive
South Burlington 05401
Tel: (802) 658-4455
TWX: 710-658-4712

VIRGINIA
P.O. Box 6514
2111 Spencer Road
Richmond 23230
Tel: (703) 282-5451
TWX: 710-956-0157

WASHINGTON
433-108th N.E.
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2303

*WEST VIRGINIA
Charleston
Tel: (304) 768-1232

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia . . . North Hollywood, California . . . Paramus, New Jersey . . . Skokie, Illinois. Their complete addresses are listed above.

*Service Only

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11745 Jasper Ave.
Edmonton
Tel: (403) 482-5561
TWX: 610-831-2431

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
1037 West Broadway
Vancouver 12
Tel: (604) 731-5301
TWX: 610-922-5059

MANITOBA
Hewlett-Packard (Canada) Ltd.
511 Bradford Ct.
St. James
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
2745 Dutch Village Rd.
Suite 203
Halifax
Tel: (902) 455-0511
TWX: 610-271-4482

ONTARIO
Hewlett-Packard (Canada) Ltd.
880 Lady Ellen Place
Ottawa 3
Tel: (613) 722-4223
TWX: 610-562-1952

Hewlett-Packard (Canada) Ltd.
50 Galaxy Blvd.
Rexdale
Tel: (416) 677-9611
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd.
275 Hymus Boulevard
Pointe Claire
Tel: (514) 697-4232
TWX: 610-422-3022
Telex: 01-20607

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada) Ltd. in Pointe Claire, at the complete address listed above.

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina
S.A.C.e.I
Lavalle 1171 - 3°
Buenos Aires
Tel: 35-0436, 35-0627, 35-0431
Telex: 012-1009
Cable: HEWPACKARG

BRAZIL
Hewlett-Packard Do Brasil
I.e.C Ltda.
Rua Coronel: Oscar Porto, 691
Sao Paulo - 8, SP
Tel: 71-1503
Cable: HEWPACK Sao Paulo

Hewlett-Packard Do Brasil
I.e.C. Ltda.
Avenida Franklin Roosevelt 84-
grupo 203
Rio de Janeiro, ZC-39, GB
Tel: 232-9733
Cable: HEWPACK Rio de Janeiro

CHILE
Héctor Calcagni y Cia, Ltda.
Bustos, 1932-3er Piso
Casilla 13942
Santiago
Tel: 4-2396
Cable: Calcagni Santiago

COLOMBIA
Instrumentacion
Henrik A. Langebaek & Kier
Ltda.
Carrera 7 No. 48-59
Apartado Aereo 6287
Bogota, 1 D.E.
Tel: 45-78-06, 45-55-46
Cable: AARIS Bogota
Telex: 044-400

COSTA RICA
Lic. Alfredo Gallegos Gurdián
Apartado 3243
San José
Tel: 21-86-13
Cable: GALGUR San José

ECUADOR
Laboratorios de Radio-Ingenieria
Calle Guayaquil 1246
Post Office Box 3199
Quito
Tel: 12496
Cable: HORVATH Quito

EL SALVADOR
Electrónica
Apartado Postal 1589
27 Avenida Norte 1133
San Salvador
Tel: 25-74-50
Cable: ELECTRONICA
San Salvador

GUATEMALA
Olander Associates Latin America
Apartado Postal 1226
Ruta 4, 6-53, Zona 4
Guatemala City
Tel: 63958
Cable: OLALA Guatemala City

JAMAICA
General Engineering Services,
Ltd.
27 Dunrobin Ave.
Kingston
Tel: 42657
Cable: GENSERV

MEXICO
Hewlett-Packard Mexicana, S.A.
de C.V.
Moras 439
Col. del Valle
Mexico 12, D.F.
Tel: 5-75-46-49

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 3451, 3452
Cable: ROTERAN Managua

PANAMA
Electrónica Balboa, S.A.
P.O. Box 4929
Ave. Manuel Espinosa No. 13-50
Bldg. Alina
Panama City
Tel: 30833
Cable: ELECTRON Panama City

PERU
Fernando Ezeta B.
Avenida Petit Thouars 4719
Miraflores
Casilla 3061
Lima
Tel: 45-2335
Cable: FEPERU Lima

PUERTO RICO
San Juan Electronics, Inc.
P.O. Box 5167
Ponce de Leon 154
Pda. 3-Pta. de Tierra
San Juan 00906
Tel: (809) 725-3342
Cable: SATRONICS San Juan
Telex: SATRON 3450 332

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA
Hewlett-Packard De Venezuela
C.A.
Apartado 50933
Caracas
Tel: 71.88.05, 71.88.69, 71.99.30
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
INTERCONTINENTAL
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 326-7000
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8461

EUROPE

AUSTRIA
Unilabor GmbH
Wissenschaftliche Instrumente
Rummelhardtgasse 6/3
P.O. Box 33
Vienna A-1095
Tel: 42 61 81
Cable: LABORINSTRUMENT
Vienna
Telex: 75 762

BELGIUM
Hewlett-Packard Benelux S.A.
348 Boulevard du Souverain
Brussels 16
Tel: 72 22 40
Cable: PALOBEN Brussels
Telex: 23 494

DENMARK
Hewlett-Packard A/S
Langebjerg 6
2850 **Naerum**
Tel: (01) 80 40 40
Cable: HEWPACK AS
Telex: 66 40

FINLAND
Hewlett-Packard Oy
Gyldenintie 3
Helsinki 20
Tel: 67 35 38
Cable: HEWPACKOY-Helsinki
Telex: 12-1563

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
91 Orsay
Tel: 920 88 01
Cable: HEWPACK Orsay
Telex: 60048

Hewlett-Packard France
4 Quai des Etroits
69 **Lyon** 5ème
Tel: 42 63 45
Cable: HEWPACK Lyon
Telex: 31617

GERMANY
Hewlett-Packard Vertriebs-GmbH
Lietzenburgerstrasse 30
1 **Berlin** W 30
Tel: 24 60 65/66
Telex: 18 34 05

Hewlett-Packard Vertriebs-GmbH
Herrenbergerstrasse 110
703 **Böblingen,** Württemberg
Tel: 07031-6671
Cable: HEPAG Böblingen

Telex: 72 65 739
Hewlett-Packard Vertriebs-GmbH
Achenbachstrasse 15
4 **Düsseldorf** 1
Tel: 68 52 58/59
Telex: 85 86 533

Hewlett-Packard Vertriebs-GmbH
Berliner Strasse 117
6 **Nieder-Eschbach/Frankfurt 56**
Tel: (0611) 50 10 64
Cable: HEWPACKSA Frankfurt
Telex: 41 32 49

Hewlett-Packard Vertriebs-GmbH
Beim Strohhause 26
2 **Hamburg** 1
Tel: 24 05 51/52
Cable: HEWPACKSA Hamburg
Telex: 21 53 32

Hewlett-Packard Vertriebs-GmbH
Reginfriedstrasse 13
8 **München** 9
Tel: 0811 69 59 71/75
Cable: HEWPACKSA München
Telex: 52 49 85

GREECE
Kostas Karayannis
18, Ermou Street
Athens 126
Tel: 230303 232947
Cable: RAKAR Athens
Telex: 21 59 62 RKAR GR

IRELAND
Hewlett-Packard Ltd.
224 Bath Road
Slough, Bucks, England
Tel: Slough 753-33341
Cable: HEWPIE Slough
Telex: 84413

ITALY
Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci 2
20124 **Milano**
Tel: 6251 (10 lines)
Cable: HEWPACKIT Milan
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Palazzo Italia
Piazza Marconi 25
00144 **Rome** - Eur
Tel: 591 2544
Cable: HEWPACKIT Rome
Telex: 61514

NETHERLANDS
Hewlett-Packard Benelux, N.V.
Weerdestein 117
Amsterdam, Z 11
Tel: 42 77 77
Cable: PALOBEN Amsterdam
Telex: 13 216

NORWAY
Hewlett-Packard Norge A/S
Box 149
Nesveien 13
Haslum
Tel: 53 83 60
Cable: HEWPACK Oslo
Telex: 6621

PORTUGAL
Telectra
Rua Rodrigo da Fonseca 103
P.O. Box 2531
Lisbon 1
Tel: 68 60 72
Cable: TELECTRA Lisbon
Telex: 1598

SPAIN
Ataio Ingenieros
Ganduxer 76
Barcelona 6
Tel: 211-44-66

Ataio Ingenieros
Enrique Larreta 12
Madrid, 16
Tel: 235 43 44
Cable: TELEATAIO Madrid
Telex: 2 72 49

SWEDEN
Hewlett-Packard (Sverige) AB
Hagakersgatan 9C
S 431 04 **Mölndal** 4
Tel: 031 - 27 68 00

Hewlett-Packard (Sverige) AB
Svetsarvägen 7
S171 20 **Solna** 1
Tel: (08) 98 12 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

SWITZERLAND
Hewlett Packard (Schweiz) AG
Zurcherstrasse 20
8952 Schlieren
Zurich
Tel: (051) 98 18 21/24
Cable: HEWPACKAG Zurich
Telex: 53933

Hewlett Packard (Schweiz) A.G.
Rue du Bois-du-Lan 7
1217 Meyrin-**Geneva**
Tel: (022) 41 54 00
Cable: HEWPACKSA Geneva
Telex: 2 24 86

TURKEY
Telekom Engineering Bureau
P.O. Box 376 - Galata
Istanbul
Tel: 49 40 40
Cable: TELEMATION Istanbul

UNITED KINGDOM
Hewlett-Packard Ltd.
224 Bath Road
Slough, Bucks
Tel: Slough 33341
Cable: HEWPIE Slough
Telex: 84413

YUGOSLAVIA
Belram S.A.
83 avenue des Mimosas
Brussels 15, Belgium
Tel: 34 33 32, 34 26 19
Cable: BELRAMEL Brussels
Telex: 21790

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard S.A.
Rue du Bois-du-Lan 7
1217 Meyrin-**Geneva**
Tel: (022) 41 54 00
Cable: HEWPACKSA Geneva
Telex: 2.24.86

AFRICA, ASIA, AUSTRALIA

ANGOLA
Telectra Empresa Técnica
de Equipamentos Eléctricos
SAR
Rua de Barbosa Rodrigues
42-1°
Box 6487
Luanda
Cable: TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia
Pty. Ltd.
22-26 Weir Street
Glen Iris, 3146
Victoria
Tel: 20.1371 (4 lines)
Cable: HEWPARD Melbourne
Telex: 31024

Hewlett-Packard Australia
Pty. Ltd.
61 Alexander Street
Crows Nest 2065
New South Wales
Tel: 43.7866
Cable: HEWPARD Sydney
Telex: 21561

Hewlett-Packard Australia
Pty. Ltd.
97 Churchill Road
Prospect 5082
South Australia
Tel: 65.2366
Cable: HEWPARD Adelaide

Hewlett Packard Australia
Pty. Ltd.
2nd Floor, Suite 13
Casablanca Buildings
196 Adelaide Terrace
Perth, W.A. 6000
Tel: 21-3330

CEYLON
United Electricals Ltd.
P.O. Box 681
Yahala Building
Staples Street
Colombo 2
Tel: 5496
Cable: HOTPOINT Colombo

CYPRUS
Kypronics
19-19D Hommer Avenue
P.O. Box 752
Nicosia
Tel: 6282-75628
Cable: HE-I-NAMI

ETHIOPIA
African Salespower & Agency
Private Ltd., Co.
P. O. Box 718
58/59 Cunningham St.
Addis Ababa
Tel: 12285
Cable: ASACO Addisababa

HONG KONG
Schmidt & Co. (Hong Kong) Ltd.
P.O. Box 297
1511, Prince's Building
10, Chater Road
Hong Kong
Tel: 240168, 232735
Cable: SCHMIDTCO Hong Kong

INDIA
The Scientific Instrument
Co., Ltd.
6, Tej Bahadur Sapru Road
Allahabad 1
Tel: 2451
Cable: SICO Allahbad

The Scientific Instrument
Co., Ltd.
12/5 Dickenson Road
Bangalore -1
Cable: SICO Bangalore

The Scientific Instrument
Co., Ltd.
240, Dr. Dadabhai Naoroji Road
Bombay 1
Tel: 26-2642
Cable: SICO Bombay

The Scientific Instrument
Co., Ltd.
11, Esplanade East
Calcutta 1
Tel: 23-4129
Cable: SICO Calcutta

The Scientific Instrument
Co., Ltd.
30, Mount Road
Madras 2
Tel: 86339
Cable: SICO Madras

The Scientific Instrument
Co., Ltd.
17, Kamalnayan Society
P.O. Navajiwan
Ahmedabad 14
Cable: SICO, Ahmedabad

The Scientific Instrument Co. Ltd.
5-8-525 Mahatma Gandhi Road
Hyderabad-1 (A-P) India
Cable: SICO Hyderabad

The Scientific Instrument Co., Ld.
B-7, Ajmeri Gate Extn.
New Delhi 1
Tel: 27-1053
Cable: SICO New Delhi

INDONESIA
Bah Bolon Trading Coy. N.V.
Djalah Merdeka 29
Bandung
Tel: 4915 51560
Cable: ILMU
Telex: 809

IRAN
Telecom, Ltd.
P. O. Box 1812
240 Kh. Saba Shomali
Teheran
Tel: 43850, 48111
Cable: BASCOM Teheran

ISRAEL
Electronics & Engineering
Div. of Motorola Israel Ltd.
17 Aminadav Street
Tel-Aviv
Tel: 36941 (3 lines)
Cable: BASTEL Tel-Aviv
Telex: Bastel Tv 033-569

JAPAN
Yokogawa-Hewlett-Packard Ltd.
Nisei Ibaragi Bldg.
2-2-8 Kasuga
Ibaragi-Shi
Osaka
Tel: 23-1641

Yokogawa-Hewlett-Packard Ltd.
Ito Building
No. 59, Kotori-cho
Nakamura-ku, **Nagoya** City
Tel: 551-0215

Yokogawa-Hewlett-Packard Ltd.
Ohashi Building
59 Yoyogi 1-chrome
Shibuya-ku, **Tokyo**
Tel: 370-2281/7
Telex: 232-2024YHP
Cable: YHPMARKET TOK 23-724

KENYA
R. J. Tilbury Ltd.
P. O. Box 2754
Suite 517/518
Hotel Ambassadeur
Nairobi
Tel: 25670, 68206, 58196
Cable: ARJAYTEE Nairobi

KOREA
American Trading Co., Korea, Ltd.
P.O. Box 1103
Dae Kyung Bldg.
107 Sejong Ro
Chongro Ku
Seoul
Tel: 75-5841 (4 lines)
Cable: AMTRACO Seoul

LEBANON
Constantin E. Macridis
Clemenceau Street
P.O. Box 7213
Beirut
Tel: 220846
Cable: ELECTRONUCLEAR Beirut

MALAYSIA
MECOMB Malaysia Ltd.
2 Lorong 13/6A
Section 13
Petaling Jaya, **Selangor**
Cable: MECOMB Kuala Lumpur

MOZAMBIQUE
A. N. Goncalves, LDA.
4.1 Apt. 14 Av. D. Luis
P.O. Box 107
Lourenco Marques
Cable: NEGON

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
32-34 Kent Terrace
P.O. Box 9443
Wellington, N.Z.
Tel: 59-559
Cable: HEWPACK Wellington

PAKISTAN (EAST)
Mushko & Company, Ltd.
Zirat Chambers
31, Jinnah Avenue
Dacca
Tel: 280058
Cable: NEWDEAL Dacca

PAKISTAN (WEST)
Mushko & Company, Ltd.
Oosman Chambers
Victoria Road
Karachi 3
Tel: 511027, 512927
Cable: COOPERATOR Karachi

PHILIPPINES
Electromex Inc.
2129 Pasong Tamo
Makati, Rizal
P.O. Box 4326
Manila
Tel: 88-91-71 or 88-83-76
Cable: ELEMEX Manila

SINGAPORE
Mechanical and Combustion
Engineering Company Ltd.
9, Jalan Kilang
Singapore, 3
Tel: 642361-3
Cable: MECOMB Singapore

SOUTH AFRICA
Hewlett Packard South Africa
(Pty.), Ltd.
Hill House
43 Somerset Rd.
Cape Town
Tel: 3-6019
Cable: HEWPACK Cape Town
Telex: 7038CT

Hewlett Packard South Africa
(Pty.), Ltd.
P.O. Box 31716
30 De Beer Street
Braamfontein, **Johannesburg**
Tel: 724-4172 724-4195
Telex: 0226 JH
Cable: HEWPACK Johannesburg

TAIWAN REP. OF CHINA
Hwa Sheng Electronic Co., Ltd.
P. O. Box 1558
Room 404
Chia Hsin Building
No. 96 Chung Shan
North Road, Sec. 2
Taipei
Tel: 555211 Ext. 532-539
545936, 546076, 548661
Cable: VICTRONIX Taipei

TANZANIA
R. J. Tilbury Ltd.
P.O. Box 2754
Suite 517/518
Hotel Ambassadeur
Nairobi
Tel: 25670, 26803, 68206, 58196
Cable: ARJAYTEE Nairobi

THAILAND
The International
Engineering Co., Ltd.
P. O. Box 39
614 Sukhumvit Road
Bangkok
Tel: 910722 (7 lines)
Telex: INTENCO BK 226
Cable: CYSOM BANGKOK

UGANDA
R. J. Tilbury Ltd.
P.O. Box 2754
Suite 517/518
Hotel Ambassadeur
Nairobi
Tel: 25670, 26803, 68206, 58196
Cable: ARJAYTEE Nairobi

VIETNAM
Peninsular Trading Inc.
P.O. Box H-3
216 Hien-Vuong
Saigon
Tel: 20.805
Cable: PENINSULA Saigon

ZAMBIA
R. J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Zambia, Central Africa

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
INTERCONTINENTAL
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 326-7000
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8461

PART NO. 09100–90001
DEC 1969

PRINTED IN U.S.A.