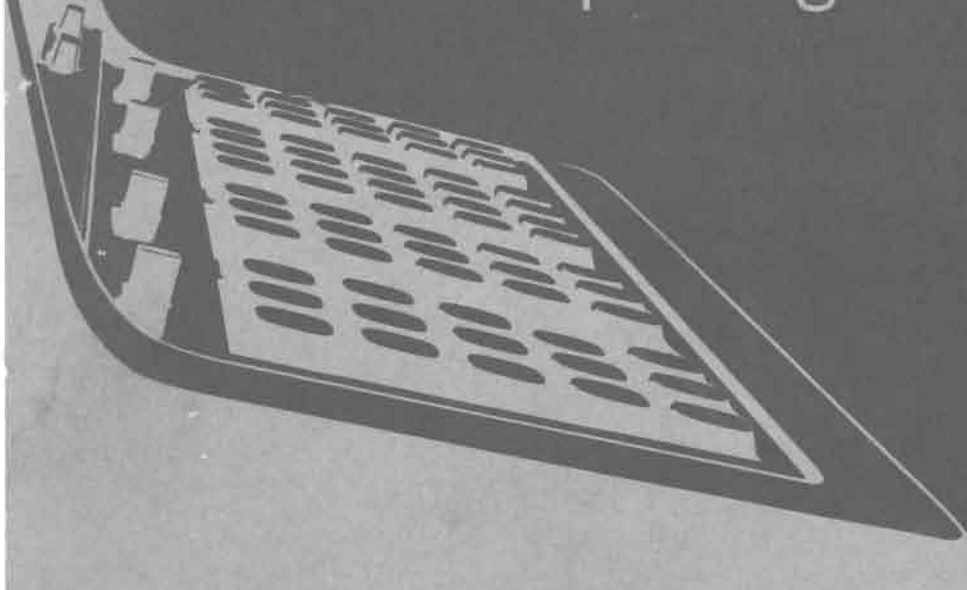
Nuclear Development Division

HEWLETT-PACKARD CALCULATOR

Model 9100B

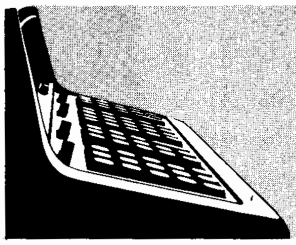
Operating and Programming



HEWLETT hp PACKARD

WARRANTY AND ASSISTANCE

The Hewlett-Packard 9100B Calculator is warranted against defects in materials 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 at the back of this manual.



MANUAL CHANGES

MODEL 9100B

OPERATING AND PROGRAMMING MANUAL
Manual Part No. 09100-90021

Instrument Serial Number	Make Manual Changes	Instrument Serial Number	Make Manual Changes
ALL	ERRATA		

Page 2. Add the following note to Table 1:

Calculators manufactured after February 1970 have an auxiliary power output connector on the rear panel. These calculators use power cord (HP Part Number) 8120-1378.

All earlier calculators, which do not have the auxiliary output, use power cord (HP Part Number) 8120-0078.

Page 77. Add to the NOTE.

Occasionally, when the PROGRAM-RUN switch is set to PROGRAM an unusual display may appear in the Z register. This may consist of a random number between the address and the octal code; alternatively, an improper address or octal code, consisting of more than two digits, may appear.

To clear this, switch to RUN and press the SUB/RETURN key; then readdress the program counter to the required address and switch back to PROGRAM.

Page 85. Insert after EXAMPLE 4.

The four conditional keys are, specifically, intended to be followed by an address for branching: therefore, in a few cases, unexpected results are obtained when operational keys are used after the 'IF' keys.

Applicable to all four IF keys when the condition is met (YES).

1.	IF x = y CHG SIGN	Regardless of the steps preceding the IF, changes the sign of the exponent of the number in X. The sign of the number remains unchanged.

Page 85. Continued.

2.	IF x = y	If the program steps are in the (+) page:
	•	$.1 \longrightarrow X$
	1	If the program steps are in the (-) page:
		$-.1 \longrightarrow X$
3.	IF x = y	The contents of (+) f are recalled to X but the
	RCL	Y register remains unchanged; also the program
		counter branches to a random address.

Applicable only to the IF FLAG key when the flag is not set (NO).

ENTER EXP	The 'not met' IF FLAG does not reset the
6	ENTER EXP key; this results in the alphameric
IF FLAG	character being entered as a digit of the
3	exponent. If RCL is used in place of the
4	alphameric then the contents of f are
	recalled to X but nonsense appears in the Y
any alphameric	register.

OPERATING AND PROGRAMMING

HEWLETT-PACKARD 9100B

PREFACE

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 iv 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.

FROM 9100A TO 9100B

For the benefit of 9100B users who have previously used and programmed only the 9100A Calculator, here is a brief summary of the 9100B features and characteristics which differ from those of the 9100A.

1. The 9100B has a larger programmable memory which is divided into two pages, the (+) page and the (–) page (see Pages 8 and 57).
   The (+) page consists of 16 registers designated (+) 0 through (+) *f*; these are identical to the 0 through *f* registers of the 9100A.
   The (–) page consists of 16 similar registers, (–) 0 through (–) *f* .
   The sign of the page (+ or –) is required for addressing the memory and for branching between pages; the sign is not required if the branch is to an address on the same page as the branch instruction (see 'GO TO' key on Page 68).

PREFACE

2. The 9100B has subroutine capability which is controlled by the 'SUB/RETURN' instruction (see Pages 60 and 94).

3. An extra recall instruction, 'to x from' ($x \leftarrow (\,)$), is available to recall data from the (+) numeric and (–) alphameric registers (see Page 38).

4. In the 'RUN' mode, 'STEP PRGM' no longer causes an immediate branch except when a 'not met' 'IF' qualifer is encountered (see 'EDITING A PROGRAM' on Page 76). In the 'PROGRAM' mode, as 'STEP PRGM' is pressed, the X register displays the address and operation (as before) and the Z register displays the next address and operation.

5. Conditional branching remains as in the 9100A except that the plus (+) and minus (–) signs cannot follow an 'IF' instruction unless they are part of an address. 'CONT(inue)' cannot be used as a 'no operation' after an 'IF' instruction if 'CONT' is to be followed by an alphameric or sign (+ or –) instruction.

6. When storing or recalling data, the sign of the page is always required if the storage register is on the (–) page; the sign of the page is never required if the storage register is on the (+) page. This is true regardless of which page contains the 'store' or 'recall' instructions.

7. Operation of the 'ACC+', 'ACC–' and 'RCL' keys is exactly as in the 9100A in that only the (+)e and (+)f registers are affected. 'CLEAR' clears the (+)e , (+)f , X, Y and Z registers and the 'ARC' and 'HYPER' conditions, as in the 9100A.

8. 'END' is **not** a required last instruction in a program which is to be recorded on a magnetic card (see 'END' on Page 70).

PAGE INDEX OF KEYS AND SWITCHES

16,21

DEGREES	19,44	RADIANS	FLOATING	19	FIXED POINT

TO POLAR	$\lvert y \rvert$	arc ▼	a	b	$x \rightleftarrows y$
51	50	45			30
TO RECT	int x	hyper ▼	c	d	ROLL ↓
52	49	45			30
RCL	e^x	sin x	e	f	↑ ROLL
42,53	47	44	41	41	29
ACC −	ln x	cos x	$y \rightarrow (\ )$	$y \rightleftarrows (\ )$	↓
53	46	44	40	40	29
ACC +	log x	tan x	$x \rightarrow (\ )$	$x \leftarrow (\ )$	↑
53	48	44	38	38	29

RECORD		**ENTER**
75		76

OFF	**POWER ON**	**PROGRAM**	**RUN**
19		19,68	

$\sqrt{x}$	**CHG SIGN**	**ENTER EXP**	**CLEAR** x	**CLEAR**	**IF FLAG**	**SET FLAG**	**DECIMAL DIGITS**
46	27	25	22	24	88	88	
$\div$	7	8	9	**FMT**	**IF** $x<y$	**PAUSE**	
33				71	83	71	
$\times$	4	5	6	**PRINT / SPACE**	**IF** $x=y$	**STOP**	
32				71	83	69	20
$-$	1	2	3	**▲SUB▼ / RETURN**	**IF** $x>y$	**END**	
32				94	83	70	
$+$	0	•	π	**CONT**	**GO TO ( )( )**	**STEP PRGM**	
31	22,41	28	50	69	68	70	

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

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

GENERAL INFORMATION

This section contains general information about the Model 9100B, 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 9100B 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 business, educational, scientific and engineering problems. Trigonometric, logarithmic and mathematical functions are performed with single key strokes.

The 9100B is also 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) and the use of subroutines provides complete programming capability. Knowledge of special 'machine language' is not required in order to program your calculator.

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, below the keyboard; the operation of each key is briefly explained on this card.

CAUTION

PLEASE DO NOT APPLY OPERATING POWER TO YOUR 9100B CALCULATOR UNTIL YOU HAVE READ THE TURN-ON INSTRUCTIONS ON PAGE 4 OF THIS MANUAL.

GENERAL INFORMATION

ACCESSORIES EQUIPMENT SUPPLIED

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

TABLE 1

ACCESSORIES/EQUIPMENT SUPPLIED

PART NO.	QUANTITY	DESCRIPTION
09100-90021	2	Operating and Programming Manual
09100-90022	1	Program Library
09100-90023	1	Program Pad
09100-90024	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 on the rear cover of the Program Library.

A box of five program pads (Part No. 09100-90020) and bulk quantities of magnetic cards are also obtainable; prices are available on request.

PROGRAM LIBRARY

The Program Library (listed in Table 1) furnished with your 9100B 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 and 9100B users.

Please complete the 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.

9100B 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.

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

PERIPHERAL EQUIPMENT

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

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.

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 equipment will be available in the future.

SERVICE CONTRACTS

Service contracts are available for the 9100B 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 4; do not, however, make this check until you have completed the TURN-ON procedure given below.

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

GROUNDING REQUIREMENTS

To protect operating personnel, the National Electrical Manufacturer's 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 9100B 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 card, containing the Diagnostic Program, is in the pocket on the inside rear cover of the Program Library; this program completely 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 FLOATING POWER ON RUN

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**

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 the A arrow pointing 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).

Reinsert the Diagnostic card as before except with the B arrow pointing down.

PRESS: **ENTER**

When the card is fully ejected

PRESS: **CONT** ('continue') to run the program.

Correct operation of the 9100B Calculator is indicated by a display which will momentarily appear on the screen at intervals of about 3 seconds. The first display will appear as shown below:

```
--.  -- --- --- ---

 0. 000 000 000  00

--.  -- --- --- ---
```

ELECTRICAL INSPECTION
CONTINUED

The second time the display appears the 0's are changed to 1's, the third time to 2's and so on until all 9's appear. Once the cycle, 0's through 9's, has been completed the program automatically recycles and will continue to do so until the 'STOP' key has been pressed.

One cycle of 0' through 9's is sufficient to verify that the calculator is operating satisfactorily.

The calculator is not operating correctly:

if there is no display at all after about 10 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, 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 9100B, 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 9100B 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 9100B Calculator can outperform many computers. Despite its tremendous capability the calculator remains simple to operate.

ROM

The 9100B uses two memory systems. One is a unique Hewlett-Packard Read - Only - Memory (ROM). The ROM contains all the routines ('wired-in' programs) necessary to execute the instructions and calculate the many functions which can be called from the keyboard. These routines 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

This memory consists of 35 'registers'. Nineteen of the registers are shown in the diagrammatic sketch of Figure 1; the registers are the horizontal rows designated (+) 0 through (+) *f* and x, y and z. The sixteen registers, (+) 0 through (+) *f*, form a 'page', the '(+)' page'; a further sixteen registers (not shown in Figure 1) designated (–) 0 through (–) *f* form a similar page, the '(–) page'. The total of thirty-two registers on the (+) and (–) pages is used for storing either program steps or data (as shown in Figure 1). The three remaining registers, (x, y and z) contain the information which is displayed on the calculator screen.

On each of the two pages, the first 14 registers, (±) 0 through (±) *d*, can be used for storing either program steps or data; each register contains 14 'characters', these are the horizontal locations designated 0 through *d* (the + or – sign is not required for the characters). Each character can contain either one program step (one key pressed) or one digit of a data number; if no data is stored in the (±) 0 through (±) *d* registers then a maximum of 392 (2 x 14 x 14) program steps can be accommodated.

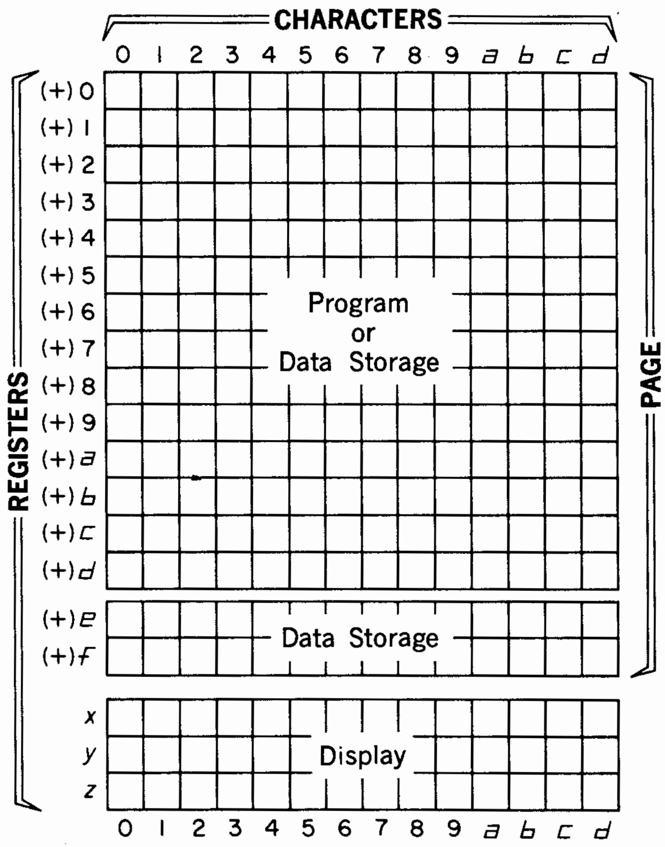


Figure 1. Magnetic-Core Memory

Program steps and data cannot both be stored in any one register at the same time. Each data number requires a complete register (14 characters) for storage, thus for each data number stored the maximum possible number of program steps is reduced by 14.

The remaining two registers (±) *e* and (±) *f*, on each page, are used for data storage only.

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:

PRESS: CLEAR 6 . 3 4

DISPLAY: 6.34 $\rightarrow x$
("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 and one in the Y register, appears in the Y register.

EXAMPLE:

6.34 ÷ 2 = 3.17

(with 6.34 → *x*)

PRESS: ↑ 2 ÷

DISPLAY: 3.17 → *y*

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 9100B Calculator is the tremendous range of numbers (1×10^{-98} to $9.999\ 999\ 999 \times 10^{99}$) which it accurately handles without special attention on the part of the operator. 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 'principal value' consists of the digits of the number with the decimal point placed after the first digit (e.g. 1.23456). 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)	1234.56	=	1.23456	x	10^3
Example (b)	.00123456	=	1.23456	x	10^{-3}
Example (c)	1.23456	=	1.23456	x	10^0

GUARD DIGITS

Although 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

GUARD DIGITS
CONTINUED

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:	CLEAR	6	•
PRESS:	0	0	0
PRESS:	0	0	0
PRESS:	0	0	1
DISPLAY:	6.000000001	$\rightarrow x$	
PRESS:	↑	3	÷
DISPLAY:	2.000000000	$\rightarrow y$	

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	$\rightarrow y$

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**

PRESS: π $\uparrow$

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

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 $\rightarrow y$

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

CONTINUED

OVERFLOW AND UNDERFLOW
CONTINUED

PRESS: CLEAR 1 2 3 •

PRESS: 4 5 6 7

PRESS: 8 9 8

DISPLAY: 123.4567898 $\rightarrow x$

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 $\rightarrow x$

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 $\rightarrow x$

CONTINUED

SET DECIMAL DIGITS: 5

DISPLAY: *.00000* $\longrightarrow x$

the number underflows and zeros appear in the X register.

SWITCH: **FLOATING**

DISPLAY: *4.* *-06* $\longrightarrow x$

accuracy is not lost even though the display did not automatically change 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 9100B 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: **X** (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 21.

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 iv 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 section of the 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	STEP	KEY
1	CLEAR	5	5
2	2	6	0
3	ENTER EXP	7	↑
4	CHG SIGN		

in which the keys are pressed in STEP sequence 1, 2, 3, 4, etc. Use of 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* $\rightarrow y$

'Enter' implies 'insertion' unless stated otherwise.

EXAMPLE:

"Enter a program" means "insert a program into the calculator."

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 '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. Resetting the switch from DEGREES (RADIANS) to RADIANS (DEGREES) will not automatically convert degrees (radians) to radians (degrees); to make the conversion refer to Page 44.

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, 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 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 = 1.234567898×10^4

DISPLAY

```
1.234 567 898  04
```

SWITCHES

FLOATING FIXED POINT
CONTINUED

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

If the number to be displayed overflows to the left of the decimal point 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 17).

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

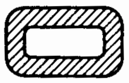
DECIMAL WHEEL SETTING	DISPLAY
SET: DECIMAL DIGITS to 5	12345.67898 → x
SET: DECIMAL DIGITS to 3	__12345.67<u>9</u> → x
SET: DECIMAL DIGITS to 6	1.234 567 898 04 → x

Leading zeros blanked

Last digit rounded

Number too large for this this setting - automatically reverts to floating point.

SWITCHES



ILLEGAL OPERATIONS LIGHT, to the left of the CRT, indicates that a mathematically illegal operation has been performed, either from the keyboard or from the 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 16 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 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 ( ) ( ), $x\rightarrow( )$, $x\leftarrow( )$, $y\rightarrow( )$ 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	$\rightarrow x$		

CLEAR x

CLEAR x **Clears the X register only (0. $\rightarrow$ 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 may have left an unterminated number in the X register.

ENTRY KEYS

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 is embarrassed as 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)

STEP	KEY	
1	CLEAR	
2	4	
3	5	
4	6	
5	↑	456 $\rightarrow y$ (accumulative total)

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

STEP	KEY		
6	7	7 $\rightarrow x$,	
7	+	7 $\rightarrow x$,	463 $\rightarrow y$
8	3		
9	2	32 $\rightarrow x$,	463 $\rightarrow y$

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

STEP	KEY		
10	1	321 $\rightarrow x$,	463 $\rightarrow y$

CONTINUED

ENTRY KEYS

CLEAR x

CONTINUED

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.	$\rightarrow$	x,	463	$\rightarrow y$
12	3					
13	2	32	$\rightarrow$	x,	463	$\rightarrow y$
14	+	32	$\rightarrow$	x,	495	$\rightarrow y$
15	4					
16	+	4	$\rightarrow$	x,	499	$\rightarrow y$
etc.	. . .					

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

STEP	KEY					
1-through-9 of the previous example		32	$\rightarrow$	x,	463	$\rightarrow y$
10	+	32	$\rightarrow$	x,	495	$\rightarrow y$

to determine if the + key has been used:

STEP	KEY					
11	1	1	$\rightarrow$	x,	495	$\rightarrow y$

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	$\rightarrow$	x,	495	$\rightarrow y$
13	4					
etc.	. . .					

CLEAR

CLEAR 'CLEAR' does not clear the entire contents of the memory.

Clears the display registers (0 $\rightarrow$ 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

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 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.

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* $\rightarrow x$

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

repeat with: **FIXED POINT**

DISPLAY: *1.000* $\rightarrow x$

(b) Enter 1 x 10^3 (1000)

SWITCH: **FLOATING**

PRESS: **CLEAR *x*** **ENTER EXP** 3

DISPLAY: *1.000 000 000 03* $\rightarrow x$

ENTRY KEYS

ENTER EXP

CONTINUED

(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

DISPLAY: 1.2678 02 → x

alternatively,

PRESS: CLEAR x 1 2 6

PRESS: 7 8 ENTER EXP 2

DISPLAY: 1.2678 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

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 **Changes the sign of the contents of the X register. If ENTER EXP was pressed, changes the sign of the exponent only.**

CHG SIGN

EXAMPLE:

Enter -4.9×10^{-3} (−.0049)

SWITCH: **FLOATING**

SET: DECIMAL DIGITS to 4

PRESS: CLEAR x 4 9 CHG SIGN

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.

EXAMPLE:

Enter -1×10^{12}

SWITCH: **FLOATING**

PRESS: CLEAR x ENTER EXP 1 2

PRESS: • CHG SIGN

DISPLAY: -1.000 000 000 12 $\rightarrow x$

normally -1×10^{12} would be entered as:

PRESS: CLEAR x CHG SIGN ENTER EXP 1 2

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

EXAMPLE:

-4.5×10^{-2} is required
but $4. \times 10^{-2}$ is entered in error.

SWITCH: **FLOATING**

PRESS: CLEAR x 4 ENTER EXP CHG SIGN 2

DISPLAY: 4. -02 $\rightarrow x$

to correct this without using CLEAR X key:

PRESS: • 5 CHG SIGN

DISPLAY: -4.5 -02 $\rightarrow x$

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 3

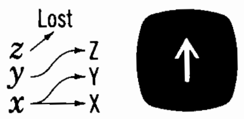
repeat the above example.

DISPLAY: -.045 $\rightarrow x$

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.**



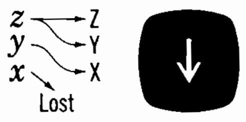
EXAMPLE:

SWITCH: **FIXED POINT**

PRESS: 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.**



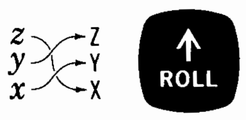
EXAMPLE:

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.**

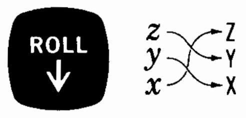


Step saving hint: ROLL ↓ has the same effect as ROLL ↑, ROLL ↑.

EXAMPLE:

See ROLL ↓ key.

CONTROL KEYS



'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**

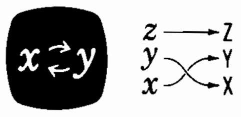
PRESS: CLEAR 9 ↑ROLL π

PRESS: ↑ROLL ↑ROLL

DISPLAY: 3.142 $\rightarrow z$
0. $\rightarrow y$
9. $\rightarrow x$

steps can be saved by using the ROLL ↓ key

PRESS: CLEAR π ROLL↓ 9



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 ↑ 2 +

PRESS: 2 0 ***x⇄y*** ÷

DISPLAY: 4.00 $\rightarrow y$

ARITHMETIC KEYS

The four arithmetic keys, +, −, x, ÷, 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.**

+

EXAMPLES:

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

(a) $8 + 4 = 12$

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

PRESS: 8 ↑ 4 +

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

(b) $6 + 7 + 5 = 18$

(contents of Z to remain unchanged throughout the calculation)

PRESS: 6 x⇄y 7 +

PRESS: 5 +

```
DISPLAY:   original contents of Z  →  z
                              18.  →  y
                               5.  →  x
```

ARITHMETIC KEYS

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

EXAMPLES:

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

(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 throughout the calculation)

PRESS: 4 7 $x \rightleftarrows y$ 1 9

PRESS: − 8 − 5 +

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

× **Multiplies the number in the Y register by the numbe in the X register. The product appears in Y; the X anc Z registers remain unchanged.**

EXAMPLES:

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

(a) $9 \times 7 = 63$

PRESS: 9 ↑ 7 ×

DISPLAY:		
original contents of Y	→	z
63.	→	y
7.	→	x

ARITHMETIC KEYS

(b) [(3 X 4) − 6] X 8 = 48

PRESS: 3 ↑ 4 ×

PRESS: 6 − 8 ×

DISPLAY: original contents of Y $\rightarrow z$
48. $\rightarrow y$
8. $\rightarrow x$

(c) $5^2 = 25$

PRESS: 5 ↑ ×

DISPLAY: 25. $\rightarrow y$

(d) $5^4 = 625$

(see also use of the ln x and e^X keys)

PRESS: 5 ↑ × × ×

DISPLAY: 625. $\rightarrow y$

÷ **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 ÷ 9 = 4

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 5

PRESS: 3 6 ↑ 9 ÷

DISPLAY: original contents of Y $\rightarrow z$
4.00000 $\rightarrow y$
9. $\rightarrow x$

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 5

		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	(3x4) ⟶ 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 (3x4)
12	+	−1	11	12	(3x4)+(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	(8x2) −6 ⟶ Y
20	ROLL ↓	10	11	6	recall (3x4) + (8−9)
21	÷	10	1.1	6	$\frac{(3x4) + (8-9)}{(8x2) - 6} = 1.1 \longrightarrow Y$

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 adding (+) 8 and (–) 9 in the numerator separately 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).

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

		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	-	(3x4)+(8–9) ⟶ Y
9	8	8	11	-	..
10	↑	8	8	11	(3x4)+(8–9) stored in Z
11	2	2	8	11	..
12	×	2	16	11	(8x2) ⟶ Y
13	6	6	16	11	..

CONTINUED

ARITHMETIC KEYS

STEP	KEY	DISPLAY REGISTERS X	Y	Z	NOTES
14	—	6	10	11	(8x2)–6 $\rightarrow$Y
15	↓	10	11	11	recall (3x4)+(8–9)
16	÷	10	1.1	11	$\frac{(3x4)+(8-9)}{(8x2)-6} = 1.1 \rightarrow Y$

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, $n_1 n_2 + n_3 n_4 + n_5 n_6 + \cdots$ 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 is shown following the program.

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 $(n_1 n_2)$
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 $(n_1 n_2) + (n_3 n_4)$
16	+	$(n_1 n_2) + (n_3 n_4) + (n_5 n_6) \rightarrow Y$
17	Enter n_7	- -

- - - - - etc. - - - - -

ARITHMETIC KEYS

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 [$x\!\to\!(\,)$, $y\!\to\!(\,)$, $x\!\leftarrow\!(\,)$, $y\!\rightleftarrows\!(\,)$] are used in conjunction with the numeric keys (0 through 9), the alphabetic keys (*a* through *f*) and the sign key (–) to provide access to the storage registers (see Magnetic Core Memory, Page 8).

When storing or recalling, the plus (+) instruction is not required if the register is on the (+) page; however, the minus (–) instruction is necessary if the register is on the (–) page.

RCL is a special case; this key is also included under VECTOR KEYS (Page 51) where the ACC+ and ACC– keys are also discussed.

It is not necessary to clear a register before storing, as the new data will automatically replace the old data.

$x\!\to\!(\,)$

$x\!\to\!(\,)$ **Stores the contents of the X register into the storage register selected by the next key pressed [for the (+) page] or the next two keys pressed [for the (–) page]; the contents of the X register remain unchanged.**

(+) page: press $x\!\to\!(\,)$, followed by one of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, or *f*;

(–) page: press $x\!\to\!(\,)$, (–), followed by one of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, or *f*.

EXAMPLE:

Store π (a) in the (+) *c* register [i.e. on the (+) page]
(b) in the (–) 2 register [i.e. on the (–) page]

PRESS: π

(a) PRESS: $x\!\to\!(\,)$ *c*

(b) PRESS: $x\!\to\!(\,)$ – 2

$x\!\leftarrow\!(\,)$

$x\!\leftarrow\!(\,)$ **Recalls, to the X register, the contents of the storage register designated by the next key(s) pressed. The contents of the storage register remains unchanged.**

(+) page: press $x\!\leftarrow\!(\,)$ followed by one of: 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9;

(–) page: press $x\!\leftarrow\!(\,)$, (–), followed by one of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, or *f*.

STORAGE AND RECALL KEYS

It is not necessary to use the $x \leftarrow (\,)$ key when recalling from a (+) alphabetic register.

NOTE

The $x \leftarrow (\,)$ and HYPER instructions have the same octal code (see Page 58). Use of these two keys is interchangeable.

EXAMPLE:

recall π from the (+)*c* and (–)2 registers (stored in the previous example).

PRESS: **CLEAR**

(the CLEAR is not a necessary operation; it is included in this example only to demonstrate that π is recalled.)

To recall π from (+)*c*:

PRESS: *c*

DISPLAY: π appears in X.

To recall π from (–)2:

PRESS: **CLEAR**

(incorrect)

PRESS: **–** **2**

DISPLAY: 2. $\rightarrow x$

(correct)

PRESS: $x \leftarrow (\,)$ **–** **2**

DISPLAY: π appears in X.

STORAGE AND RECALL KEYS

$y \to (\ )$

$y \to (\ )$ Stores the contents of the Y register in the storage register selected by the next key(s) pressed. The contents of Y remain unchanged.

(+) page: press $y \to (\ )$ followed by one of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, or *f*;

(–) page: press $y \to (\ )$, (–), followed by one of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, or *f*.

No example is given as the $y \to (\ )$ key stores the contents of Y in exactly the same manner as the $x \to (\ )$ key stores the contents of X.

$y \rightleftarrows (\ )$

$y \rightleftarrows (\ )$ 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(s) pressed.

(+) page: press $y \rightleftarrows (\ )$, followed by one of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, or *f*;

(–) page: press $y \rightleftarrows (\ )$, (–), followed by one of: 0,1,2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, or *f*.

EXAMPLE:

enter π into the Y register and exchange with the contents of the (+)6 register.

PRESS: π ↑

PRESS: $y \rightleftarrows (\ )$ 6

DISPLAY: original contents of (+)6 $\longrightarrow y$

(if the display in Y appears to be nonsensical this is because the (+)6 register originally contained program steps.)

To recall π:

PRESS: $y \rightleftarrows (\ )$ 6

DISPLAY: $\pi \longrightarrow y$

the (+)6 register now contains its original contents.

The $y \rightleftarrows (\ )$ key is particularly useful for moving data from one location to another in the memory:

STORAGE AND RECALL KEYS

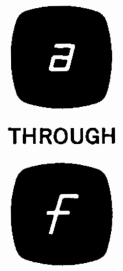
PRESS: *y*⇄( ) *f*

PRESS: *y*⇄( ) – *b*

PRESS: *y*⇄( ) 9

the contents of Y → (+)*f*,
the contents of (+)*f* → (–) *b*,
the contents of (–)*b* → (+)9,
the contents of (+)9 → Y.

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



STORAGE: See *x*→( ), *y*→( ), *y*⇄( ), ACC+ and ACC– keys.

RECALL: To recall the contents of the selected register to the X register (the contents of the recalled register remains unchanged):

(+) page: press one of *a*, *b*, *c*, *d*, *e*, or *f*;

(–) page: press *x*←( ), (–), followed by one of *a*, *b*, *c*, *d*, *e*, or *f*.

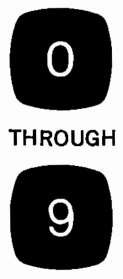
NOTE

The CLEAR instruction clears only the (+)*e* and (+)*f* registers.

See also RCL key in this section and GO TO ( ) ( ) key in the programming section.

The numeric keys [0 through 9 for the (+) page; –0 through –9 for the (–) page] when used for storage or recall must be used after one of the following keys:
***x*→( ), *x*←( ), *y*→( ), *y*⇄( ).**

(See also GO TO ( ) ( ) key in programming section.)



Unlike the alphabetic keys, if a numeric key only, or the minus (–) key followed by a numeric key, is pressed then the storage register will not be recalled; instead, the contents of the X register [X and Y if the minus (–) key is used] will be changed.

STORAGE AND RECALL KEYS

RCL

RCL Recalls (both) the contents of the (+)*e* register to Y and the contents of the (+)*f* register to X. The contents of (+)*e* and (+)*f* remain unchanged. It is not necessary to press (+)*e* and (+)*f* keys after RCL has been used.

RCL is not a 'general recall' instruction; it is, however, particularly valuable when used with the VECTOR KEYS (TO POLAR, TO RECT, ACC+, ACC–) as explained on Page 53.

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

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 recall from a numerical register requires at least two key strokes):

PRESS: 1 • 6 8 4

PRESS: *x*→( ) *a*

STORAGE AND RECALL KEYS

Enter n_1, recall k and multiply:

PRESS: 3 ↑ a ×

DISPLAY: (kn_1) *5.052* $\rightarrow y$

Enter n_2, recall k and multiply:

PRESS: 1 1 • 2 ↑

PRESS: a ×

DISPLAY: (kn_2) *18.8608* $\rightarrow y$

Enter n_3 - - - - - etc.

FUNCTION KEYS

DEGREES RADIANS

This switch selects the units, degrees or radians, to be used anc 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° (oı greater) can be calculated; however, the inverse trigonometrio functions are calculated only for the principle values of the functions:

$\theta = \sin^{-1}x;\ -90° \leq \theta \leq +90°;\quad (-\pi/2 \leq \theta \leq +\pi/2$
$\theta = \cos^{-1}x;\quad 0° \leq \theta \leq +180°;\quad (\quad 0 \leq \theta \leq +\pi$
$\theta = \tan^{-1}x;\ -90° \leq \theta \leq +90°;\quad (-\pi/2 \leq \theta \leq +\pi/2$

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 intc 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 $\frac{\pi}{180°}$ to convert to radians;

multiply radians by $\frac{180°}{\pi}$ 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 the ARC and HYPEF 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:**

arc ▼

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.**

hyper ▼

NOTE

The $x \leftarrow ()$ and HYPER instructions have the same octal code (67) (see Page 58); the use of these two keys is interchangeable.

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 $\longrightarrow x$

CONTINUED

FUNCTION KEYS

hyper ▼

CONTINUED

(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.

$\sqrt{x}$

$\sqrt{x}$ 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 the TO POLAR key.

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.

FUNCTION KEYS

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).**

e^x

EXAMPLE:

enter e into the X register.

SWITCH: **FLOATING**

PRESS: 1 e^x

DISPLAY: 2.718 281 828 00 $\rightarrow x$

(NOTE: the guard digits contain · ·48)

EXAMPLE:

$19^{1 \cdot 6} = 1.111\ 746\ 475 \times 10^2$

SWITCH: **FLOATING**

PRESS: 1 9 ln x ↑

PRESS: 1 • 6 ×

PRESS: ↓ e^x

DISPLAY: 1.111 746 475 02 $\rightarrow x$

EXAMPLE:

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

SWITCH: **FLOATING**

PRESS: 2 3 ln x ↑

PRESS: 5 ÷ ↓ e^x

DISPLAY: 1.872 171 230 00 $\rightarrow x$

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 take antilogarithms to the base 10, the following formula is used:

$x = 10^y$ (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: ln 10 $\rightarrow y$

multiply ln 10 by y:

PRESS: • 6 0 2

PRESS: 0 6 × ↓

DISPLAY: $y \cdot \ln 10$ $\rightarrow x$

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

PRESS: e^x

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

FUNCTION KEYS

int x

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

int x

EXAMPLE:

integer −5.9 = −5

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 4

PRESS: CHG SIGN 5 • 9

DISPLAY: -5.9 $\rightarrow x$

PRESS: int x

DISPLAY: -5. $\rightarrow x$

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 —

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. $\rightarrow z$

(') 43.2 $\rightarrow y$

FUNCTION KEYS

$|y|$

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

EXAMPLE:

SWITCH:	**FLOATING**		
PRESS:	CLEAR x	CHG SIGN	5
PRESS:	ENTER EXP	CHG SIGN	↑
DISPLAY:	-5.	-00	$\rightarrow y$
PRESS:	$|y|$		
DISPLAY:	5.	-00	$\rightarrow y$

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.

π

π **Clears the X register and enters pi.**
Up to and including ten digits are displayed, but two more digits (- -60) are included in the guard digits (see Page 11).

VECTOR KEYS

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

Conversion from rectangular to polar coordinates will calculate the angle, Θ, in the range;

$-180^\circ < \Theta \leq +180^\circ$
$-\pi$ radians $< \Theta \leq +\pi$ radians

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:**

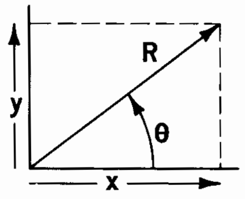
TO POLAR

Angle (Θ) $= \text{Tan}^{-1}\, y/x \longrightarrow$ Y

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

EXAMPLE:

(a): $x = 4$, $y = 3$, convert to polar form.



SWITCH: **DEGREES**

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 3

PRESS: 3 ↑ 4 TO POLAR

DISPLAY: (Θ, in degrees) 36.870 $\longrightarrow y$
(R) 5.000 $\longrightarrow x$

SWITCH: **RADIANS**

PRESS: repeat the preceeding example.

DISPLAY: (Θ, in radians) .644 $\longrightarrow y$
(R) 5.000 $\longrightarrow x$

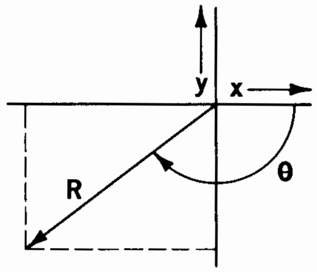
VECTOR KEYS

TO POLAR

CONTINUED

EXAMPLE:

(b): x = –4, y = –3, convert to polar form.



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*

TO RECT

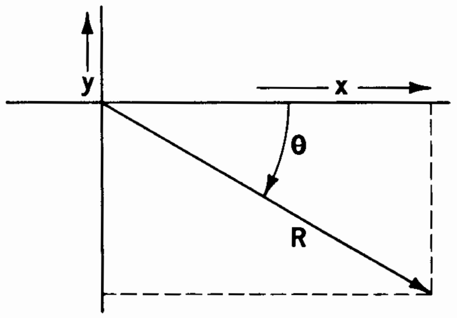
TO RECT **Converts polar coordinates, consisting of radius (R) in the X register at an angle (Θ) in the Y register, to rectangular coordinates:**

y component = R sin Θ →Y

x component = R Cos Θ →X

EXAMPLE:

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



VECTOR KEYS

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 $\rightarrow y$
(x) 6.928 $\rightarrow x$

The ACC+, ACC–, and RCL keys are special storage and recall keys, for use with the (+)*e* and (+)*f* registers only. These keys provide capability of vector addition and subtraction with single keystrokes.

Care should be taken, when using these keys, of the CLEAR operation as the CLEAR key clears both the (+)*e* and (+)*f* registers.

ACC +

ACC –

RCL

NOTE

The + and – signs on the ACC+ and ACC– keys are not references to the (+) and (–) pages of the memory; both keys are used only with the (+)*e* and (+)*f* registers.

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.

$(+)e + y \rightarrow (+)e$

$(+)f + x \rightarrow (+)f$

CONTINUED

VECTOR KEYS

ACC +

ACC −

RCL

CONTINUED

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.

$$(+)e - y \longrightarrow (+)e$$

$$(+)f - x \longrightarrow (+)f$$

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.

$$(+)e \longrightarrow Y$$

$$(+)f \longrightarrow X$$

EXAMPLE:

(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 +	$0+3=3 \longrightarrow (+)e$ $0+2=2 \longrightarrow (+)f$
6	5	- -
7	↑	- -
8	4	- -
9	ACC +	$3+5=8 \longrightarrow (+)e$ $2+4=6 \longrightarrow (+)f$
10	6	- -
11	CHG SIGN	- -
12	↑	- -
13	3	- -
14	ACC −	$8-(-6)=14 \longrightarrow (+)e$ $6-3=3 \longrightarrow (+)f$
15	RCL	$(+)e \longrightarrow Y$ $(+)f \longrightarrow X$

DISPLAY: (y) 14. $\longrightarrow y$

(x) 3. $\longrightarrow x$

VECTOR KEYS

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.

EXAMPLE:

(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 can be easily adapted to solve problems with many complex terms:

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

SWITCH: **DEGREES**

SWITCH: **FIXED POINT**

SET: DECIMAL DIGITS to 3

clear the $(+)e$ and $(+)f$ registers:

PRESS: **CLEAR**

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

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 ($R_2\ \angle\theta_2$):

PRESS: **3** **↑** **2** **CHG SIGN** **TO POLAR**

CONTINUED

VECTOR KEYS

ACC +

ACC −

RCL

CONTINUED

take the log of R_2 and add ln R_2 and $\underline{/\Theta_2}$ to (+)*f* and (+)*e* respectively:

PRESS: ln *x* ACC +

take the antilog of (ln R_1 + ln R_2)

PRESS: RCL e^x

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

PRESS: TO RECT

DISPLAY: (j) 1. $\rightarrow y$
-18. $\rightarrow x$

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 9100B 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 9100B is simple as the keyboard operations are the program instructions and no special program 'language' has to be learned. It is, however, 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, 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 one page of the magnetic-core memory. The storage memory consists of two pages, the '(+) page' and the '(–) page'.

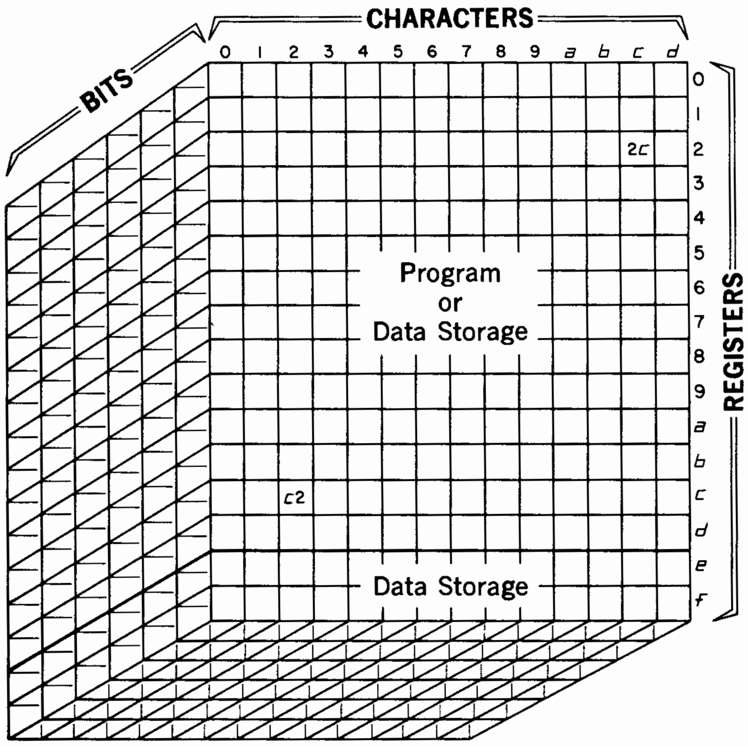


Figure 2. Programmable Memory

INTRODUCTION TO PROGRAMMING

REGISTERS
CONTINUED

As previously described, there are sixteen registers per page:

(±) 0 through (±)*d* – these registers are used for either program steps or data storage;

(±)*e* through (±)*f* – these registers are used for data storage only.

CHARACTERS

The twenty-eight program registers, (±) 0 through (±) *d*, each contain fourteen locations called 'characters'; each character can contain either, one program step (one key stroke) which gives a maximum capacity of 392 program steps, or one digit of a data number. Each piece of data consists of a twelve-digit number 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 392 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.

MEMORY ADDRESS

Each 'character' has a unique three-part address; the three parts are (always in this order):

1) the sign of the page (+ or –),
2) the register designation (0 through *d*),
3) the character designation (0 through *d*).

Thus, the third character in register *c* on the (–) page is addressed as '(–) *c*2';

'(–)2*c*' is the address of character *c* in the third register on the (–) page;

'(+)2*c*' is the address of character *c* in the third register on the (+) page.

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 the two-digit instruction codes can be stored in the six 'bits' of 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).

INTRODUCTION TO PROGRAMMING

It is not necessary to use the octal code in order to program the calculator, so until it becomes necessary to edit and correct a program, it is more convenient to consider the memory in terms of only the two dimensions, registers and characters.

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 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, (+)0*a*, (+)0*b*, (+)0*c*, (+)0*d*, (+)10, · · · · · (+)19, (+)1*a*, (+)1*b*, (+)1*c*, (+)1*d*, (+)20, · · · · · up to · · · · · (+)*dc*, (+)*dd*.

When address (+)*dd* is filled the (+) page is completed and the 'program counter' automatically switches to the (–) page:

· · · (+)*dc*, (+)*dd*, (–)00, (–)01, · · · · · 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.

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 68).

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 83.

An 'unconditional' branch gives the calculator no option; it must branch to the address indicated in the program (See GO TO key on Page 68).

INTRODUCTION TO PROGRAMMING

SUBROUTINES

The 9100B has subroutine capability. A subroutine is a sequence of program steps which is to be used many times, but which is stored only once in the memory. The program may 'call for' (i.e. branch to) the subroutine at any point designated in the program. When the subroutine is completed the program automatically returns to the step following the program step from which the subroutine was 'called'. Thus a repetitive subroutine may be stored only once in the memory yet used many times throughout a program; this results in considerable saving of program memory space. Subroutines are fully described, starting on Page 94.

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.

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 display; (in most cases) if the sequence was not followed exactly, the resulting display 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 $\rightarrow z$

(B) 5 $\rightarrow y$

(C) 3 $\rightarrow x$

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).

		DISPLAY		
STEP	KEY	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

SWITCH: **PROGRAM**

PRESS: the keys in the program sequence.

SWITCH: **RUN**

PRESS: END

to run (and rerun) the program:
enter A, B and C into Z, Y and X respectively.

PRESS: CONT

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	$\frac{A \times B}{C}$	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	$\frac{A \times B}{C}$	A

PROGRAM WRITING

WRITING A PROGRAM

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

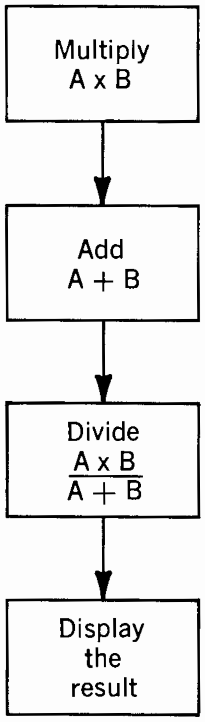
a) define the problem;
b) decide how the problem is to be solved;
c) write the steps sequentially for the calculator.

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 chart should be as simple as possible.

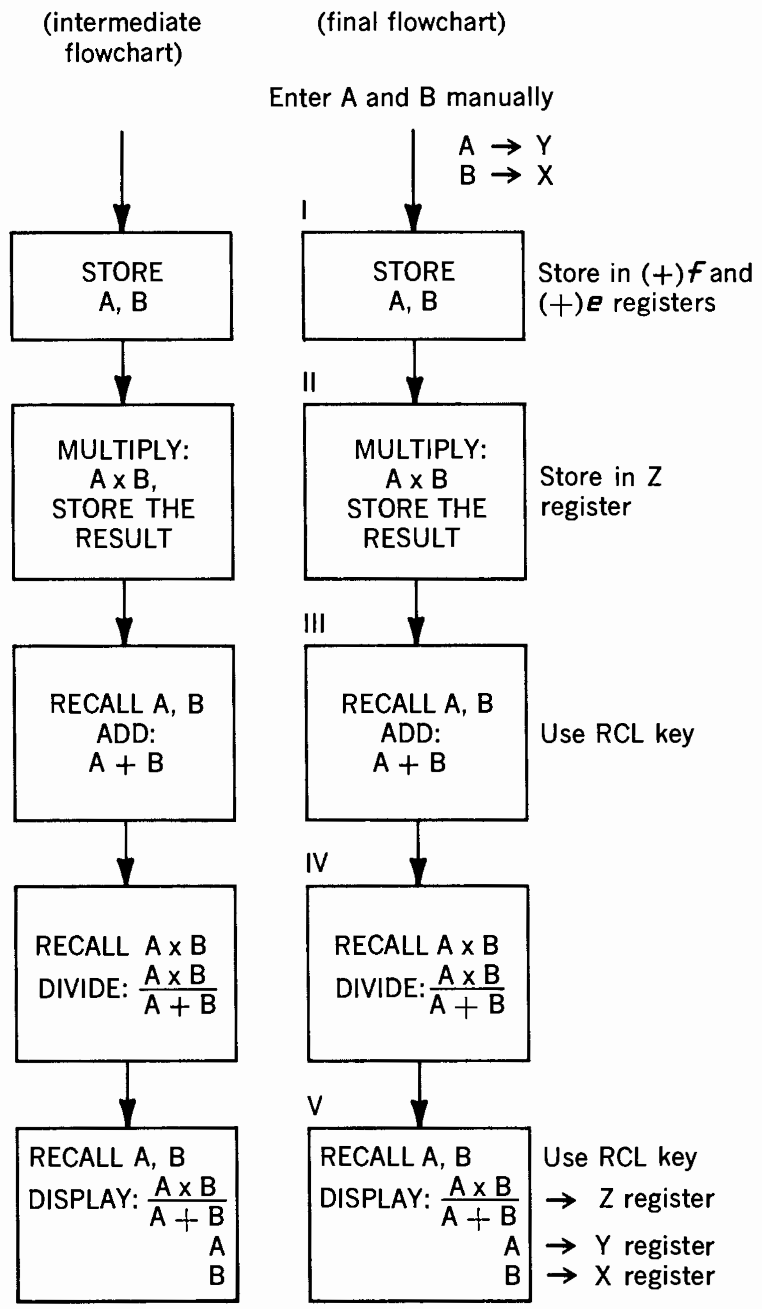
(initial flowchart)



Next, draw the 'flowchart' 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



c) 'write the steps sequentially for the calculator'

Write the steps exactly as the keys would b pressed if the operations were to be performe manually; at each step write down the effect o the operation on the display and storage regis ters (the program pad, supplied with each cal culator, may be used for this).

PROGRAM WRITING

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

STEP	KEY	DISPLAY			STORAGE	
(address)		X	Y	Z	(+)f	(+)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:

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

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

STEP	KEY	DISPLAY			STORAGE	
(address)		X	Y	Z	(+)f	(+)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	KEY	DISPLAY			STORAGE	
(address)		X	Y	Z	(+)f	(+)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

PROGRAM WRITING

WRITING A PROGRAM
CONTINUED

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

STEP (address)	KEY	DISPLAY X	Y	Z	STORAGE (+)*f*	(+)*e*
0*a*	↑	A + B	A + B	$\frac{A \times B}{A + B}$	B	A
0*b*	RCL	B	A	$\frac{A \times B}{A + B}$	B	A
*0*c*	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 it automatically resets the program counter to (+)00 ready for the next value of A and B to be entered.

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	Y	Z	STORAGE (+)*f*	(+)*e*
(+)00	$x\to( )$	23	B	A	–	–	–
01	*f*	15	B	A	–	B	–
02	$y\to( )$	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
0*a*	↑	27	A + B	A + B	$\frac{A \times B}{A + B}$	B	A
0*b*	RCL	61	B	A	$\frac{A \times B}{A + B}$	B	A
0*c*	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.

The preceeding program could have been started as:

STEP	KEY	KEY	DISPLAY			STORAGE	
(address)		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 CONT (continue) key twice to run the program; the first time to start the program and the second time after the data is entered. With the original program CONT has to be pressed only once, after data entry (see Page 69 for use of CONT key).

PROGRAM KEYS

PROGRAM RUN

Selects the mode of operation.

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

RUN MODE: used when performing calculations manually, re cording a program on a magnetic card, entering a progra from a magnetic card, running a program and addressin the program counter.

GO TO ( )( )

GO TO ( )( ) Used to address the program counter and as an unco ditional branch to the address designated by the ste immediately following the GO TO instruction.

GO TO, followed by the SUB/RETURN instruction, followed b an address, causes an unconditional branch to the subroutin starting at the address designated (see Page 94).

When GO TO is used either from the keyboard or as a progra operation use of the page sign (+ or –) is page dependent; th sign of the page is required only if the branch is to be betwee pages:

from ( ) page	to ( ) page	Required Instructions	Number of steps
(+)	(+)	GO TO, 4, 3	3
(–)	(–)	GO TO, 4, 3	3
(+)	(–)	GO TO, –, 4, 3	4
(–)	(+)	GO TO, +, 4, 3	4

NOTE

Remember that, whereas branching is page dependent, addressing a register, for data storage or recall, is not page dependent. In this case, regardless of whether the addressing instructions are on the (+) page or the (–) page, a plus (+) register is automatically assumed unless the (–) is used to indicate that the required register is on the (–) page.

PROGRAM KEYS

One program step is required for each key stroke used to address the memory:

CORRECT

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

INCORRECT

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

If the GO TO, followed by an address (but no page sign), requires program steps on both pages, then the branch will be to the address on the same page as the GO TO instruction:

STEP	KEY
(+)dc	GO TO ( )( )
(+)dd	5
(−)00	8
(−)01	

the program will go to address (+)58.

CONT 'CONTINUE' is 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 79).

If the program has halted at a STOP instruction, press CONT to continue running the program (at the next program step).

If the program has halted at an END instruction, press CONT to run the program [starting at address (+)00].

CONT

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.

STOP

PROGRAM KEYS

END

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

When used as a program step END is the equivalent of:

STOP, GO TO, +, 0, 0.

When used from the keyboard, END is the equivalent of:

GO TO, +, 0, 0.

If END is included in a program being entered from a magnetic card, the END instruction will stop the card reader and reset the program counter to (+)00. This means that anything following the END instruction, if it is on the same page as the END instruction, cannot be re-entered into the calculator. For this reason if the 'END' appears other than as the last required instruction on a page, or if data follows the END, then replace the END instruction, by any other instruction, before recording the program. When the program has been re-entered from the card the END instructions can be re-inserted (see CORRECTING A PROGRAM on Page 79).

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, each time STEP PRGM is pressed, the next address to be stepped (together with the instruction code at that address) appears in the Z register. The last address stepped (and the instruction code at that address) appears in the X register.

NOTE

The original contents of the Z register will reappear when the PROGRAM - RUN switch is switched back to RUN; the original contents of the X register are lost.

In RUN mode, STEP PRGM single-steps the program, displaying the contents of 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 and CORRECTING A PROGRAM (Pages 76 and following). Several program instructions do not operate as expected when the STEP PRGM key is used; these are also explained under EDITING A PROGRAM.

PROGRAM KEYS

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 CONT to continue running the program).

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 stopping the program, 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 CONT after each STOP had occurred). Without the PAUSE instruction the display will remain blanked for the entire calculation; the operator has no way of knowing, during that time, whether the program is running correctly or not; in this situation the PAUSE can provide a valuable confidence factor.

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/SPACE instruction will act as the STOP instruction; this will require CONT to be pressed to continue running the program.

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. If the FMT is included in a program and no peripheral equipment is being used, then the program will stop after reading FMT and the next instruction; press CONT to continue running the program, starting at the second step after the FMT.

When the FMT key is pressed, the display will blank; press any key and the display will return.

For complete information, see the Operating and Service Manual for the particular peripheral.

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.

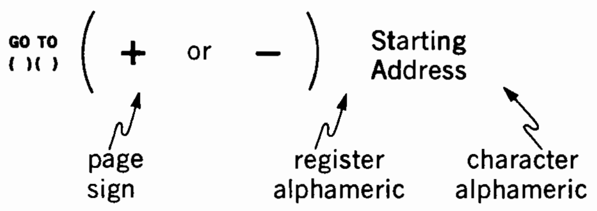
Recording a program (and data) on a magnetic card.

Entering a program from a magnetic card.

Editing a program.

Correcting a program.

In this section the keying symbols shown below will be used when addressing the program counter:



Use either the (+) or (–) key, depending on which page, the (+) page or the (–) page, is to be addressed.

ENTERING A PROGRAM FROM THE KEYBOARD

1. Address the program counter:

   SWITCH: **RUN**

   PRESS: **GO TO ( )( )** (**+** or **−**) Starting Address

   NOTE

   If the starting address is (+)00 then 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.

PROGRAM OPERATION

As each step is entered, the X register displays the address and the octal code for the instruction; the Z register displays the next address to be used and the instruction contained in that address. The original contents of Y remain in Y.

EXAMPLE:

```
DISPLAY:          0.1          17    →  z
          original contents of Y    →  y
                  0.0          23    →  x
```

address (+)00 contains octal code 23, the code for the $x\rightarrow$( ) key; address (+)01, the next address to be used, contains octal 17, the code for the *d* key.

3. Run the program:

SWITCH: **RUN**

PRESS: GO TO ( )( ) $\left(+ \text{ or } - \right)$ Starting Address

or

PRESS: END if starting address is (+)00.

a. The program will run, beginning with the instruction in the starting address, when CONT is pressed.
b. Enter data and press CONT 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**

CONTINUED

PROGRAM OPERATION

ENTERING A PROGRAM FROM THE KEYBOARD
CONTINUED

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; the Z register displays the next address to be used together with whatever step is in that address.

Address the program counter:

SWITCH: **RUN**

PRESS: GO TO ( )( ) 0 0

or

PRESS: END

Enter data (A and B):

PRESS: digit keys for A

PRESS: ↑

PRESS: digit keys for B

Run the program:

PRESS: CONT

DISPLAY:

$$\frac{A \times B}{A + B} \rightarrow z$$

$$A \rightarrow y$$

$$B \rightarrow x$$

To rerun the program, enter data as before and press CONT. In this program it is not necessary to readdress the program counter each time, the END instruction automatically resets the counter to (+)00.

PROGRAM OPERATION

RECORDING A PROGRAM ON A MAGNETIC CARD

RECORD

The magnetic card has two parts, A and B. The entire contents of one page 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 card-reader slot with the A arrow pointing down. Ensure that the printed side of the card is facing the keyboard. To record (or read) part B, insert the card in the same manner with the B arrow pointing down.

> **NOTE**
> If data is to be recorded, following the program steps, the END instruction cannot be used on the same page as the data; if it is, the data will not be re-entered into the calculator memory as END will stop the card reader during entry.

To record a program:

SWITCH: **RUN**

PRESS: GO TO (+ or −) [illegible] rti dr [illegible]

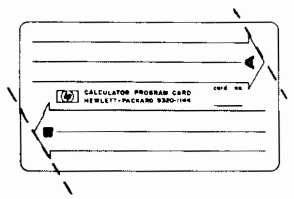
INSERT: magnetic card into the card reader (with the A arrow down if both parts of the card are to be used).

PRESS: RECORD (hold the key pressed until the card is partially ejected). If both A and B parts of the card are to be used, reinsert the card with the B arrow pointing down and press RECORD.

> **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.



PROGRAM OPERATION

ENTERING A PROGRAM FROM A MAGNETIC CARD

ENTER

SWITCH: **RUN**

PRESS: GO TO ( ) ( ) (+ or −) Start Addr

INSERT: magnetic card into the card reader (with A or B arrow pointing down as required - A if both pages are to be entered).

PRESS: ENTER

NOTE

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

If the (+) page of the memory has just been filled, the pro gram counter will now be set to (–)00, unless the END instruc tion was entered on the (+) page. In this case the progran counter will now be set to (+)00.

To enter the B part of the card on the (–) page:

(if necessary)

PRESS: GO TO ( ) ( ) − Starting Address

insert the card, with the B arrow down and pres ENTER.

The program is now inserted and ready to be run.

EDITING A PROGRAM

When a program has been entered into the calculator it may b 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 step have been entered as written in the program.

SWITCH: **RUN**

PRESS: GO TO ( ) ( ) (+ or −) art dr

SWITCH: **PROGRAM**

PRESS: STEP PRGM

PROGRAM OPERATION

Each time the STEP PRGM 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. The Z register will display the next program address and the instruction in that address.

> **NOTE**
>
> When in the PROGRAM mode pressing any key, other than STEP PRGM, will cause that instruction 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.

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

SWITCH: **RUN**

PRESS: GO TO ( ) (+ or −) [illegible] rt dr [illegible]

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.

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

CONT FMT IF x<y IF x=y IF x>y [illegible]

If CONTinue is included in the program, this step must be replaced by a STOP or PAUSE instruction (see CORRECTING A PROGRAM, for this procedure) before the program can be edited in the RUN mode. The calculator will automatically continue execution of the program any time a CONT instruction is encountered.

When STEP PRGM is pressed at a FMT instruction both the FMT and the next instruction are read; the next time STEP PRGM is pressed the second instruction after the FMT will be executed.

The 'conditional' branch instructions (IF FLAG, IF x <y, IF x = y, IF x >y) cause the program counter to branch

PROGRAM OPERATION

EDITING A PROGRAM
CONTINUED

immediately, when the STEP PRGM key is used, if the condition is not met (see CONDITIONAL BRANCHING keys, Page 83).

As an example, suppose the following is part of a program to be edited in the RUN mode:

STEP	KEY
.....	.....
(+)03	+
.....	.....
08	IF $x=y$
09	0
0*a*	3
0*b*	×

When STEP PRGM is pressed at step 08, if the number in the X register is equal to the number in the Y register then, as would be expected, the STEP PRGM must be pressed at steps 09 and 0*a* before the program counter goes to address 03.

If the numbers in X and Y are not equal, then, when STEP PRGM is pressed at step 08, the program counter automatically goes to step 0*b* and multiplies the numbers in X and Y.

The other 'IF' keys operate in the same way.

EXAMPLE:

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

address the program counter

SWITCH: **RUN**

PRESS: GO TO ( )( ) + 0 0

or

PRESS: END

enter data

PRESS: any digit keys for A.

PRESS: ↑

PRESS: any digit keys for B.

PROGRAM OPERATION

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.

CORRECTING A PROGRAM

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

SWITCH: **RUN**

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

SWITCH: **PROGRAM**

the step to be changed will appear in the Z register.

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

the correct step will appear in the X register.

SWITCH: **RUN**

EXAMPLE:

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

SWITCH: **RUN**

PRESS: GO TO ( ) ( ) + 0 7

SWITCH: **PROGRAM**

PRESS: −

SWITCH: **RUN**

PRESS: END

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.

PROGRAM OPERATION

CORRECTING A PROGRAM
CONTINUED

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).

EXAMPLE:

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	ROLL ↓	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:

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

SWITCH: RUN

PRESS: GO TO ( ) + 6 5

INSERT: magnetic card

PRESS: RECORD

PROGRAM OPERATION

Insert the two extra instructions in locations 65 and 66:

PRESS: GO TO ( ) ( ) + [illegible] [illegible]

SWITCH: **PROGRAM**

PRESS: ROLL ↑ |y|

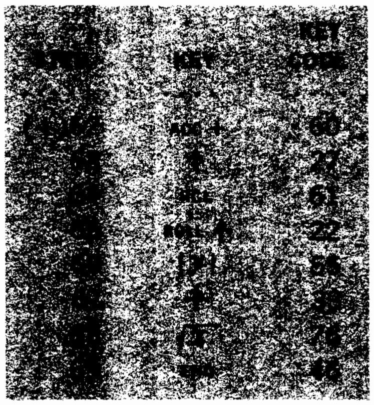
Re-enter the remaining steps (recorded on the program card) into address 67 and following:

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:



STEP	KEY	KEY CODE
[illegible]	ACC +	60
[illegible]	↑	27
[illegible]	RCL	61
[illegible]	ROLL ↑	22
[illegible]	|y|	56
[illegible]	+	33
[illegible]	√x	76
[illegible]	END	46

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

If a 'patch-in' technique is used to insert extra steps it is not necessary to change any branching address in the program.

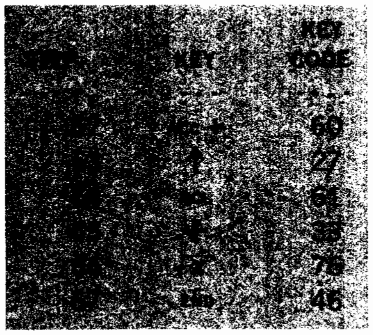
Using the same example as previously, it is required to insert

PROGRAM OPERATION

CORRECTING A PROGRAM

CONTINUED

two extra instructions (ROLL ↑ and |y|) between steps 64 and 65.

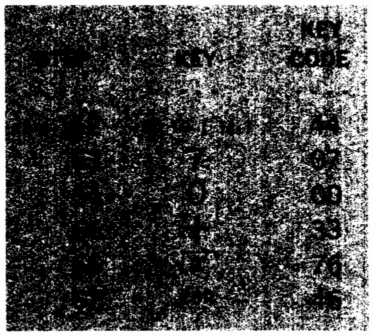


STEP	KEY	KEY CODE
[illegible]	[illegible]	60
[illegible]	↑	27
[illegible]	[illegible]	61
[illegible]	[illegible]	33
[illegible]	[illegible]	76
[illegible]	[illegible]	46

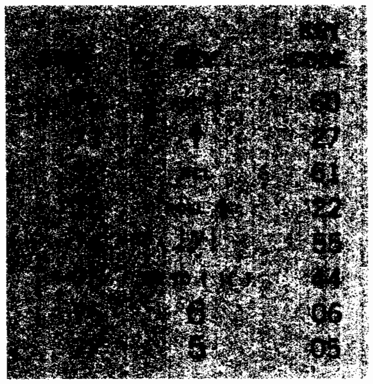
To insert steps:

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

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
[illegible]	[illegible]	44
[illegible]	7	07
[illegible]	0	00
[illegible]	[illegible]	33
[illegible]	[illegible]	76
[illegible]	[illegible]	46



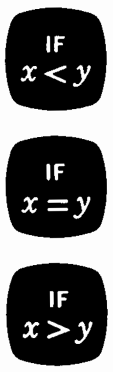
STEP	KEY	KEY CODE
[illegible]	[illegible]	60
[illegible]	↑	27
[illegible]	[illegible]	41
[illegible]	[illegible]	22
[illegible]	[illegible]	55
[illegible]	[illegible]	44
[illegible]	6	06
[illegible]	5	05

PROGRAM KEYS – CONDITIONAL BRANCHING

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

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

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.



NOTE

The 'IF' instructions compare all twelve (12) digits and the two-digit exponent of the numbers in the X and Y registers when testing for the condition indicated. However, if the principle value of a number consists of digit nine (9) in all twelve places then that number is considered to be equal to the next higher power of 10.
(e.g. $9.999\ 999\ 999\ 99 \times 10^2 = 10^3$).

CONDITION MET: When an 'IF' qualifier is met (YES) a 'GO TO' condition is set up. If the 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.

Any one of the instructions listed below, if contained in the step following the 'IF', constitutes the start of an address so that the next steps must contain the remainder of the address:

SUB/RETURN, (+), (–), any alphameric.

PROGRAM KEYS – CONDITIONAL BRANCHING

IF $x<y$

IF $x=y$

IF $x>y$

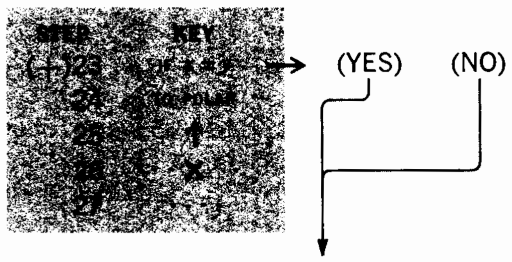
CONTINUED

> **NOTE**
>
> a) Use of the (+) or (–) page signs is page dependent (see GO TO key on Page 68) and, therefore, is not required unless the branch is between pages.
>
> b) The SUB instruction will cause a branch to the subroutine at the address designated.
>
> c) A 'continue' (CONT) cannot be used as a 'no operation', in the step after the 'IF' instruction, if it is followed by any one of the following:
>
> SUB/RETURN, (+),(–), any alphameric.
>
> This is because CONT does not clear the 'GO TO' condition set up when an IF instruction is met (YES).

The following examples, (1) through (4), are parts of imaginary programs. Each program contains some combination of possible instructions following the IF instruction. In each case the course of the program is mapped for 'condition met' (YES) and 'condition not met' (NO).

EXAMPLE (1)

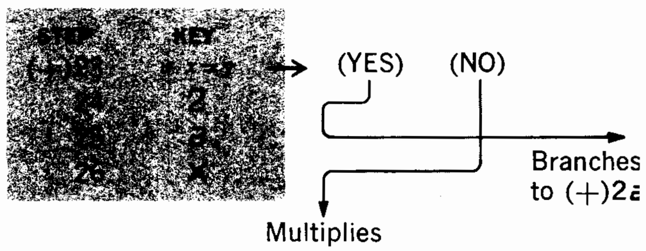
Operations follow the 'IF' instruction:



STEP	KEY
(+)23	IF $x=y$
24	CONTINUE
25	↑
26	×
27	[illegible]

EXAMPLE (2)

Address follows the 'IF' instruction:

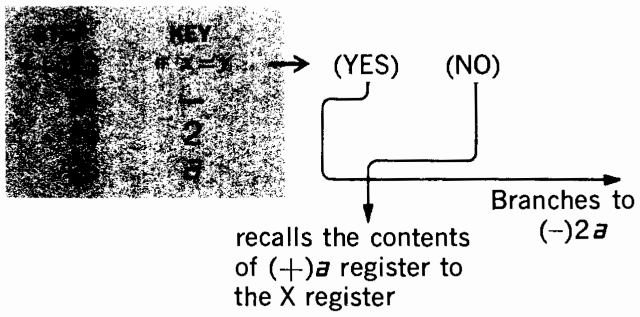


STEP	KEY
(+)23	IF $x=y$
24	2
25	3
26	×

PROGRAM KEYS – CONDITIONAL BRANCHING

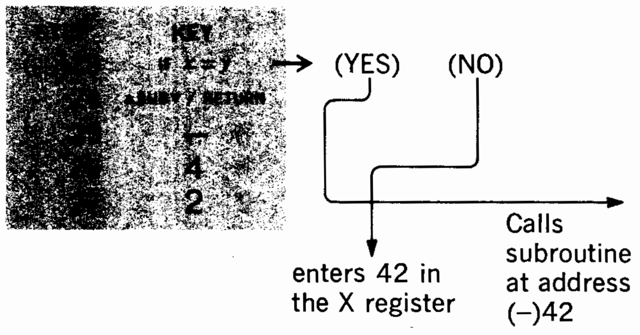
EXAMPLE (3)

Address (with sign) follows 'IF' instruction:



EXAMPLE (4)

SUB/RETURN follows 'IF' instruction:


KEY
IF x = y
SUB / RETURN
←
4
2
(YES)
(NO)
Calls subroutine at address (–)42
enters 42 in the X register

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

this program sums any number (N) until the total (Σn) is equal to, or greater than, 100;

the final display shows N in the Z register, 100 in Y and ΣN (the final Σn) in X. 'IF x <y'

is used to test if Σn (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

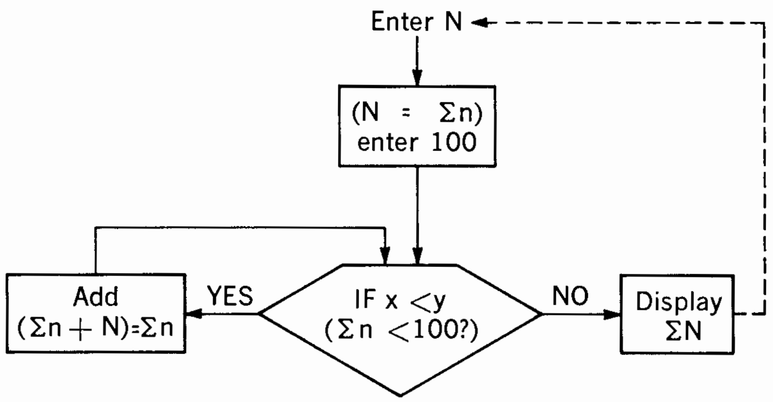
IF $x<y$

IF $x=y$

IF $x>y$

CONTINUED

FLOWCHART:



PROGRAM

to run the program press END (first time only) enter N and press CONT.

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

* flashing display
† final display

At step 06, if Σn 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 (+)0*a*. When Σn is not less than 100 (i.e. Σn = ΣN $\geq$ 100) the condition is not met; steps 07 and 08 are skipped and the instruction in 09 is executed.

PROGRAM KEYS – CONDITIONAL BRANCHING

The above program also illustrates the following:

a) CLEAR is not always a necessary instruction;

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.

e) the sign of the page is not required in the branching instructions (steps 07, 08 and 10, 11) as the branches do not change pages.

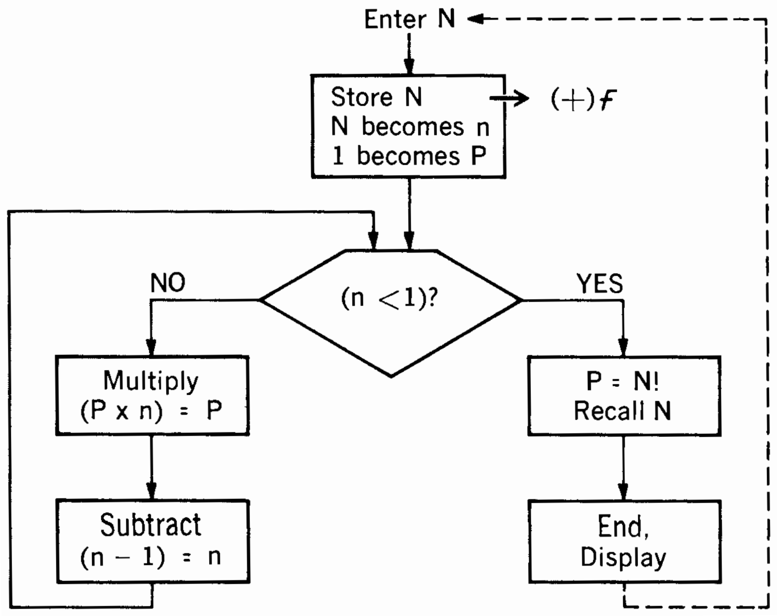
The following program calculates N! A new value for N can be entered at the end of the program, since END automatically sets the program counter back to (+)00. This program can calculate N! for values of N up to, and including, 69.

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

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

$0! = 1$ (by definition)

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



PROGRAM KEYS – CONDITIONAL BRANCHING

IF $x<y$

IF $x=y$

IF $x>y$

CONTINUED

PROGRAM

To run the program: press 'END' (the first time only), enter N and press 'CONT'.

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	STORAGE (+)*f*
(+)00	x→( )	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!	
13	END	46	N	0	N!	→ final display

IF FLAG

SET FLAG

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.

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.)

PROGRAM KEYS – CONDITIONAL BRANCHING

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:

$$\frac{x_1 + x_2 + x_3 + \cdots x_n}{N} = \frac{\Sigma x_i}{N} = \overline{X}$$

This program illustrates the following:

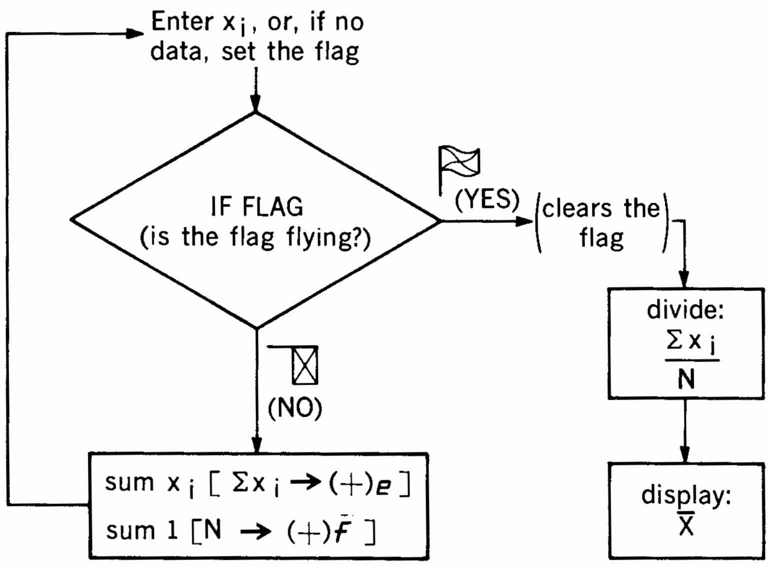
a) use of IF FLAG, with the SET FLAG instruction set from the keyboard;

b) generation of a program counter (the number of data points is counted by adding 1 to the program 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*, (+)*f* and the SET FLAG.

FLOW CHART


Enter x_i, or, if no data, set the flag
IF FLAG (is the flag flying?)
(YES)
clears the flag
divide: $\frac{\Sigma x_i}{N}$
(NO)
sum x_i [$\Sigma x_i \rightarrow (+)e$]
sum 1 [N $\rightarrow$ (+)f]
display: $\overline{X}$

PROGRAM KEYS – CONDITIONAL BRANCHING

IF FLAG

SET FLAG

CONTINUED

To operate the program:

1) PRESS: END CONT
2) enter x_1 in the X register
3) PRESS: CONT
4) enter x_2 in the X register
5) PRESS: CONT
6) repeat steps 4 **and** 5 for **all** remaining data points up to, **and including,** x_n
7) PRESS: SET FLAG CONT

FINAL DISPLAY: $\bar{X} \rightarrow y$
$N \rightarrow x$

		KEY	DISPLAY			STORAGE	
STEP	KEY	CODE	X	Y	Z	(+)f	(+)e
(+)00	CLEAR	20	0	0	0	0	0
01	STOP	41	$\begin{bmatrix}\text{Enter}\\ x_i\end{bmatrix}$				
02	IF FLAG	43					
03	0	00					
04	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	00					
0a	1	01					
0b	RCL	61	N	Σx_i	(x_{i-1})	N	Σx_i
0c	÷	35	N	$\frac{\Sigma x_i}{N}$	(x_{i-1})		
0d	END	46	$\boxed{N}$	$\boxed{\bar{X}}$	$\boxed{(x_{i-1})}$	→ final display	

PROGRAM KEYS – CONDITIONAL BRANCHING

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

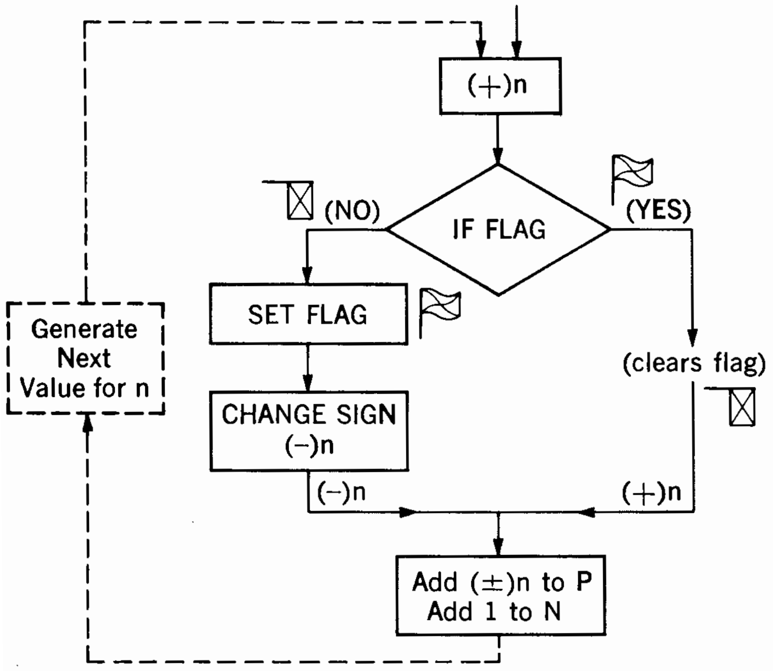
$$1 - 3 + 5 - 7 + 9 \cdots\cdots\cdots\cdots \infty$$

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.

The next time through the flag is not set:

(a) negative (–)n is added to P;

(b) the flag is set [so that the next time through a (+)n is added].

PROGRAM KEYS – CONDITIONAL BRANCHING

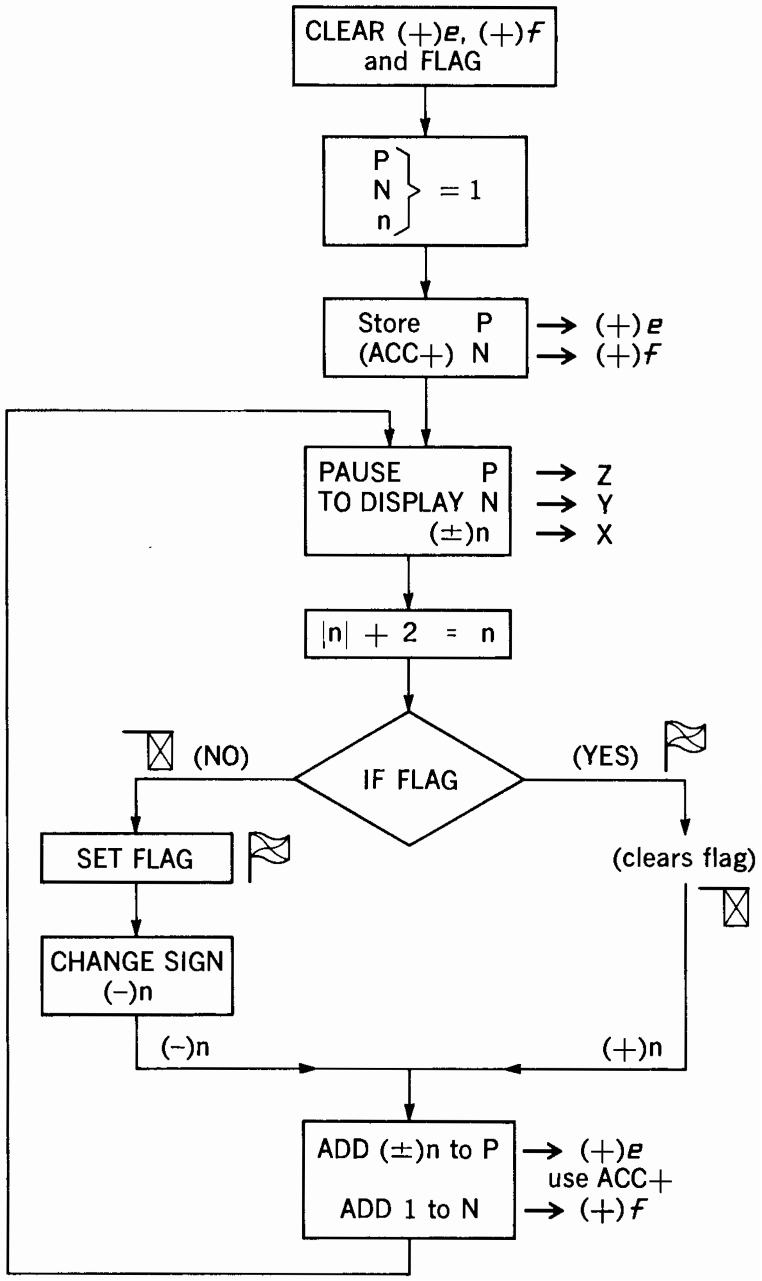
IF FLAG

SET FLAG

CONTINUED

FLOW CHART: for series $1 - 3 + 5 - 7 + 9 \cdots\cdots \infty$

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



PROGRAM KEYS – CONDITIONAL BRANCHING

To operate the program:

PRESS: END CONT

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

PRESS: PAUSE ; press 'CONT' to continue running the program.

To restart the program:

PRESS: PAUSE (or STOP) END CONT

		KEY		DISPLAY		STORAGE	
STEP	KEY	CODE	X	Y	Z	(+)*f*	(+)*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		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		
0*a*	+	33	2	n+2=n	N		
0*b*	↓	25	n	N	N		
0*c*	IF FLAG	43					
0*d*	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					
1*a*	0	00					
1*b*	5	05					

PROGRAM KEYS – SUBROUTINES

▲SUB▼ RETURN

▲SUB▼ RETURN Causes an unconditional branch to, execution of, and a return from, a subroutine (see also Page 60).

To call (branch to) a subroutine, the following program step sequence is used:

GO TO ( )( )   ▲SUB▼ RETURN   (+ or −)   Subroutine Starting Address

NOTE

Use of the (+) or (−) sign is page dependent; see GO TO on Page 68.

To return from a subroutine:

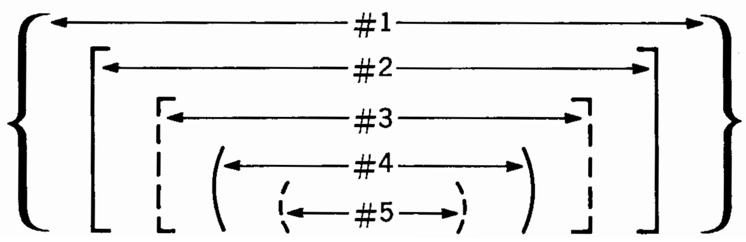
SUB/RETURN must be the last instruction of a subroutine. When the subroutine is completed, the program automatically returns to the address immediately following the last program step used to call the subroutine.

NOTE

When a GO TO instruction followed by SUB and an address is encountered in a program, the calculator 'remembers' the address to which it must return after the subroutine is completed: if the program counter is addressed from the keyboard to the starting address of a subroutine and the CONT key is pressed, then, after the subroutine is completed, the program will not return to the correct address.

NESTING SUBROUTINES: Subroutines can be 'called' during execution of a subroutine; this is known as 'nesting subroutines'. Any number of subroutines can be included in a program (limited only by program memory space); also any number of subroutines can be called during a subroutine provided that subroutines are 'nested' no more than five (5) deep at any one time during the program.

Subroutines nested five deep are shown in the illustration below; subroutine #2 is called during subroutine #1, then subroutine #3 is called during #2 and so on up to #5.



PROGRAM KEYS – SUBROUTINES

When a subroutine is called the calculator 'remembers' the return-address; up to five return-addresses can be 'remembered' at any one time. As soon as a return is made from a subroutine, the return-address of that subroutine is 'forgotten'; the calculator is now ready to remember another return-address.

In the preceeding illustration the number of subroutines called, during subroutine #1, is (apparently) limited to four (#2 through #5); this is because the subroutines (#1 through #5) are 'nested' to a depth of five. However, more subroutines may be called during subroutine #1. For example, after the return from #4, the return-address of #4 and #5 are 'forgotten'; there is now room for two more subroutines to be called.

In summary:

1. any number of subroutines can be used during a program;
2. any number of subroutines can be used during a subroutine, however,
3. no more than five subroutines can be used at any one time during a program.

The following is part of the program (on Page 97) which uses a subroutine.

STEP	KEY	STEP	KEY
.....	.....	.....	.....
(+)17	↓	(+)24	$x \rightarrow ( )$
18	GO TO ( )( )	.....	.....
19	▲SUB▼ / RETURN	38	PAUSE
1*a*	2	39	PAUSE
1*b*	4	3*a*	▲SUB▼ / RETURN
1*c*	$y \rightleftarrows ( )$	3*b*	.....

At step (+)1*b* the program branches to the subroutine starting at address (+)24 and executes the instructions of the subroutine; at step (+)3*a* the program branches back to address (+)1*c* and continues executing the program.

The following program illustrates the use of a subroutine; the N! program (Page 88), slightly altered, is used as the subroutine.

This 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 = 3,003

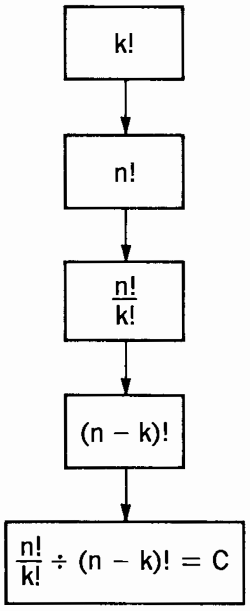
PROGRAM KEYS – SUBROUTINES

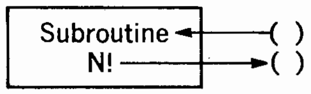
▲SUB▼ RETURN

CONTINUED

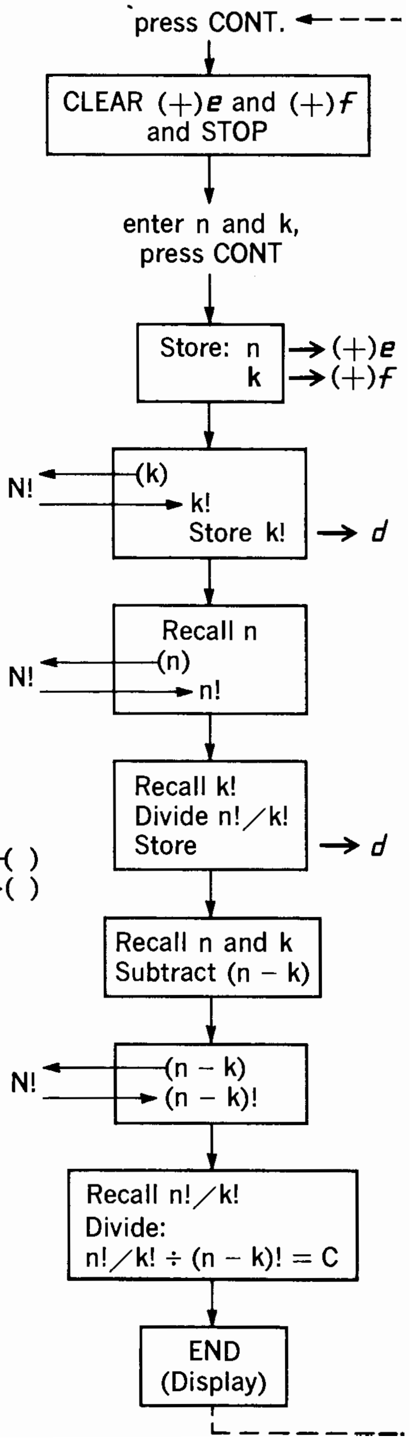
The formula used is: $C^n_k = \frac{n!}{k!\,(n-k)!}$

SIMPLIFIED FLOW CHART:





FINAL FLOW CHART:



PROGRAM KEYS – SUBROUTINES

To run the program; press 'END' (the first time only), CONT; then enter n and k and press CONT.

		KEY		DISPLAY	
STEP	KEY	CODE	X	Y	Z
(+)00	CLEAR	20	Enter		
01	STOP	41	k	n	
02	ACC +	60	k	n	
03	GO TO ( )( )	44			
04	▲SUB▼ / RETURN	77			
05	2	02			
06	4	04			
07	x→( )	23	k!		
08	*d*	17			
09	*e*	12	n		
0*a*	GO TO ( )( )	44			
0*b*	▲SUB▼ / RETURN	77			
0*c*	2	02			
0*d*	4	04			
(+)10	↑	27	n!	n!	
11	*d*	17	k!	n!	
12	÷	35	k!	n!/k!	
13	y→( )	40			
14	*d*	17			
15	RCL	61	k	n	
16	–	34	k	n – k	
17	↓	25	n – k		
18	GO TO ( )( )	44			
19	▲SUB▼ / RETURN	77			
1*a*	2	02			
1*b*	4	04			
1*c*	y⇄( )	24	(n–k)!		
1*d*	*d*	17	(n–k)!	n!/k!	
(+)20	÷	35		C	

CONTINUED

STORAGE

(+)*f*	Ø, k
(+)*e*	Ø, n
(+)*d*	~~k!~~, n!/k!
(+)*c*	N
(+)*b*	

PROGRAM KEYS – SUBROUTINES

▲SUB▼ RETURN

CONTINUED

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z
21	↑	27			C
22	RCL	61	k	n	C
23	END	46	k	n	C

final display

change to STOP (41) for recording on a magnetic card.

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z
(+)24	$x\rightarrow($ $)$	23	N		
25	*c*	16			
26	↑	27			
27	1	01			
28	↑	27			
29	ROLL ↓	31			
2*a*	IF $x>y$	53			
2*b*	3	03			
2*c*	6	06			
2*d*	ROLL ↓	31			
(+)30	×	36			
31	ROLL ↑	22			
32	−	34			
33	GO TO ( )( )	44			
34	2	02			
35	*a*	13			
36	*c*	16	N	0	N!
37	ROLL ↑	22	N!	N	0
38	PAUSE	57	flashing display		
39	PAUSE	57	N!	N	0
3*a*	▲SUB▼ / RETURN	77			

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.

Data stored on the (+) page requires less steps to store and recall than does data stored on the (–) page.

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.

1) Generally, start program steps at address (+)00 and sequentially fill the (+) page; then continue with address (–)00 and fill the (–) page.

2) Start data storage on the (+) page; fill register (+)*f* to (+)*a*, in that order. Next, use the (–)*f* and (–)*e* registers before storing data in the (+) numeric registers, (+)9, (+)8, · · · etc. Reserve the (+)*e* and (+)*f* registers, when applicable, for use with the ACC+, ACC– and RCL instructions.

3) If a program requires a complete page for data storage, start the steps at address (–)00 and reserve the (+) page for data storage.

4) 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.

5) '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 may be used, this will stop the card reader at the

ADDITIONAL PROGRAMMING

required point in the memory. Alternatively, reserve th((+) page for data storage only and write all program step: on the (–) page.

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' (CONT) 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 'CONT' 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;

succesive PRINT/SPACE instructions cause the printer to space after printing.

For the Model 9125A X-Y RECORDER make provision for a subroutine to be written to control the recorder. In this case insert five (5) successive 'CONT' instructions in the program, then when required, these can be replaced by GO TO, SUB, (+ or –), ( ), ( ) and the subroutine can be inserted in any available part of the memory.

STEP SAVING

ENTERING A CONSTANT

Following are more step-saving hints.

If a constant is required during a program this can be entered as program steps, thus saving part of a storage register. This is particularly valuable when used in a subroutine.

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

after step 26, 2.13 will be in the X register.

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.

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.

STEP SAVING

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

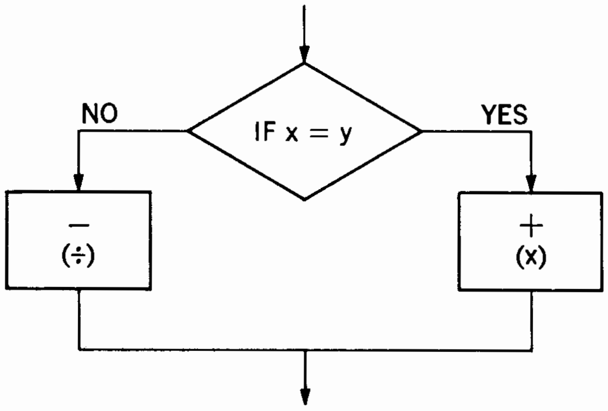
To calculate the square root of the sum of two squares ($\sqrt{A^2 + B^2}$):

	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 co-ordinates, the angle (θ) in either degrees or radians, appears in the Y register.

CONDITIONAL OPERATIONS

To conditionally add or subtract (multiply or divide):



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

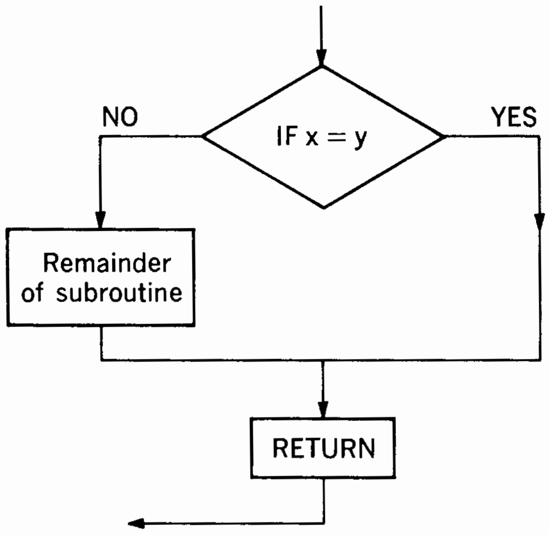
if x = y, steps 63 through 65 result in a net addition (multipli-cation);

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

The plus (+) and (−) signs cannot be used as they constitute the start of a branching address.

STEP SAVING

To make a conditional return from a subroutine:



STEP	KEY
. . .	. . . .
75	IF $x = y$
76	9
77	2
78	. . . .
. . .	. . . .
92	▲SUB▼ / RETURN

APPENDIX

RANGE OF ARGUMENTS

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

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

Range of the 9100B $9.999\ 999\ 999 \times 10^{99}$ 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.

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

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

$f(X)$ = exact value

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

CONTINUED

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(+, −)		$\pm 5 \times 10^{-11} \times 10^{(\text{larger addend exponent})}$
II.	MPY		$\pm 1/2$ visible count
III.	DIV		$\pm 1/2$ visible count
IV.	SQRT		$\pm 1/12$ visible count
V.	COS θ	$\theta < 1$ rad	$\pm 4 \times 10^{-11}$
		1 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 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	$\pm 1\ 1/2$ visible count
		1 rad $\leq \theta < \pi$	$\pm 1/2$ (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	$\pm 1/10$ visible count
		degrees	$\pm 1/4$ visible count
X.	SIN^{-1} (a)	a$<$ .707	$\pm 1/2$ visible count
XI.	COS^{-1} (a)	a$<$ .707	$\pm 1/2$ visible count $\pm 10^{-11}$
XII.	SIN^{-1} (a)	a$>$ .707	rad $\pm 10^{-10}$ deg $\pm 70 \times 10^{-10}$
XIII.	COS^{-1} (a)	a$>$ .707	rad $\pm 10^{-10}$ deg $\pm 70 \times 10^{-10}$

KEY CODES

$\sqrt{x}$	CHG SIGN	ENTER EXP	CLEAR x	CLEAR	IF FLAG	SET FLAG
76	32	26	37	20	43	54
÷	7	8	9	FMT	IF $x < y$	PAUSE
35	07	10	11	42	52	57
×	4	5	6	PRINT / SPACE	IF $x = y$	STOP
36	04	05	06	45	50	41
−	1	2	3	▲SUB▼ / RETURN	IF $x > y$	END
34	01	02	03	77	53	46
+	0	•	π	CONT	GO TO ( )( )	STEP PRGM
33	00	21	56	47	44	

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.

TO POLAR	$\lvert y \rvert$	arc ▼	a	b	$x \rightleftarrows y$
62	55	72	13	14	30
TO RECT	int x	hyper ▼	c	d	ROLL ↓
66	64	67	16	17	31
RCL	e^x	sin x	e	f	↑ ROLL
61	74	70	12	15	22
ACC −	ln x	cos x	$y \rightarrow (\ )$	$y \rightleftarrows (\ )$	↓
63	65	73	40	24	25
ACC +	log x	tan x	$x \rightarrow (\ )$	$x \leftarrow (\ )$	↑
60	75	71	23	67	27

PART NO. 09100-90021
SEP 1969

PRINTED IN U.S.