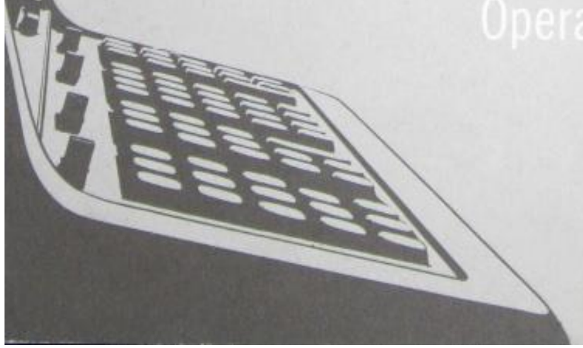
HEWLETT-PACKARD CALCULATOR EXTENDED MEMORY

Model 9101A

Operating Manual



HEWLETT hp PACKARD

WARRANTY AND ASSISTANCE

This Hewlett-Packard instrument is warranted against defects in material and workmanship. This warranty applies for one year from date of delivery. We will repair or replace components which prove to be defective during the warranty period. No other warranty is expressed or implied. We are not liable for consequential damages.

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

OPERATING MANUAL

HEWLETT-PACKARD 9101A

TABLE OF CONTENTS

TABLE OF CONTENTS

PREFACE

This manual contains the information required to operate a 9101A Extended Memory. Instructions concerning the operation of the calculator are mentioned only when they affect the 9101A or in the examples of 9101A operation. This manual does not attempt to teach calculator operation and assumes the operator is already familiar with the operation of his calculator. For calculator operating information, please consult the calculator operating manual supplied with the calculator.

DESCRIPTION

The 9101A Extended Memory attaches to the Model 9100 (A or B) Calculator and provides 248 additional registers of memory. Similar in configuration to the registers in the calculator, these registers can accommodate either 3,472 program steps or 248 constants or a combination of both (14 program steps being used for each constant stored).

The usefulness of the 9101A is further extended by its indirect arithmetic capability; in this mode, the contents of the Y register can be added to, subtracted from, multiplied by, or divided into the contents of any 9101A register specified by the contents of the X register, the result of each operation is automatically stored in the 9101A.

The 9101A also has subprogram capability. In this mode the program in operation calls another program. When that program has finished, control is returned to the calling program. This feature of the 9101A gives the 9100A a capability similar to the subroutine feature of the 9100B.

CAUTION
CONSULT THE TURN-ON INSTRUCTIONS PRIOR TO APPLYING POWER TO THE 9101A.

ACCESSORIES EQUIPMENT SUPPLIED

The accessories and equipment supplied with each model 9101A are listed in Table 1.

TABLE 1
Accessories/Equipment supplied

PART NO.	QUANTITY	DESCRIPTION
09101-90001	2	Operating Manual
09101-90002	1	Program Library
09101-90003	1	Memory Map Pad
09101-90004	1	Magnetic Program Card containing the Diagnostic Program.
8120-1378	1	Power Cord
2140-0092	1	Lamp 5V .06A
2110-0312	1	Fuse 1A 250V
2110-0202*	2	Fuse .5A 250V

*Used for 230 line voltage operation.

PROGRAM LIBRARY

The program library (listed in Table 1) furnished with the 9101A Extended Memory contains general purpose programs in many disciplines. It serves as both a source of ready-to-use programs and as an illustration of programming techniques.

9101A OPTIONS

Extended Memories may be purchased with the instruction card printed in languages other than English.

9101A, Standard:	Card Printed in English
9101A, Option 001:	Card Printed in French
9101A, Option 002:	Card Printed in German
9101A, Option 003:	Card Printed in Italian
9101A, Option 004:	Card Printed in Spanish

OTHER 9100 PERIPHERALS

The 9101A is one of the many peripheral devices available for the 9100 series Calculator. The following is a list of some of the other peripheral devices. They lend great versatility to the 9100 series Calculator and may be added at any time.

Model 9120A PRINTER: Provides fast, quiet Printer capability for use with the 9100 Calculator. The printer prints the contents of any combination of the X, Y and Z Calculator display registers upon command. Quiet operation is obtained using a unique electrosensitive printing principle. The 9120A Printer mounts on top of the 9100 Calculator to ensure easy access and minimum space requirements. Operation of the Printer is initiated either manually by a single keystroke or in a program by one program step.

Model 9125A PLOTTER: Provides permanent graphic solutions to problems solved by the Calculator. The Plotter plots the point, specified by the numbers in the Calculator's X and Y registers, when the format (FMT) instruction is activated. The relationship between the variables can be programmed in the calculator. The Calculator can also be used in the manual mode to transfer data coordinates directly to the Plotter.

Model 9150A DISPLAY: A large screen display of the 9100 X, Y and Z registers, which allows a large group to see the calculations. Instructors find this peripheral exceptionally valuable as a visual aid while explaining scientific concepts. The 9150A requires that a modification be made to the calculator.

Model 9160A MARKED CARD READER: Optically reads cards marked with a soft lead pencil. By using properly marked cards, programs and numerical data can be entered quickly and conveniently into a calculator.

Model 9102A CALCULATOR BUFFER: Allows other peripherals to be used with the 9101A or 2570A. It also allows greater cable length (5') to be used between peripherals.

SERVICE CONTRACTS

Service Contracts are available for the Model 9101A Extended Memory. These contracts guarantee a fixed maintenance cost for the customer. 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 Extended Memory 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 the Extended Memory for physical damage caused in transit and check for the accessories listed in Table 1. Also, check the electrical performance of the memory as described under ELECTRICAL INSPECTION; do not, however, make this check prior to completing the INSTALLATION and TURN-ON PROCEDURE sections.

POWER REQUIREMENTS

The Model 9101A Extended Memory requires either 115V or 230V ac, ±10%, 48 to 66 Hertz; power requirements are less than 95 voltamps. A slide switch, located on the rear of the instrument, selects either 115 or 230V operation. A fuse change is required for 230 line voltage operation.

GROUNDING REQUIREMENTS

To protect operating personnel, the NATIONAL ELECTRICAL MANUFACTURERS' ASSOCIATION (NEMA) recommends that the Extended Memory panel and cabinet be grounded. The 9101A is equipped with a three-conductor power cable which, when plugged into an appropriate receptacle, grounds the panel and cabinet of the memory. The center pin on the power cable connector is the ground connection.

INSTALLATION

The following figure shows connection of the 9101A in your system. Connect the 9101A signal connector as shown in the appropriate example.

> **CAUTION**
>
> **TURN THE CALCULATOR OFF PRIOR TO MAKING ANY CONNECTIONS TO ITS REAR CONNECTOR.**

> **CAUTION**
>
> **DO NOT APPLY OPERATING POWER TO THE 9101A UNLESS THE LINE SWITCH ON THE REAR PANEL IS IN THE PROPER POSITION; OTHERWISE, DAMAGE TO THE POWER TRANSFORMER MAY RESULT.**

INSTALLATION
CONTINUED

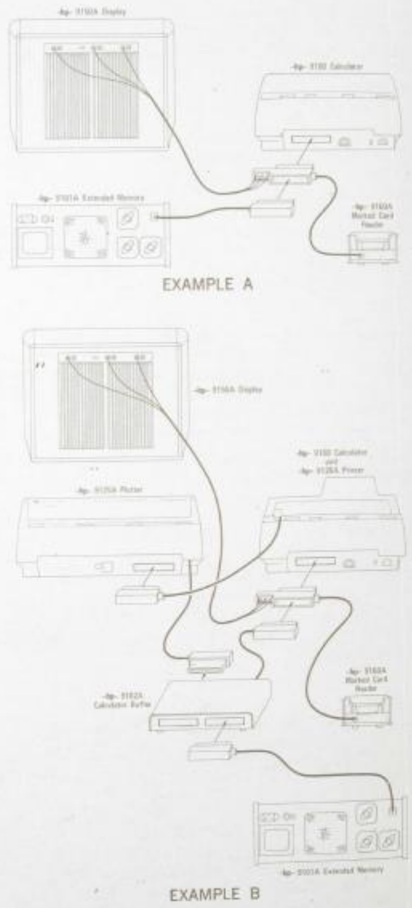


Figure 1. Possible 9100 Series Systems Configuration

RACK MOUNTING

The Model 9101A may be rack mounted by using the 7" rack mount kit (-hp- Part No. 5060-8741). Instructions are included with the kit. The rack mount for the 9101A is an EIA standard width of 19". When mounted in a rack using the rack mount kit, additional support should be provided at the rear of the instrument.

TURN-ON PROCEDURE

With the Extended Memory disconnected from the a.c. power source, slide the line voltage switch, located to the right of the fuse on the rear panel, to the position where the line voltage to be used (115 or 230) appears on the switch. IF 230 LINE VOLTAGE IS TO BE USED, REPLACE THE 1A FUSE SHIPPED IN THE INSTRUMENT WITH THE .5A FUSE SHIPPED AS AN ACCESSORY. Connect the Extended Memory to the a.c. power source, switch the Calculator OFF-POWER ON switch, located above the keyboard, to the POWER ON position and switch the 9101A's LINE switch to the ON position. The Extended Memory's LINE lamp will light, indicating that power is applied to the 9101A.

NOTE

If the lamp fails to light, the fuse or lamp may be defective. See the operator maintenance section. Check the fuse first and then the lamp.

ELECTRICAL INSPECTION

A magnetic program card, containing the diagnostic Program, is provided with the 9101A Extended Memory; this program tests the electrical performance of the memory. (The steps of the Diagnostic Program are included in the Program Library).

To exercise the program:

set the switches on the calculator and memory to the following positions:

FIXED POINT   POWER ON

RUN   FILE PROTECT OFF

Rotate the Decimal Digits wheel until 3 appears.

3

ELECTRICAL INSPECTION
CONTINUED

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

PRESS: CLEAR FMT SET FLAG END

Insert the diagnostic card into the magnetic card-reader with the A arrow pointing down.

PRESS: ENTER

PRESS: 1 FMT FMT END

DISPLAY:
0. → z
0. → y
10. → x

Insert the diagnostic card into the magnetic card-reader with the B arrow pointing down.

PRESS: ENTER

PRESS: 2 FMT FMT

DISPLAY:
0. → z
0. → y
21. → x

PRESS: 1 FMT GO TO (II) CONT

Correct operation of the 9101A is indicated by the display shown below flashing on the display screen. The numbers will flash at a much slower rate than the calculator diagnostic. The Extended Memory has been completely checked when the numbers cycle from one to twelve.

N.
N.
N.

N = 1, 2, 3 . . . . 12, 1; Cyclic

To stop the program press STOP. To restart the program:

PRESS: 2 2 FMT SET FLAG

PRESS: 1 FMT GO TO ( )( ) CONT

If the Extended Memory will not operate properly check the fuse on the rear panel (See the operator maintenance section). If the fuse is O.K., consult the warranty information on the inside front cover of this manual.

INTRODUCTION

This section introduces the general operating characteristics of the Model 9101A Extended Memory. This section is not intended to teach operating specifics (covered in the KEYBOARD and PROGRAMMING sections) but rather to give the operator a general background of instrument operation and terminology.

MEMORY CONSTRUCTION

The 9101A contains 248 registers numbered 0 through 247. Program steps or constants may be stored in these registers. The registers are similar to those in the Calculator, except that individual characters within a register (eg. $S\,7$) cannot be addressed in the 9101A.

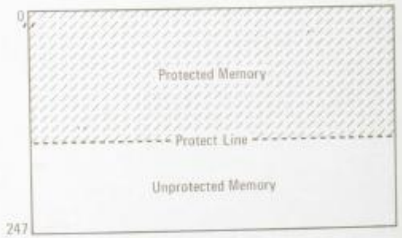


Figure 2. 9101A Memory Map

PROTECT LINE

The 9101A memory has a protect line. The contents of the memory above this line are protected against destruction; the contents may be read from the protected memory, but new material cannot be written over it. The protect line moves automatically to protect programs being stored in the 9101A memory or may be manually moved by the operator to protect previously stored data.

DATA TRANSFER

Data must originally be stored in the unprotected region of the 9101A memory. Data cannot be stored in the protected region of the memory. If an effort is made to store data in the protected memory, the IMPROPER FORMAT lamp will light and a diagnostic code will appear in the X register informing the operator of an illegal operation. The operator may protect data by storing the data in the unprotected region of the 9101A memory and then moving the protect line.

INDIRECT ARITHMETIC

The contents of the Y register (operator) may be added to, subtracted from, multiplied by, or divided into any specified 9101A register (operand). If the operation is performed on an unprotected 9101A register, the result of the operation will be stored in the specified register. If the specified 9101A register is protected, the operation will occur and the result will appear in the Y register; however, the contents of the 9101A register will remain unchanged. As this is an illegal operation, the IMPROPER FORMAT lamp will light and the X register will contain a diagnostic code..

PROGRAM TRANSFER

A program must be resident in the 9100 memory before it can be transferred into the 9101A memory and have a program number assigned to it. The 9101A can store up to 100 programs (0 - 99); they need not be numbered sequentially. Programs that are too long to be contained in the 9100 memory must be divided into segments which can be contained in the calculator's memory and numbered, so that they can be identified later. The program must start at location 00 (+00 in the 9100B) and must terminate with an END statement. Failure to properly terminate the program will result in the program being read into the 9101A repeatedly until the 9101A memory is completely filled. At that time the IMPROPER FORMAT lamp will light and a diagnostic code will appear in the X register notifying the operator that the available 9101A memory has been exceeded.

NOTE
Programs must terminate with an END statement.

Two basic approaches for storing programs in the 9101A exist: A sequential program technique and a subprogram technique.

PROGRAM TRANSFER CONTINUED

SEQUENTIAL PROGRAM TECHNIQUE

In the sequential program technique (see Figure 3) programs are assigned numbers and are stored in the 9101A memory. In the example shown in Figure 3, program number one is called into the calculator memory and started (the program is not destroyed, however, and is now resident in the 9100 and 9101A.) When program number one is finished, it calls program number two, which is written over program number one in the 9100 memory. When program number two is finished, it calls program number three. The END statement in program number three stops the program.

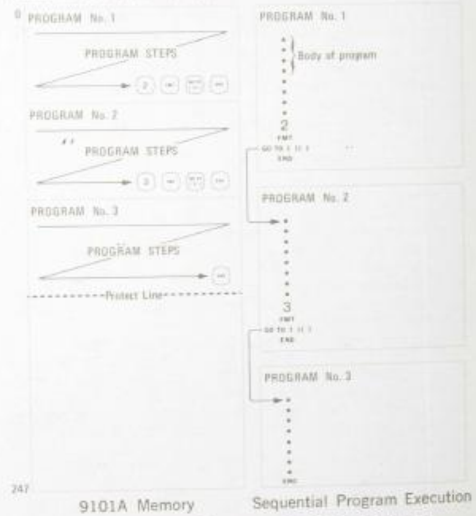
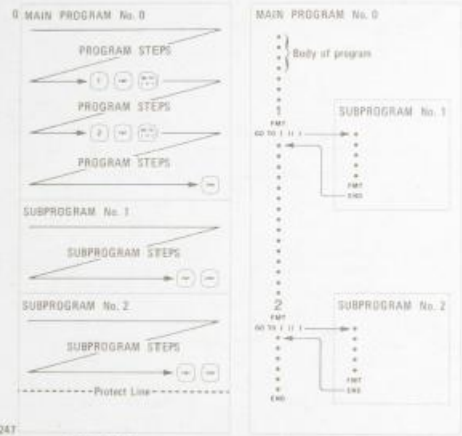


Figure 3. Sequential program storage and Execution

In the preceding example, the programs were of approximately the same length and numbered sequentially. It is important to note that the programs could have been quite different in length and numbered anything (as long as they were uniquely numbered) between zero and ninety-nine. The programs were all resident in the protected area of the 9101A memory because the protect line moved automatically to protect them as they were stored; programs cannot be called into the 9100 memory if they reside outside the protected memory.

SUBPROGRAM TECHNIQUE

There are many variations of this technique: in the example, shown in Figure 4, a main program calls subprograms (similar to the subroutine capability of the 9100B). Often the main program is shorter than any of the subprograms.



9101A Memory Subprogram Execution

Figure 4. Subprogram storage and execution

In this example, the main program begins the operation and, after a time, calls subprogram number one. Subprogram number one, when finished, recalls the main program and execution of the main program is resumed one step after the step that called subprogram number one. Execution of the main program continues until it calls subprogram number two. When subprogram number two is finished, it recalls the main program; again, operation is resumed one step following the step that called subprogram number two. The main program will continue until the END statement is encountered, which stops the sequence of events.

NOTE

Generally, all programs must be resident in the 9101A (see PROGRAM TRANSFER SUMMARY for exceptions).

PROGRAM TRANSFER
CONTINUED

Subprograms can call subprograms. This is known as "nesting subprograms". Subprograms can be nested fourteen deep; that is, subprograms calling subprograms until the "nest" is fourteen programs deep.

Figure 5. Subprogram nesting

Figure 6 illustrates the principle of subprogram nesting. Although this example illustrates the nesting of only three subprograms, the same principle can be easily expanded to fourteen (the capability of the 9101A). In the illustration, subprogram one calls subprogram two, which calls subprogram three. When subprogram three has been executed, it recalls two which recalls one.

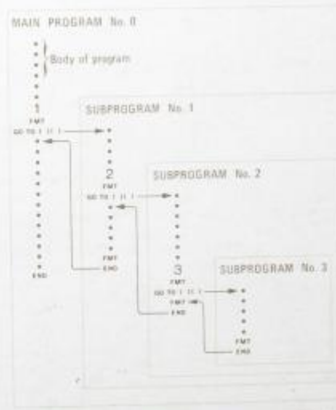


Figure 6. Subprogram nesting execution

Figure 6 shows subprogram execution. The example nests subprograms three deep. In the execution sequence the path followed during nesting is retraced during unnesting. Although the example shows one calling two and two calling three, subprogram one could call sixty-three and sixty-three could call subprogram four. Program numbers must not be confused with the ability of the 9101A to nest fourteen programs.

PROGRAM TRANSFER SUMMARY

Generally all programs, whether used as a sequential program segment, as a main program or as a subprogram are stored in the 9101A prior to execution. This allows programs to be reused without re-entry. However, as programs become more difficult, the operator has at his option certain techniques for using the calculator's memory in addition to the 9101A memory. For example, in the sequential program technique the first program segment could be stored only in the calculator memory. When the second segment is called from the 9101A it will be written over the first, destroying it.

In the subprogram technique, the main program can be stored in the higher number calculator registers and only the subprograms stored in the 9101A. (In the 9100B the main program could be stored on the minus page with the plus page reserved for subprograms.) However, this will work only if any subprogram recalled does not require the part of the calculator memory space occupied by the main program.

All 9101A programs will not be complicated. Many will be relatively simple using large amounts of memory for data storage only.

INTRODUCTION

This section describes the 9101A Extended Memory commands. The commands are described in two methods — a brief explanation in bold print (provided for quick reference) and a more detailed explanation with examples. The 9101A diagnostic codes are also presented in two ways, a chart on page 16 and detailed explanations included with the examples under each command.

NOTATION

N refers to the first unprotected register in the 9101A. The protected area of the 9101A memory always starts at register zero. Counting from memory location 0 in the 9101A toward memory location 247, the protect line is located between N and N-1. N-1 is the last protected register.

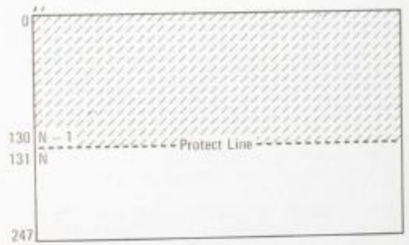


Figure 7. Identification of N and N-1 in the 9101A.

Figure 7 shows the location of the protect line between registers 130 and 131. In this example N would be 131 and N-1 would be 130.

X, Y and Z refer to the calculator's display registers. X is the keyboard register, Y is the accumulator register and Z is the temporary register.

x, y and z refer to the contents of the calculator's display registers.

P_x refers to the program specified by x (two digit maximum). Programs stored in the 9101A must be assigned a program number. This number is contained in the X register when the program is transferred or recalled and must be numbered between zero and ninty-nine (two digit maximum).

R_x refers to the 9101A register specified by x (three digit maximum ≤ 247). There are 248 registers in the 9101A (0 through 247). A specific 9101A register is identified by x.

r_x refers to the contents of register R_x. Since R_x is a 9101A register specified by x, then r_x is the contents of a 9101A register which is specified by x.

KEYING INSTRUCTIONS

In the examples, the keying instructions will be given in the following format:

STEP	KEY	KEY CODE
1	CLEAR	20
2	2	02
3	ENTER EXP	26
4	CHG SIGN	32
5	5	05
6	0	00
7	↑	27

in which keys are pressed in STEP sequence 1, 2, 3, 4, etc.

SWITCHING INSTRUCTIONS

SWITCH: **RUN**

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

FRONT PANEL

LINE

Applies line power to the Extended Memory. When power is applied to the 9101A, the LINE lamp will be lit. If the calculator is OFF, the 9101A is in an inoperative standby mode.

> **NOTE**
>
> Power requirements and Turn-on instructions are contained in the General Information section.

FILE PROTECT

When the FILE PROTECT switch is ON, the protect line **CANNOT** be moved manually (explained under FMT SET FLAG). The protect line still moves automatically to protect programs being stored in the 9101A regardless of the switch's position.

IMPROPER FORMAT

Lights when an illegal 9101A operation is attempted (eg. a manual attempt at moving the protect line when the FILE PROTECT is turned ON). Whenever the IMPROPER FORMAT lamp lights, a diagnostic code appears in the X register and program execution is stopped. The lamp will reset when the STOP or FMT instruction is given or when either the 9100 or 9101A is switched OFF.

The following is a list of diagnostic codes. One of these codes will appear in the X register when the IMPROPER FORMAT lamp lights. The list contains the diagnostic code, the explanation of why it lit and a list of FMT commands (one of which caused the illegal operation).

● IMPROPER FORMAT ~ DIAGNOSTIC CODE IN X REGISTER

CODE	EXPLANATION	FMT COMMAND
1.111 111 111 15	Attempt to store in protected register R_x.	y→( )
2.222 222 222 15	Attempt to alter contents of protected register R_x	+, −, ×, ÷
3.333 333 333 15	$R_x > 247$.	+, −, ×, ÷, π, y→( ), SET FLAG
4.444 444 444 15	P_x store incomplete: memory exceeded.	FMT
5.555 555 555 14	P_x recall incomplete: only protected part recalled.	GO TO ( ) END
5.555 555 555 15	Attempt to recall unprotected program P_x.	GO TO ( ) END
6.666 666 666 15	FILE PROTECT switch on.	SET FLAG
7.777 777 777 13	Over 3 digits in R_x.	+, −, ×, ÷, π, y→( ), SET FLAG
7.777 777 777.14	Over 2 digits in P_x	FMT, GO TO

These diagnostic codes will be presented again with the examples under the Extended Memory commands.

COMMAND SET

Each Extended Memory command consists of two instructions: FMT (format) and another key which defines the operation. The command set is easily divided into four groups: Data Transfer, Indirect Arithmetic, Program Transfer and Subprograms. The one exception to the preceding grouping is the FMT SET FLAG command which moves the PROTECT LINE. It is included, for convenience, under Program Transfer on the 9101A instruction card but is, in fact, a "stand alone" instruction, since all instructions require knowledge of the PROTECT LINE and an understanding of how it may be moved.

DESIGNATING A 9101A REGISTER

Most of the Extended Memory commands require that a particular 9101A register be designated.

A specific register is designated by the contents of the X register. Excluding leading zeros, which are always ignored, the X register must contain no more than three digits and may be in fixed or floating display. Signs which may be present in the display are ignored. The exponent of the display is also ignored. For example, each of the following displays contained in the X register would specify register 49:

.000049		→ x
4.9	78	→ x
-4.9	78	→ x
4.9	-78	→ x
4.9	-09	→ x

49.0 would not designate register 49, instead, the 9101A would interpret this number as 490. This situation often occurs in computing a register address (i.e. 147÷3 = 49.0 . . .) in which case simply follow the computation with INT X.

FMT SET FLAG

N = x The protect line moves to N, protecting all lower numbered registers. If the FILE PROTECT is ON the command is ignored and a diagnostic code (6.666 666 666 15) appears in the X register notifying the operator that the FILE PROTECT switch is ON.

EXAMPLE:

SWITCH: RUN FILE PROTECT **ON**

PRESS: 1 FMT SET FLAG

DISPLAY: 6.666 666 666 15 → x

CONTINUED

COMMAND SET

FMT SET FLAG

CONTINUED

The IMPROPER FORMAT lamp will light and the diagnostic code will appear indicating the FILE PROTECT switch is ON.

SWITCH: **RUN** FILE PROTECT **OFF**

PRESS: 1 0 0 FMT SET FLAG

The protected region of the memory now extends to register 100. Register 99 (N-1) is protected; register 100 (N) is not.

NOTE

The only way that register 247 can be protected with the FMT SET FLAG is to set N = 248. This will cause the IMPROPER FORMAT lamp to light and the diagnostic code 3.333 333 333 15 to appear in X.

N-1 PROGRAM

The following program locates the last protected register (N-1) and displays its contents. Key it into the calculator memory, starting at location 00 (+00 in the 9100B).

	STEP	KEY	KEY CODE	STEP	KEY	KEY CODE
(+)	00	CLEAR	20	0a	FMT	42
	01	2	02	0b	+	33
	02	4	04	0c	IF x = y	50
	03	8	10	0d	CLEAR	20
	04	ACC +	60	10	STOP	41
	05	1	01	11	GO TO ( ) ( )	44
	06	ACC −	63	12	0	00
	07	f	15	13	5	05
	08	↑	27	14	END	46
	09	ROLL ↓	31			

SWITCH: **RUN FLOATING**

PRESS: END CONT

COMMAND SET

DISPLAY: 9.9 01 → z
Contents of Register 99 → y
2.222 222 222 15 → x

(Register 99 was the last register protected using the example under FMT SET FLAG).

The diagnostic code in the X register indicates that the register identified by the contents of Z is protected. The program does not alter the contents of any 9101A register.

Exception:

The display will change if N = 0.

EXAMPLE:

PRESS: 0 FMT SET FLAG

There are no protected registers in the 9101A because N = 0.

PRESS: END CONT

DISPLAY: 0. 00 → z
0. 00 → y
0. 00 → x

This display indicates there are no protected registers in the 9101A.

COMMAND SET – DATA TRANSFER

FMT y→( )

Y→R. This instruction stores the contents of the Y register in a 9101A register designated by the contents of the X register. The old contents of the designated 9101A register are destroyed. The X and Y registers are not changed. Data CANNOT be transferred to a protected register.

EXAMPLE:

SWITCH: RUN FLOATING

SWITCH: FILE PROTECT OFF

PRESS: CLEAR FMT SET FLAG

These keystrokes will move the protect line so that it does not interfere with the following examples. Then:

PRESS: 1 0 ↑

PRESS: 5 0 FMT y→( )

The number 10 has been stored in the 50 register. To prove that the value is indeed stored:

PRESS: CLEAR 5 0 FMT π

NOTE

FMT π is a recall instruction.

DISPLAY: 1.0 01 → x

The 50 register is not protected. New data can be written over the old, destroying it:

PRESS: 1 2 3 ↑

PRESS: 5 0 FMT y→( )

COMMAND SET – DATA TRANSFER

To prove the value for 10 has been destroyed:

PRESS: FMT π

DISPLAY: 1.23 02 → X

Data **CANNOT** be stored in a 9101A register that is protected.

PRESS: 5 0 FMT SET FLAG

These keystrokes will move the protect line to register 50. N-1 (register 49) is protected.

PRESS: 3 0 ↑

PRESS: 4 9 FMT y→( )

DISPLAY: 1.111 111 111 15 → X

A diagnostic code appears in the X register indicating R_x is protected.

Data may be desguised as a program and stored in the 9101A. To do this, store the constants in the calculator memory (one per register). After the last constant has been placed in the calculator memory, store an END statement. This entire block of data may be transferred to the 9101A using the FMT FMT command.

FMT π

X←r. This instruction recalls to the X register the contents of a 9101A register designated by the contents of X. The Y and Z registers are not changed. The content of the recalled register is not destroyed.

EXAMPLE:

SWITCH: **RUN FLOATING**

SWITCH: FILE PROTECT **OFF**

CONTINUED

COMMAND SET – DATA TRANSFER

FMT π

CONTINUED

PRESS: 5 0 FMT SET FLAG

These keystrokes will move the protect line so that it does not interfere with the following example.

PRESS: 1 2 ↑

PRESS: 5 0 FMT y→( )

These keystrokes stored 12 in the 50 register. To recall 12:

PRESS: CLEAR 5 0 FMT π

DISPLAY: 1.2 01 → X

The content of register 50 is not destroyed and is available to be recalled any number of times. However, the register is not protected (N-1 = 49) and may therefore be changed by subsequent data storage. If the data is to be kept for future operations it is advisable to protect the data (using the protect line) from accidental loss. As will be explained later, programs are transferred into the 9101A starting immediately below the protect line; if needed data is unprotected, there is a possibility that a program could be written over it.

COMMAND SET – INDIRECT ARITHMETIC

FMT +

$r_x + y \to R_x$ Adds the contents of the Y register to the contents of a 9101A register specified by the contents of X. The sum is stored in R_x; the X, Y and Z registers are unchanged. The operation CANNOT be performed on a protected register. If the attempt is made, the program is stopped and a diagnostic code appears in the X register; the sum of y and r_x appears in the Y register.

EXAMPLE:

SWITCH: RUN FLOATING

SWITCH: FILE PROTECT OFF

PRESS: CLEAR FMT SET FLAG

These instructions move the protect line so that it will not interfere with the following examples.

PRESS: 1 2 ↑

PRESS: 5 0 FMT y→( )

The number 12 is stored in the 50 register.

PRESS: FMT +

The 12 in the Y register has been added to the 12 in the 50 register. The 50 register now contains 24.

PRESS: FMT π

DISPLAY: 2.4 01 → x

Numbers cannot be added to a protect register.

PRESS: 6 0 FMT SET FLAG

CONTINUED

COMMAND SET – INDIRECT ARITHMETIC

FMT +

CONTINUED

Register 50 is now in the protected region of the memory.

PRESS: 5 0 FMT +

DISPLAY: 3.6 01 → y
2.222 222 222 15 → x

The sum of the contents of the Y register and register 50 appears in Y. However, register 50 still contains 24 since the sum of the two numbers was not stored in the 50 register.

FMT −

$r_x - y \to R_x$ Subtracts the contents of the Y register from the contents of a 9101A register specified by the contents of X. The difference is stored in R_x; the X, Y and Z registers are unchanged. The operation CANNOT be performed on a protected register. If the attempt is made, the program is stopped and a diagnostic code appears in the X register; the difference of y and r_x appears in the Y register.

EXAMPLE:

SWITCH: RUN FLOATING

SWITCH: FILE PROTECT OFF

PRESS: CLEAR FMT SET FLAG

These instructions move the protect line so that it will not interfere with the following examples.

PRESS: 3 6 ↑

PRESS: 5 0 FMT y→( )

The number 36 is stored in the 50 register.

PRESS: x⇄y 1 2 x⇄y

PRESS: FMT −

COMMAND SET – INDIRECT ARITHMETIC

The 12 in the Y register has been subtracted from the 50 register. The 50 register now contains 24.

PRESS: FMT π

DISPLAY: 2.4 01 → x

Numbers cannot be subtracted from a protected register.

PRESS: 6 0 FMT SET FLAG

Register 50 is now in the protected region of the memory.

PRESS: 5 0 FMT –

DISPLAY: 1.2 01 → y
2.222 222 222 15 → x

The difference between the contents of the Y register and register 50 appear in Y. However, register 50 contains 24 since the difference of the two numbers was not stored in the 50 register.

FMT ×

$r_i \times y \to R_i$ Multiplies the contents of the Y register by the contents of a 9101A register specified by the contents of X. The product is stored in R_i; the X, Y and Z registers are unchanged. The operation cannot be performed on a protected register. If the attempt is made, the program is stopped and a diagnostic code appears in the X register; the product of y and r_i appears in the Y register.

EXAMPLE:

SWITCH: RUN FLOATING

SWITCH: FILE PROTECT OFF

PRESS: CLEAR FMT SET FLAG

These instructions move the protect line so that it will not interfere with the following examples.

COMMAND SET – INDIRECT ARITHMETIC

FMT ×
CONTINUED

PRESS: 1 2 ↑

PRESS: 5 0 FMT y→()

The number 12 is stored in the 50 register.

PRESS: FMT ×

The 12 in the Y register has been multiplied by the 12 in register 50. The 50 register now contains 144.

PRESS: FMT π

DISPLAY: 1.44 02 → x

Numbers cannot be multiplied by a protected register.

PRESS: 6 0 FMT SET FLAG

Register 50 is now in the protected region of the memory.

PRESS: 5 0 FMT ×

DISPLAY: 1.728 03 → y
2.222 222 222 15 → x

The product of the contents of the Y register and register 50 appears in Y. However, register 50 contains 144 since the product of the two numbers was not stored in the 50 register.

CLEARING THE 9101A REGISTERS

Since $i \times 0 = 0$, the multiplication ability of the indirect arithmetic provides a quick, easy method of clearing unprotected 9101A registers. Here is the program, key it into the calculator's memory starting at memory location 00 (+00 in the 9100B).

COMMAND SET – INDIRECT ARITHMETIC

	STEP	KEY	KEY CODE	STEP	KEY	KEY CODE
(+)	00	CLEAR	20	07	f	15
	01	2	02	08	FMT	42
	02	4	04	09	×	36
	03	8	10	0a	GO TO ( )( )	44
	04	ACC +	60	0b	0	00
	05	1	01	0c	5	05
	06	ACC –	63	0d	END	46

The program will continue until it encounters a protected 9101A register, at that time the program will stop and a diagnostic code (2.222 222 222 15) will appear in the X register.

To clear the entire 9101A memory, set the protect line to register 0 (N = 0) and run the program. When the memory has been cleared the program will stop and a diagnostic code (3.333 333 333 15) will appear in the X register.

FMT ÷

$r_i \div y \rightarrow R_i$ Divides the contents of the Y register into the contents of a 9101A register specified by the contents of X. The quotient is stored in R_i; the X, Y and Z registers are unchanged. The operation CANNOT be performed on a protected register. If the attempt is made, the program is stopped and a diagnostic code appears in the X register; the quotient of y and r_i appears in the Y register.

EXAMPLE:

SWITCH: RUN FLOATING

SWITCH: FILE PROTECT OFF

PRESS: CLEAR FMT SET FLAG

These instructions move the protect line so that it will not interfere with the following examples.

PRESS: 1 4 4 ↑

PRESS: 5 0 FMT y→( )

The number 144 is stored in the 50 register.

CONTINUED

COMMAND SET – INDIRECT ARITHMETIC

FMT ÷

CONTINUED

PRESS: x⇄y 1 2 x⇄y

PRESS: FMT ÷

The 12 in the Y register has been divided into the 144 in register 50. The 50 register now contains 12.

PRESS: FMT π

DISPLAY: 1.200 000 000 01 → x

Numbers cannot be divided into a protected register.

PRESS: 6 0 FMT SET FLAG

Register 50 is now in the protected region of the memory.

PRESS: 5 0 FMT ÷

DISPLAY: 1.000 000 000 00 → y
2.222 222 222 15 → x

The quotient of the operation appears in Y. However, register 50 contains 12 since the quotient of the operation was not stored in the 50 register.

DIVISION BY ZERO

The error light on the calculator will not appear to set when division by zero is performed using the FMT ÷ commands. In reality the light does set; however, instantaneously the 9101A sends the calculator an OPERATION Code which resets the light. The calculator's interpretation of infinity (9.999 999 999 99) is stored in the specified 9101A register.

EXAMPLE:

SWITCH: RUN FLOATING

SWITCH: FILE PROTECT OFF

COMMAND SET – INDIRECT ARITHMETIC

PRESS: CLEAR FMT SET FLAG

PRESS: 1 2 3 ↑

PRESS: 5 0 FMT y→( )

The value 123 is stored in register 50.

PRESS: CLEAR 5 0 FMT ÷

The value 9.999 999 999 99 is now present in register 50.

PRESS: FMT π

DISPLAY: 9.999 999 999 99 → X

COMMAND SET – PROGRAM TRANSFER

FMT FMT

P, → 9101A Transfers a program from the calculator memory to the 9101A memory. The transfer will begin at the first unprotected 9101A register. FMT FMT resets the calculator program counter to 00 (+00 in the 9100B) and initiates the transfer. At the time the transfer occurs, the X register must contain the assigned program number (0-99). As the program is being transferred the protect line moves to protect the program. When the transfer is complete, the X register will contain the number of the last protected 9101A register.

NOTE

Programs transferred into the 9101A must contain an END statement. Failure to properly terminate a program will result in the calculator's entire memory being repeatedly transferred into the 9101A until the 9101A memory is filled. At that time the transfer will STOP and a diagnostic code (4.444 444 444 15) will appear in the X register.

NOTE

Program number assignments must be unique; if two programs are assigned the same number, the program stored in the largest numbered registers cannot be recalled. Memory maps have been provided so that you will be able to assign unique program numbers.

Examples for FMT FMT are contained in the examples under FMT GO TO (the recall command).

ASSIGNMENT OF PROGRAM NUMBERS

Program numbers are assigned by the contents of the X register when FMT FMT is pressed. Excluding leading zeros, which are always ignored, the X register must contain no more than two digits and may be in fixed or floating display. Signs and exponents which may be present in the display are ignored. For example, all of the following displays contained in the X register when FMT FMT was pressed would assign program number 49:

	.000049	→	X
4.9	78	→	X
-4.9	78	→	X
4.9	-09	→	X

49.0 would be interpreted by the 9101A as 490 (an illegal program number assignment).

COMMAND SET – PROGRAM TRANSFER

FMT GO TO ( )( )

9100←P. Recalls to the 9100 memory protected programs identified by the contents of X. The calculator's program counter is automatically set to 00 (+00 in the 9100B) before the transfer and after. When the recall is finished the X register will contain the number of the last 9101A register recalled.

NOTE

Unprotected programs cannot be recalled. If an attempt is made to recall an unprotected program, the IMPROPER FORMAT lamp will light and a diagnostic code (5.555 555 555 15) will appear in the X register. If an attempt is made to recall a partially protected program, the protected part only will be recalled; the IMPROPER FORMAT lamp will light and a diagnostic code (5.555 555 555 14) will appear in the X register.

EXAMPLE:

SWITCH: **RUN FLOATING**

SWITCH: FILE PROTECT **OFF**

PRESS: CLEAR FMT SET FLAG END

SWITCH: **PROGRAM**

Key the following program into the calculator memory.

	STEP	KEY	KEY CODE
(+)	00	CLEAR	20
	01	1	01
	02	+	33
	03	PAUSE	57

STEP	KEY	KEY CODE
04	GO TO ( )( )	44
05	0	00
06	1	01
07	END	46

SWITCH: **RUN**

Assign one to the program and transfer it to the 9101A.

PRESS: 1 FMT FMT

CONTINUED

COMMAND SET – PROGRAM TRANSFER

FMT GO TO ( )( )

CONTINUED

DISPLAY: 0. 00 → X

The protect line (as indicated by the display) has moved to protect the program.

Recall the program:

PRESS: 1 FMT GO TO ( )( )

DISPLAY: 0. 00 → X

The program has been recalled and the program counter has been set to 00 (+00 in the 9100B). The change in display indicates that the zero register was the last register recalled. To execute the program, press CONTINUE, then press STOP to halt execution.

EXAMPLE:

A program that is not protected cannot be recalled.

PRESS: CLEAR FMT SET FLAG

This will set N = 0. The program just stored in the memory is unprotected.

PRESS: 1 FMT GO TO ( )( )

DISPLAY: 5.555 555 555 15 → X

The diagnostic code that appears in X indicates that program number one is not found in the protected area of the 9101A memory.

COMMAND SET – SUBPROGRAMS

FMT END

Always the last two instructions of a subprogram. These instructions terminate subprogram execution and return control to the calling source.

The keyboard as a calling source - FMT END stops subprogram and resets program counter to 00 (+00 in the 9100B). The X, Y and Z registers contain the results of the last subprogram operation.

A program as a calling source - FMT END terminates subprogram execution and returns control to the calling program. Execution of the calling program is resumed at the step immediately following the FMT GO TO which called the subprogram. When control is returned, the contents of the X register are replaced with the number of the last 9101A register occupied by the calling program. The Y and Z registers contain the results of the last subprogram operation.

NOTE

Subprograms are called with FMT GO TO.

Subprograms can call subprograms. This is known as "nesting" subprograms. Subprograms can be nested 14 deep.

FMT END is a good termination for programs which have value in their own right as "stand alone" programs and which are also frequently used as part of a larger program. By storing these programs in the 9101A with a FMT END termination they can be easily incorporated into a larger program or recalled for use as "stand alone" programs.

EXAMPLE:

The following program converts degrees, minutes and seconds into decimal degrees and sums the entries. Convert the program so that it contains a subprogram.

	STEP	KEY	KEY CODE
(+)	00	CLEAR	20
	01	x→( )	23
	02	d	17
	03	STOP	41
	04	ACC +	60
	05	6	06
	06	0	00
	07	÷	35
	08	ROLL ↓	31
	09	+	33
	0a	f	15
	0b	ROLL ↑	22
	0c	÷	35

STEP	KEY	KEY CODE
0d	÷	35
10	↓	25
11	+	33
12	d	17
13	+	33
14	y→( )	40
15	d	17
16	CLEAR	20
17	GO TO ( )( )	44
18	0	00
19	2	02
1a	END	46

COMMAND SET – SUBPROGRAMS

FMT END

CONTINUED

Break the program between steps 04 and 05; form the subprogram starting with step 04. End the subprogram after step 15.

Rewrite the "main" program.

	STEP	KEY	KEY CODE
(+)	00	CLEAR	20
	01	x→( )	23
	02	d	17
	03	STOP	41
	04	ACC +	60
	05	1	01
	06	FMT	42

STEP	KEY	KEY CODE
07	GO TO ( )( )	44
08	CLEAR	20
09	GO TO ( )( )	44
0a	0	00
0b	2	02
0c	END	46

Write the "subprogram".

	STEP	KEY	KEY CODE
(+)	00	6	06
	01	0	00
	02	÷	35
	03	ROLL ↓	31
	04	+	33
	05	f	15
	06	ROLL ↑	22
	07	÷	35
	08	÷	35

STEP	KEY	KEY CODE
09	↓	25
0a	+	33
0b	d	17
0c	+	33
0d	y→( )	40
10	d	17
11	FMT	42
12	END	46

Key the "main" program into the calculator memory starting at location 00 (+00 in the 9100B).

Move the protect line to N = 0.

PRESS: CLEAR FMT SET FLAG

Assign zero as the program number and store the program in the 9101A.

PRESS: FMT FMT

Key the "subprogram" into the calculator memory starting at location 00 (+00 in the 9100B).

COMMAND SET – SUBPROGRAMS

Assign one as the the program number and store the program in the 9101A.

PRESS: 1 FMT FMT

PRESS: 0 FMT GO TO ( )( ) CONT

ENTER DATA:

Degrees $\rightarrow z$
Minutes $\rightarrow y$
Seconds $\rightarrow x$

PRESS: CONT

DISPLAY: Decimal Degrees $\rightarrow x$

	DEGREES	MINUTES	SECONDS
	13	15	49
+	12	18	5
	25.565		

This section is divided into two parts: Sample Programs and Program Editing. The sample programs illustrate the use of the 9101A command set in a program. Program editing presents methods for finding and correcting program errors.

SAMPLE PROGRAMS

DATA STORAGE, INDIRECT ARITHMETIC

The following program (using FMT Y→( ) and FMT —) illustrates data storage and indirect arithmetic. In the program, data points (X_i) are manually entered. The data points are then stored in the 9101A [using FMT Y→( )] and a running sum is maintained in the calculator. After the last data point has been entered the arithmetic mean ($\bar{x}$) is computed and is subtracted (using FMT —) from each of the previously stored data points, resulting in each of the 9101A registers containing the deviation from the arithmetic mean ($X_i - \bar{x}$). The program uses the following equation:

$$\bar{x} = \frac{X_1 + X_2 + X_3 \ldots \ldots X_n}{n} = \frac{\sum_{i=1}^{n} X_i}{n}$$

Where n = number of data points.

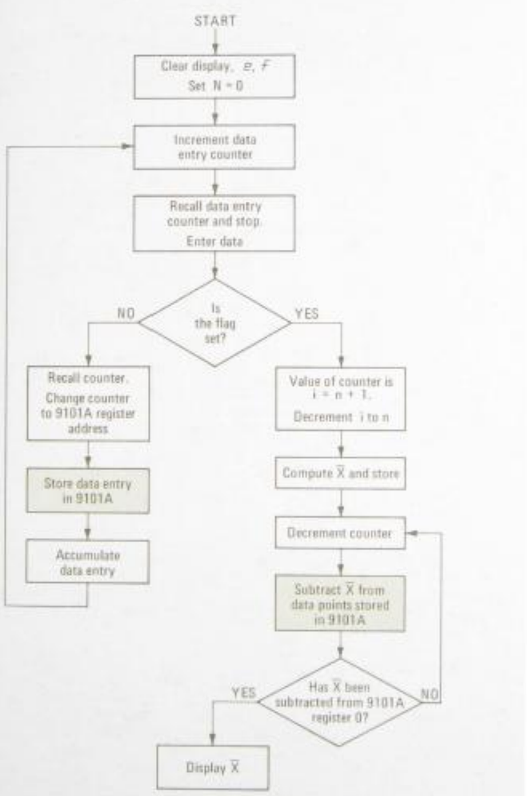
Execution of the program.

STEP	USER INSTRUCTIONS	DISPLAY X	Y	Z
1	PRESS: END, Enter program in 9100.			
2	PRESS: END, CONTINUE	i	0	0
3	Enter data: X_i	X_i	0	0
4	PRESS: CONTINUE			
5	To enter another data point go to Step 3. If the last data point has been entered, go to Step 6.			
6	PRESS: SET FLAG, CONTINUE	$\bar{x}$	0	0
7	To form a new average go to Step 2.			

NOTE

To observe the deviation from the mean of each data point, recall the contents of the appropriate 9101A register (starting with register Zero) using the FMT π instruction.

SAMPLE PROGRAMS



Shaded parts of flow chart indicate active use of the 9101A.

SAMPLE PROGRAMS

DATA STORAGE, INDIRECT ARITHMETIC
CONTINUED

PROGRAM STEPS OF PROGRAM

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	
00	CLEAR	20	0	0	0	Clear display, e, f Set N = 0
01	FMT	42	0	0	0	
02	SET FLAG	54	0	0	0	
03	1	01	1	0	0	Increment counter
04	ACC +	60	1	0	0	
05	f	15	i	0	0	Recall counter
06	STOP	41	X_i	0	0	Stop for data entry
07	IF FLAG	43	i	0	0	Branch after last entry
08	1	01	i	0	0	
09	b	14	i	0	0	
0a	↑	27	X_i	X_i	0	Recall counter, change counter to 9101A register address
0b	f	15	i	X_i	0	
0c	↑	27	i	i	X_i	
0d	1	01	1	i	X_i	
10	—	34	1	i−1	X_i	
11	↓	25	i−1	X_i	X_i	
12	FMT	42	i−1	X_i	X_i	Store data entry in 9101A
13	y→( )	40	i−1	X_i	X_i	
14	CLEAR x	37	0	X_i	X_i	Accumulate data, clear display registers
15	ACC +	60	0	X_i	X_i	
16	↑	27	0	0	X_i	
17	↑	27	0	0	0	
18	GO TO ( )( )	44	0	0	0	Branch for new data entry
19	0	00	0	0	0	
1a	3	03	0	0	0	
1b	1	01	1	0	0	Value of counter is i = n + 1. Decrement i to n
1c	ACC −	63	1	0	0	
1d	RCL	61	n	ΣX_i	0	Compute $\bar{X}$
20	÷	35	n	$\bar{X}$	0	
21	y→( )	40	n	$\bar{X}$	0	Store $\bar{X}$
22	e	12	n	$\bar{X}$	0	
23	↓	25	$\bar{X}$	0	0	Decrement counter
24	1	01	1	0	0	
25	ACC −	63	1	0	0	
26	RCL	61	i	$\bar{X}$	0	Recall counter and $\bar{X}$
27	FMT	42	i	$\bar{X}$	0	Subtract $\bar{X}$ from data point stored in 9101A
28	—	34	i	$\bar{X}$	0	
29	↓	25	$\bar{X}$	0	0	If $\bar{X}$ has been subtracted from 9101A register 0 then branch
2a	f	15	i	0	0	
2b	IF x = y	50	i	0	0	
2c	3	03	0	0	0	
2d	3	03	0	0	0	
30	GO TO ( )( )	44	i	0	0	Branch to calculate deviation for next data point
31	2	02	i	0	0	
32	3	03	i	0	0	
33	e	12	$\bar{X}$	0	0	Finalize display
34	END	46	$\bar{X}$	0	0	Display

Shaded parts of program indicate active use of the 9101A.

STORAGE:
e ΣX_i, $\bar{X}$
f i

SAMPLE PROGRAMS

SEQUENTIAL PROGRAM EXECUTION

The following example (using FMT GO TO) illustrates sequential program execution. In the first program, data points (X_i) are manually entered. A running sum is maintained of the data points ($\sum_{i=1}^{n} X_i$) and the data points squared ($\sum_{i=1}^{n} \left[X_i^2\right]$). After the last data point has been entered $\overline{X}$ is computed. The first program stores $\sum_{i=1}^{n} X_i$, $\sum_{i=1}^{n} \left[X_i^2\right]$ and $\overline{X}$ and calls program number two (using FMT GO TO). Program number one uses the following equations:

$$\sum_{i=1}^{n} X_i = X_1 + X_2 + X_3 \ldots X_n$$

$$\sum_{i=1}^{n} \left[X_i^2\right] = X_1^2 + X_2^2 + X_3^2 \ldots X_n^2$$

$$\overline{X} = \frac{X_1 + X_2 + X_3 \ldots X_n}{n}$$

Where n = number of data points.

Program number two calculates the standard deviation, using the summations formed in program one. Program number two uses the following equation:

$$S = \sqrt{\frac{1}{(n-1)}\left(\sum_{i=1}^{n}\left[X_i^2\right] - \frac{\left(\sum_{i=1}^{n} X_i\right)^2}{n}\right)}$$

SAMPLE PROGRAMS

SEQUENTIAL PROGRAM EXECUTION
CONTINUED

Storing the Programs in the 9101A.

STEP	USER INSTRUCTIONS	DISPLAY X	Y	Z
1	PRESS: END, Enter program No. 1 into the calculator memory.			
2	SWITCH: FILE PROTECT OFF			
3	PRESS: CLEAR, FMT, SET FLAG			
4	PRESS: 1, FMT, FMT			
5	PRESS: END, Enter program No. 2 into the calculator memory.			
6	PRESS: 2, FMT, FMT			

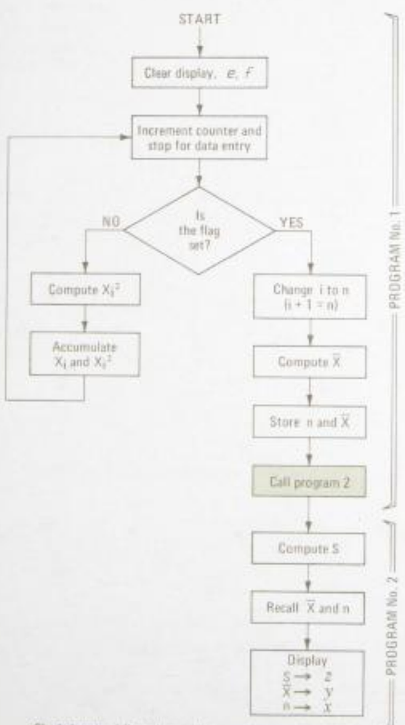
Execution of the Programs.

STEP	USER INSTRUCTIONS	DISPLAY X	Y	Z
1	PRESS: 1, FMT, GO TO			
2	PRESS: CONTINUE	0	i	0
3	Enter data	X_i	i	0
4	PRESS: CONTINUE			
5	If another data point is to be entered go to Step 3. If last data point has been entered go to Step 6.			
6	PRESS: SET FLAG, CONTINUE	N	$\overline{X}$	S

SAMPLE DATA:

$$\left.\begin{matrix} 4 \\ 5 \\ 6 \\ 4 \\ 7 \\ 2 \end{matrix}\right\} = \begin{matrix} S & 1.751 \\ \overline{X} & 4.667 \\ N & 6 \end{matrix}$$

SAMPLE PROGRAMS



Shaded parts of flow chart indicate active use of the 9101A.

SAMPLE PROGRAMS

SEQUENTIAL PROGRAM EXECUTION
CONTINUED

Program steps of program No. 1

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	
00	CLEAR	20	0	0	0	Clear display, e, f
01	1	01	1	0	0	Increment counter
02	+	33	1	i	0	
03	CLEAR x	37	0	i	0	Finalize display, stop for data entry
04	STOP	41	X_i	i	0	
05	IF FLAG	43	0	i	0	Branch after last entry
06	1	01	0	i	0	
07	1	01	0	i	0	
08	↑	27	X_i	X_i	i	Compute X_i^2
09	×	36	X_i	$[X_i^2]$	i	
0a	ACC +	60	X_i	$[X_i^2]$	i	Accumulate X_i and X_i^2
0b	CLEAR x	37	0	$[X_i^2]$	i	Prepare display for branch
0c	ROLL ↓	31	$[X_i^2]$	i	0	
0d	GO TO ( ) ( )	44	$[X_i^2]$	i	0	Branch for new data entry
10	0	00	$[X_i^2]$	i	0	
11*	1	01	$[X_i^2]$ 1	i	0	Change i to n (i + 1 = n)
12	−	34	1	n	0	
13	↑	27	1	1	n	Compute $\bar{X}$
14	RCL	61	ΣX_i	$\Sigma[X_i^2]$	n	
15	ROLL ↑	22	n	ΣX_i	$\Sigma[X_i^2]$	
16	÷	35	n	$\bar{X}$	$\Sigma[X_i^2]$	
17	x→( )	23	n	$\bar{X}$	$\Sigma[X_i^2]$	Store n
18	d	17	n	$\bar{X}$	$\Sigma[X_i^2]$	
19	y→( )	40	n	$\bar{X}$	$\Sigma[X_i^2]$	Store $\bar{X}$
1a	c	16	n	$\bar{X}$	$\Sigma[X_i^2]$	
1b	2	02	2	$\bar{X}$	$\Sigma[X_i^2]$	Branch to program 2
1c	FMT	42	2	$\bar{X}$	$\Sigma[X_i^2]$	
1d	GO TO ( ) ( )	44	2	$\bar{X}$	$\Sigma[X_i^2]$	
20	END	46	2	$\bar{X}$	$\Sigma[X_i^2]$	

* Step 11 has a double function.

Shaded parts of program indicate active use of the 9101A.

STORAGE:
- c $\bar{X}$
- d n
- e $\Sigma[X_i^2]$
- f ΣX_i

SAMPLE PROGRAMS

Program steps of program No. 2.

	STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	
(+)	00	f	15	ΣX_i	$\bar{X}$	$\Sigma[X_i^2]$	
	01	↑	27	ΣX_i	ΣX_i	$\bar{X}$	
	02	×	36	ΣX_i	$(\Sigma X_i)^2$	$\bar{X}$	
	03	d	17	n	$(\Sigma X_i)^2$	$\bar{X}$	
	04	÷	35	n	$(\Sigma X_i)^2/n$	$\bar{X}$	
	05	e	12	$\Sigma[X_i^2]$	$(\Sigma X_i)^2/n$	$\bar{X}$	
	06	$x \rightleftarrows y$	30	$(\Sigma X_i)^2/n$	$\Sigma[X_i^2]$	$\bar{X}$	
	07	—	34	$(\Sigma X_i)^2/n$	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	$\bar{X}$	
	08	d	17	n	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	$\bar{X}$	
	09	↑	27	n	n	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	Compute S
	0a	1	01	1	n	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	
	0b	—	34	1	$n-1$	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	
	0c	$x \rightleftarrows y$	30	$n-1$	1	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	
	0d	÷	35	$n-1$	$1/n-1$	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	
	10	ROLL ↓	31	$1/n-1$	$\Sigma[X_i^2]-(\Sigma X_i)^2/n$	$n-1$	
	11	×	36	$1/n-1$	S^2	$n-1$	
	12	↓	25	S^2	$n-1$	$n-1$	
	13	$\sqrt{x}$	76	S	$n-1$	$n-1$	
	14	↑	27	S	S	$n-1$	
	15	c	16	$\bar{X}$	S	$n-1$	Recall $\bar{X}$
	16	↑	27	$\bar{X}$	$\bar{X}$	S	
	17	d	17	n	$\bar{X}$	S	Recall n
	18	END	46	n	$\bar{X}$	S	Display

STORAGE:
c $\bar{X}$
d n
e $\Sigma[X_i^2]$
f ΣX_i

SAMPLE PROGRAMS

SUBPROGRAMS

The following program (using FMT GO TO and FMT END) illustrates subprogramming. The program computes possible combinations (C) of N objects, taken K at a time using the following equation:

$$C_K^N = \frac{N!}{K!\,(N-K)!}$$

Where $n! = n(n-1)\ (n-2) \ldots (3)\ (2)\ (1)$ and $0 < N \leq 69$

A subprogram is used to calculate n!

Storing the Programs in the 9101A.

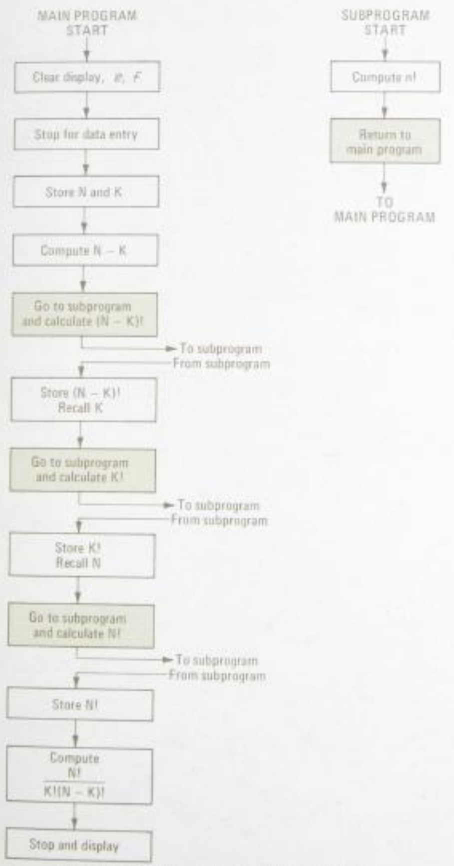
STEP	USER INSTRUCTIONS	DISPLAY X	DISPLAY Y	DISPLAY Z
1	PRESS: END, Enter main program into the calculator memory.			
2	SWITCH: FILE PROTECT OFF			
3	PRESS: CLEAR, FMT, SET FLAG			
4	PRESS: 1, FMT, FMT			
5	PRESS: END, Enter subprogram into the calculator memory.			
6	PRESS: 2, FMT,FMT			

Execution of the Programs.

STEP	USER INSTRUCTIONS	DISPLAY X	DISPLAY Y	DISPLAY Z
1	PRESS: 1, FMT, GO TO			
2	PRESS: CONTINUE	0	0	0
3	Enter data	K	N	
4	PRESS: CONTINUE	0	0	C

SAMPLE DATA:
N = 15
K = 5
C = 3003

SAMPLE PROGRAMS



Shaded parts of flow chart indicate active use of the 9101A.

SAMPLE PROGRAMS

SUBPROGRAMS
CONTINUED

Program steps of main program.

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	
(+) 00	CLEAR	20	0	0	0	Clear display, e, f
01	STOP	41	K	N	0	Stop for data entry
02	ACC +	60	K	N	0	Store K and N
03	—	34	K	(N − K)	0	Compute N − K
04	2	02	2	(N − K)	0	Go to subprogram to calculate (N − K)!
05	FMT	42	2	(N − K)	0	
06	GO TO ( ) ( )	44	2	(N − K)	0	
07	↓	25	(N − K)	(N − K)!	(N − K)!	Store (N − K)!
08	y→( )	40	(N − K)	(N − K)!	(N − K)!	
09	d	17	(N − K)	(N − K)!	(N − K)!	
0a	f	15	K	(N − K)!	(N − K)!	Recall K
0b	↑	27	K	K	(N − K)!	
0c	2	02	2	K	(N − K)!	Go to subprogram to calculate K!
0d	FMT	42	2	K	(N − K)!	
10	GO TO ( ) ( )	44	2	K	(N − K)!	
11	↓	25	K	K!	K!	Store K!
12	y→( )	40	K	K!	K!	
13	x↔f	15	K	K!	K!	
14	e	12	N	K!	K!	Recall N
15	↑	27	N	N	K!	
16	2	02	2	N	K!	Go to subprogram to calculate N!
17	FMT	42	2	N	K!	
18	GO TO ( ) ( )	44	2	N	K!	
19	↓	25	N	N!	N!	Store N!
1a	y→( )	40	N	N!	N!	
1b	e	12	N	N!	N!	
1c	d	17	(N − K)!	N!	N!	Given (N − K)!, N!, K! Compute N!/K!(N − K)!
1d	↑	27	(N − K)!	(N − K)!	N!	
20	f	15	K!	(N − K)!	N!	
21	×	36	K!	K!(N − K)!	N!	
22	ROLL ↓	31	K!(N − K)!	N!	K!	
23	÷	35	K!(N − K)!	N!/K!(N − K)!	K!	
24	CLEAR x	37	0	N!/K!(N − K)!	K!	Finalize display
25	↑	27	0	0	N!/K!(N − K)!	
26	END	46	0	0	N!/K!(N − K)!	Display

Shaded areas of program indicate active use of the 9101A.

MAIN PROGRAM STORAGE

d (N − K)!
e ← N!
f ← K!

SAMPLE PROGRAMS

Program steps of n! subprogram.

	STEP	KEY	KEY CODE	DISPLAY X	Y	Z	
(+)	00	CLEAR x	37	0	n		
	01	↑	27	0	0	n	
	02	ROLL ↑	22	n	0	0	
	03	IF x=y	50				If n is zero enter 1
	04	ACC +	72				
	05	1	01				
	06	↑	27	n	n		
	07	↑	27	n	n	n	
	08	1	01	1	n	n	Decrement n
	09	−	34	1	n − 1	n	
	0a	IF x>y	53	1	n − 1	n	Branch if n! has been computed
	0b	1	01	1	n − 1	n	
	0c	5	05	15	n − 1	n	
	0d	ROLL ↓	31	n − 1	n	1	
	10	×	36	n − 1	(n) (n−1)	1	
	11	ROLL ↑	22	1	n − 1	(n) (n−1)	
	12	GO TO ( ) ( )	44	1	n − 1	(n) (n−1)	
	13	0	00	1	n − 1	(n) (n−1)	
	14	9	11	1	n − 1	(n) (n−1)	
	15	FMT	42	1	n − 1	n!	Return to main program
	16	END	46				

Shaded areas of subprogram indicate active use of the 9101A.

SAMPLE PROGRAMS

MATRIX STORAGE AND RECALL

The following program demonstrates (using FMT Y→( ) and FMT π) a method of storing and recalling data in a two dimensional array of the format:

$$\begin{matrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1M} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2M} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3M} \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NM} \end{matrix}$$

The address of the 9101A register $(a_{ij}) = N(i-1) + j-1$.

Where: i is the row
j is the column
N is the number of rows
M is the number of columns
" MN ≤ 248

NOTE

This program need not be stored in the 9101A. If it is, the following user instructions must be modified to include recall from the 9101A.

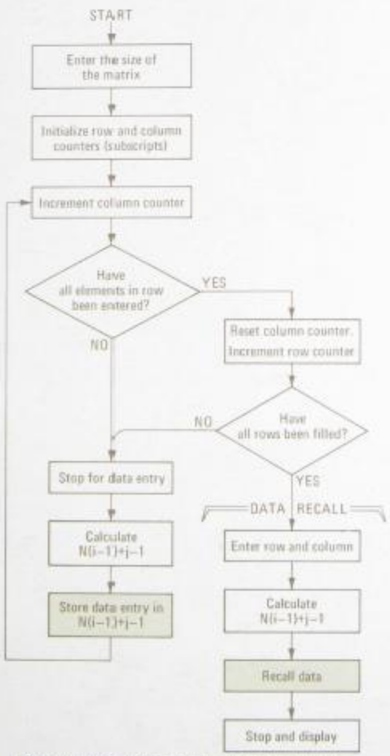
Data Entry

STEP	USER INSTRUCTIONS	DISPLAY X	Y	Z
1	PRESS: CLEAR, FMT, SET FLAG, END. Enter program.			
2	PRESS: END, CONTINUE	0	0	0
3	Enter N and M	M	N	0
4	PRESS: CONTINUE	0	j	i
5	Enter data	aij	j	i
6	If all rows and columns have been filled go to Step 7. If not, go to Step 4.			
7	Display	0	0	0

Data Recall

STEP	USER INSTRUCTIONS	DISPLAY X	Y	Z
1	PRESS: GO TO, 3, 3	0	0	0
2	PRESS: CONTINUE	j	i	0
3	Enter some j and i	0	0	aij
4	PRESS: CONTINUE			
5	To recall more data go to Step 3.			

SAMPLE PROGRAMS



Shaded parts of flow chart indicate active use of the 9101A.

SAMPLE PROGRAMS

MATRIX STORAGE AND RECALL
CONTINUED

	STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	
(+)	00	CLEAR	20	0	0	0	Clear display, e, f
	01	STOP	41	M	N	0	Enter and store size of matrix
	02	x→( )	23	M	N	0	
	03	c	16	M	N	0	
	04	y→( )	40	M	N	0	
	05	d	17	M	N	0	
	06	1	01	1	N	0	Initialize row counter (i)
	07	x→( )	23	1	N	0	
	08	e	12	1	N	0	
	→ 09	CLEAR x	37	0	N	0	Increment column counter (j)
	0a	↑	27	0	0	N	
	0b	1	01	1	0	N	
	0c	ACC +	60	1	0	N	
	0d	↑	27	1	1	0	Branch if columns have not been filled
	10	c	16	M	1	0	
	11	+	33	M	M + 1	0	
	12	f	15	j	M + 1	0	
	13	IF x<y	52	j	M + 1	0	
	14	2	02				
	15	1	01				
	16	RCL	61	j	i	0	Clear row and column counters
	17	ACC −	63	j	i	0	
	18	1	01	1	i	0	Increment row counter. Reset column counter to one
	19	+	33	1	i + 1	0	
	1a	ACC +	60	1	i + 1	0	
	1b	d	17	N	i + 1	0	Branch if all rows have been filled
	1c	IF x<y	52	N	i + 1	0	
	1d	3	03				
	20	5	05				
	→ 21	RCL	61	j	i	0	Arrange display; stop for data entry
	22	↑	27	j	i	i	
	23	CLEAR x	37	0	i	i	
	24	STOP	41	aij	i	i	
	25	ROLL ↓	31	j	i	aij	Calculate N(i−1)+j−1
	26	d	17	N	i	aij	
	27	×	36	N	Ni	aij	
	28	−	34	N	N(i−1)	aij	
	29	f	15	j	N(i−1)	aij	
	2a	+	33	j	j+N(i−1)	aij	
	2b	1	01	1	j+N(i−1)	aij	
	2c	−	34	1	j−1+N(i−1)	aij	
	2d	↓	25	j−1+N(i−1)	aij	aij	
	30	FMT	42	j−1+N(i−1)	aij	aij	Store aij
	31	y→( )	40	j−1+N(i−1)	aij	aij	
	32	GO TO ( ) ( )	44	j−1+N(i−1)	aij	aij	Branch for next entry
	33	0	00	j−1+N(i−1)	aij	aij	
	34	9	11	j−1+N(i−1)	aij	aij	
	→ 35	CLEAR	20	0	0	0	Clear display, e, f

CONTINUED

Shaded areas of program indicate active use of the 9101A.

PROGRAM STORAGE:

c Number of columns (M)

d Number of rows (N)

e Row counter (i)

f Column counter (j)

SAMPLE PROGRAMS

STEP	KEY	KEY CODE	DISPLAY X	DISPLAY Y	DISPLAY Z	
36	STOP	41	j	i	0	Enter some i and j
37	x⇄y	30	i	j	0	Calculate N(i−1)+j−1
38	↑	27	i	i	j	
39	d	17	N	i	j	
3a	×	36	N	Ni	j	
3b	−	34	N	N(i−1)	j	
3c	↓	25	N(i−1)	j	j	
3d	+	33	N(i−1)	j+N(i−1)	j	
40	1	01	1	j+N(i−1)	j	
41	−	34	1	j−1+N(i−1)	j	
42	↓	25	j−1+N(i−1)	j	j	
43	FMT	42	j−1+N(i−1)	j	j	Arrange display
44	π	56	aij	j	j	
45	↑	27	aij	aij	j	
46	CLEAR X	37	0	aij	j	
47	↑	27	0	0	aij	
48	GO TO ( ) ( )	44	0	0	aij	Branch for display
49	3	03	0	0	aij	
4a	6	06	0	0	aij	
4b	END	46				

Shaded areas of program indicate active use of the 9101A.

PROGRAM STORAGE:

c Number of columns (M)

d Number of rows (N)

e Row counter (i)

f Column counter (j)

EDITING

Editing procedures differ, in certain respects, between the 9100A and the 9100B. This section will first cover the techniques common to both calculators; then, to cover the differences, the reader will be directed to the proper section which discusses his particular calculator.

EDITING IN PROGRAM MODE

Editing a program with the PROGRAM-RUN switch in the PROGRAM position is used to verify that steps of a program have been correctly entered in the Calculator as written in the program. Programs that are stored in the 9101A can be checked by recalling them to the 9100 using the FMT GO TO instruction and checking them.

NOTE

An individual 9101A register containing program steps **CANNOT** be recalled for checking using the FMT π instruction. This instruction **CANNOT** transfer program steps. Instead it will interpret the contents of the 9101A register as data and recall a meaningless constant to the X register.

FMT END

In the RUN mode the FMT END instruction will not perform as expected when the STEP PRGM key is being used. When encountered, this command will not return control of the calling program. It will, instead, reset the Calculator memory to (+)00 (the start of the subprogram).

Here is one procedure for manually performing the function performed by the FMT END instruction:

1. Use FMT GO TO to return to the calling program (its number must be placed in the X register).
2. Use GO TO to address the program counter to the proper location in the calling program.

PAUSE

The pause key **MUST NOT BE USED** to manually halt the execution of a 9100/9101A program. If this key is used, program steps could be changed to key codes 57 or 77. There is also a possibility in a 9100B that the instrument would do a series of SUB RETURN instructions. The pause key can, however, be stored in memory for use as a temporary display.

NOTE

The pause key must not be used to manually halt the execution of a 9100/9101A program.

EDITING

FMT GO TO

Ensure the number which designates a program to be called (using FMT GO TO) is not considered by the calculator to be a continuation of a previously entered number. If it is, either the wrong program will be called or the IMPROPER FORMAT lamp will light and a diagnostic code will appear in the X register. For example: the keys 1 and 2 are pressed and the operator is interrupted. Upon returning, he presses 3 FMT GO TO. This will cause the IMPROPER FORMAT lamp to light and a diagnostic code (7.777 777 777 14) to appear in the X register. This occurance is difficult for the operator to analyze because the improper number in the X register is replaced with the diagnostic code and could lead the operator to believe his 9101A is defective when it is not. This problem can be avoided by the use of CLEAR X (or any other operation key) whenever the possibility exists that a digit key was the last key pressed.

9100A ONLY

The 9100A user will find that when he is in the RUN mode and stepping through a program with the STEP PRGM key the following instructions will initiate automatic program execution: CONTINUE, PRINT/SPACE, FMT. (PRINT/SPACE only when a 9120A is present in the system.)

Here is one method for overcoming this situation. In this method, only FMT is discussed; however, the procedure could be used for the other instructions. (See the EDITING A PROGRAM section of the 9100A Operating and Programming Manual for more instructions that do not function as expected.)

1. Use the STEP PRGM key to step to (not through) the FMT instruction.
2. Manually branch (using GO TO) around the FMT and associated instruction.
3. Manually key the instructions that were skipped.

EXAMPLE:

Assume the following program segment is in the 9100A memory:

STEP	KEY	KEY CODE
37	+	33
38	3	03
39	FMT	42
3a	y→( )	40
3b	×	36

EDITING

9100A ONLY
CONTINUED

Step to *38* using the STEP PRGM key. Then, press these keys:

GO TO ( ) 3 *b*

FMT y→( )

and continue stepping through the memory until the next FMT instruction is encountered.

9100B ONLY

The 9100B/9101A user will have no special problem with editing a program in the RUN mode. You should, however, review the EDITING A PROGRAM section of the 9100B Operating and Programming manual; there are a few keys that do not act as expected.

The STEP PRGM key offers the 9100B user the feature of checking his 9100B/9101A program in a very dynamic way. If the STEP PRGM key is pressed and held down a 9100B/9101A program will be stopped immediately after the next FMT and associated instruction. For example, assume the following program steps are in a program being run and the STEP PRGM key is being held down:

STEP	KEY	KEY CODE
22	↑	27
23	5	05
24	0	00
25	FMT	42
26	y→( )	40
27	RCL	61
28	—	34

The program will be stopped after step 26. If CONT is then pressed the program will resume, executing the RCL instruction in step 27.

If the STEP PRGM key is pressed and held down through a FMT END instruction, control will not be returned to the calling program. It will, instead, reset the Calculator memory to +00 (the start of the subprogram) and stop the program execution.

CORRECTING PROGRAMS

Programs are transferred between the 9100 and the 9101A one register at a time. If a program is stored in the 9101A that extends to character location 0 of a particular register, then the 9101A must assign that entire register to that program, leaving character locations 1 through d unused.

One of three situations will confront the operator attempting to correct a program stored in the 9101A:

1. The correction will shorten the program.
2. The correction will lengthen the program but not extend the program length beyond the unused locations which may be present in the last register (eg. 1 through d).
3. The correction will lengthen the program and extend it into an extra register (or registers) thus overlapping the next program.

Because of their similarity, situations 1 and 2 will be treated as one in the following procedures.

SITUATION 1 & 2

1. From the memory map obtain the number of the first 9101A register occupied by the program. Record this number.
2. Recall the program into the calculator memory and make the needed change.
3. Move the protect line so that N = the number recorded in 1 (above).
4. Assign the program number and store the corrected program.
5. Move the protect line to protect the desired area of the memory. (Programs cannot be recalled if they are unprotected.)

SITUATION 3

1. From the memory map, obtain the number of the first 9101A register occupied by the program to be corrected. Record this number.
2. Begin recalling all of the programs stored in the memory below the program to be corrected (higher numbered registers). Recall them to the calculator memory one at a time and record them on magnetic cards.
3. Recall the program to be corrected. Perform the correction.
4. Move the protect line to N = the number recorded in 1 (above).
5. Restore the correct program.
6. One at a time, ENTER the programs from the magnetic cards into the calculator and store them in the 9101A.

PERFORMANCE ASSURANCE

The filter for the blower motor is the only part of the 9101A that requires periodic attention. Under normal office environment, this filter should be cleaned every six months with a warm detergent solution. The filter, located on the rear panel of the instrument, is removed by pressing one side of the plastic frame toward the center of the filter. After cleaning the filter, allow it to dry and reinstall.

NOTE

The motor is sealed and requires no periodic oiling.

CHANGING THE FUSE

The 9101A has one fuse located on the rear panel of the instrument. Spare fuses were shipped with the instruments as supplied accessories. To replace the fuse:

1. Switch the LINE switch to OFF.
2. Disconnect the 9101A from the a.c. power source.
3. Pushing in slightly, rotate the fuse holder cap, counter clockwise, approximately 90 degrees; remove the cap.
4. Replace the fuse and reinstall the cap and fuse in the instrument.

If the new fuse burns out immediately after being installed, maintenance for the instrument is needed.

CHANGING THE LAMPS

The 9101A has three lamps: the LINE, FILE PROTECT and IMPROPER FORMAT. These lamps are all the same and are located under their respective covers. To change the bulb, unscrew the cover, install the new bulb and replace the cover in the front panel.

The following procedure lights all the lamps; unlit lamps are defective and should be replaced.

NOTE

This procedure assumes the 9101A fuse is good.

1. Switch the 9100 and 9101A ON —the LINE lamp will light
2. Switch the FILE PROTECT ON —the FILE PROTECT lamp will light
3. PRESS: FMT, SET FLAG —the IMPROPER FORMAT lamp will light