600

600 Series

Reference Manual with Programming

WANG

600 SERIES

Reference Manual with Programming

CHAPTER 1: PREPARING MACHINE FOR MANUAL OPERATION

CHAPTER 2: DATA ENTRY KEYS AND RIGHT AND LEFT REGISTERS

CHAPTER 3: SCIENTIFIC AND ENGINEERING KEYS

CHAPTER 4: FUNCTION SELECTOR KEYS AND REGISTERS

CHAPTER 5: REVIEW BY EXAMPLES

CHAPTER 6: FLOW DIAGRAMS

CHAPTER 7: RELATING FLOW DIAGRAMS TO COMPUTER PROGRAMMING

CHAPTER 8: PROGRAMMING CONCEPTS – LOOPS, DECISIONS, AND SUBROUTINES

CHAPTER 9: DEBUGGING

CHAPTER 10: DATA STORAGE

CHAPTER 11: USING INDIRECT ADDRESSING IN YOUR 600

CHAPTER 12: AN EXERCISE IN PROGRAMMING

CHAPTER 13: CASSETTE OPERATING SYSTEM

CHAPTER 14: PRINTING CAPABILITY (OPTIONAL)

CHAPTER 15: PROGRAM STEP TRACE VIA PRINTER CONTROL (AUTOPRINT ONLY)

© Wang Laboratories, Inc., 1972

WANG LABORATORIES, INC.

836 NORTH STREET, TEWKSBURY, MASSACHUSETTS 01876, TEL. (617) 851-7311, TWX 710 343-6769, TELEX 94-7421

INTRODUCTION

The WANG MODEL 600 PROGRAMMABLE CALCULATOR is one of the more advanced calculators on the market. As a unit, the Model 600 combines tremendous calculating power with a programming and data manipulation capability unmatched in its field.

Features include a programmable memory capable of storing from 312 to a maximum of 1,848 program steps, 247 storage registers, or many combinations of the two. As far as scientific and engineering type calculations are concerned, a glance of the keyboard will show you just how powerful a calculator you have. Such functions as x^2, $\sqrt{x}$, $\log_e x$, e^x, 10^x, $1/x$, $|x|$, Integer X, as well as a complete set of trigonometric functions are available, at the touch of a key. In addition, a floating decimal point or a true scientific notation can be easily obtained, while Trigonometric input can be in either radians or degrees, as is convenient.

If you choose the printing option, a 21-column drum printer will print out on a paper tape the results of calculations — fully labeled — as well as provide a listing of any program entered in memory, along with identifying symbols. The column printer can also be used to trace the logic of a program during its execution. In this way, the user can watch the program's flow as it progresses, easily picking out programming problems.

THE MODEL 600 HAS EXTENSIVE PROGRAMMING CAPABILITY

But, this is only half of the story, and roughly a quarter of this manual! As you will see, everything you can do manually, can also be done under program control. This requires a decision-making capability, enabling the program flow to change as is necessary. To accomplish this, the Model 600 Calculator can make any of 9 different decisions automatically, and follow the course of action you choose. Included in these decisions is a set of alphanumeric compares which allow you, the programmer, to compare entire strings of characters — letters as well as numbers — and branch accordingly. This is especially useful when storing and recalling information from storage peripherals.

Programming the Model 600 can be easily accomplished in a number of ways: Either directly into the memory via the keyboard; by marked sense card, or by magnetic tape from an optional tape cassette unit built into the calculator. This tape cassette unit is one of the most useful features available. Programs or data recorded on tape can be loaded into the Model 600's memory either manually or under program control. Each tape cassette can hold a maximum of 32,000 program steps (stored on two tracks). As a result, programs can be easily recorded on tape and reloaded back into the memory as needed.

Once a program is entered into the Model 600's memory area, a touch of a key is all that it takes to execute it. As a matter of fact, you can enter 32 or more different routines into the Model 600, and execute any one of them just this easily! If you find that you are constantly referring to a particular routine or set of routines, you can order a special "hard-wired" version of your program(s) which you can literally "plug-in" to your Model 600 Calculator. This will allow you to use these programs at any time while the regular Model 600 memory area is left entirely free for your other programming uses.

MODEL 600 SERIES ADD-ON PERIPHERALS

The Model 600's capabilities do not end here, however. You can increase the power of your calculating system, as your needs expand, by adding on any number of input/output and data storage devices. These include a full line of output writers, plotters, paper tape and marked sense card readers, as well as peripheral data and program storage devices of several types. You'll want to contact your Wang representative for full particulars.

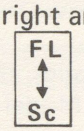
TABLE OF CONTENTS

PART I OPERATING YOUR WANG 600 MANUALLY

CHAPTER 1: Preparing Machine for Manual Operation 1

Specific instructions for setting up your Wang Model 600 and clearing the machine of previous calculations; the five modes of operation are briefly described.

CHAPTER 2: Data Entry Keys and Right and Left Registers 3

Description with appropriate examples for each of the numeric keys (0-9), decimal point, change sign, clear display, set exponent (for scientific notation), recall right and left, store right and left, x = right and left, ÷ = right and left, + right and left, – right and left, T (total) right and left, and floating to scientific 

switch.

An example of mixing operations is included at the end of the chapter.

CHAPTER 3: Scientific and Engineering Keys 12

Description with appropriate examples for each of the following keys:
x^2, INT x, $\sqrt{x}$, $|x|$, $LOG_e x$, $LOG_{10} x$, e^x, 10^x, 1/x, DEG → RAD, RAD → DEG, SIN, SIN^{-1}, COS, COS^{-1}, TAN, TAN^{-1}, DEG-TO-RADIAN SWITCH, and the SHIFT key for upper case functions.

CHAPTER 4: Function Selector Keys and Registers 20

The basic keyboard registers [00] through [15] are described as storage registers with the capability of the adding, subtraction, multiplying, dividing, storing, recalling, totalling and swapping operations within each. Also explained are the following switches: clear and f(x); f(x) with the special registers will take on additional capability in later chapters on programming.

CHAPTER 5: Review By Examples 28

Eight examples utilizing combinations of the keys explained in the first four chapters are presented here as exercises for using your new Wang 600 as a manually operated problem solver. Each example has a comment column included to instruct you on where numbers and expressions are stored.

TABLE OF CONTENTS (Continued)

PART II PROGRAMMING YOUR WANG 600

Introduction to Programming 46

CHAPTER 6: Flow Diagrams 48

This chapter introduces some basic concepts of programming through logic diagrams, better known as flow diagrams. This is must reading for individuals without prior programming experience. The four examples included in the chapter are logically presented and easy to follow.

CHAPTER 7: Relating Flow Diagrams to Computer Programming 54

Just as the title implies, the concepts of Chapter 6, Flow Diagrams, are expanded and tied into the Wang Model 600 for a user's first attempt at writing a program. This chapter with its many examples, provides the ground work for forthcoming and more advanced concepts presented in Chapter 8.

CHAPTER 8: Programming Concepts — Loops, Decisions, and Subroutines 66

Endless and terminating loops are presented first. With the introduction of the notion of decisions on a programmable calculator, the concept of subroutines emerges from a simple one-level to five-level subroutines. The subroutine "Alpha Return and Print" feature is explained. The many examples included in the chapter are diversified enough so that every concept is related to a real-life application.

CHAPTER 9: Debugging 97

The ability to use debugging techniques in program writing is as important as writing the program itself. The use of the RUN/LEARN Mode is thoroughly explained, as is the use of the various debugging keys — STEP, PROGRAM COUNT (P.C.), BACKSTEP, PROGRAM ERROR INDICATOR, MACHINE ERROR INDICATOR, VERIFY PROGRAM (Verify), MARK SEARCH and STOP, DELETE, and INSERT are discussed in detail. Numerous examples are given.

CHAPTER 10: Data Storage 109

The internal memory space of your Model 600 with its ability to use this space for either program steps or additional data storage is explained here. A memory diagram is given for each of the three 600 models.

CHAPTER 11: Using INDIRECT Addressing in Your 600 117

The concept of INDIRECT Addressing is explained, including the use of the Special Selector Switches and the "Pointer" register.

CHAPTER 12: An Exercise in Programming 124

A fictitious character tries and succeeds in thinking through a programming problem. The pitfalls and their solutions are provided. The four problems are more complex than normally included in a manual, but hopefully will be helpful in preparing Wang users for out-of-the-ordinary programming.

TABLE OF CONTENTS (Continued)
PART II

CHAPTER 13: Cassette Operating System 138

Wang's time-tested and proven tape cassette for program and data storage is fully explained. Chaining of programs and the procedure for using the ALPHA LOAD PROG. Automatic program block load is presented. The chapter includes the procedure for transferring a program from memory to tape and from tape to memory, as well as the procedure for storing and recalling the contents of data registers with the tape.

CHAPTER 14: Printing Capability (Optional) 147

Examples and procedures for the manual, automatic and programming operations necessary to gain maximum benefit from your high-speed alpha-numeric drum printer.

CHAPTER 15: Program Step Trace via Printer Control (Autoprint Only) 164

A highly efficient editing and debugging aid is introduced in the form of a step-wise program operation trace.

PERIPHERALS:

A brief description of peripherals available for all 600 models. 202

APPENDICES

Page

APPENDIX A: Direct Addressing 169

APPENDIX B: The 15 07 Subroutine Command 174

A procedure is given for using the new 15 07 code to increase the number of subroutines from 32 to 255.

APPENDIX C: The RUN/LEARN MODE features 177

This appendix contains a concise review of all the non-programmable debugging features.

APPENDIX D: Example of ALPHA RETURN 179

An example of the use of ALPHA RETURN is given for a subroutine which determines whether a keyed-in number is a prime.

APPENDIX E: The Wang User's ROM 181

APPENDIX F: Printing Symbol Identification 182

Identification of symbols used (i.e., JE means Jump if Error) in printout.

APPENDIX G: Glossary of Keyboard Operations 183

A brief description of the functions performed by keys, and switches.

APPENDIX H: 600 Library of Available Programs 191

Table of Contents of present printed Volumes of Programs. These volumes are constantly being updated and expanded by our professional 600 programming staff.

APPENDIX I: Program Codes . . . . . 194 & 201

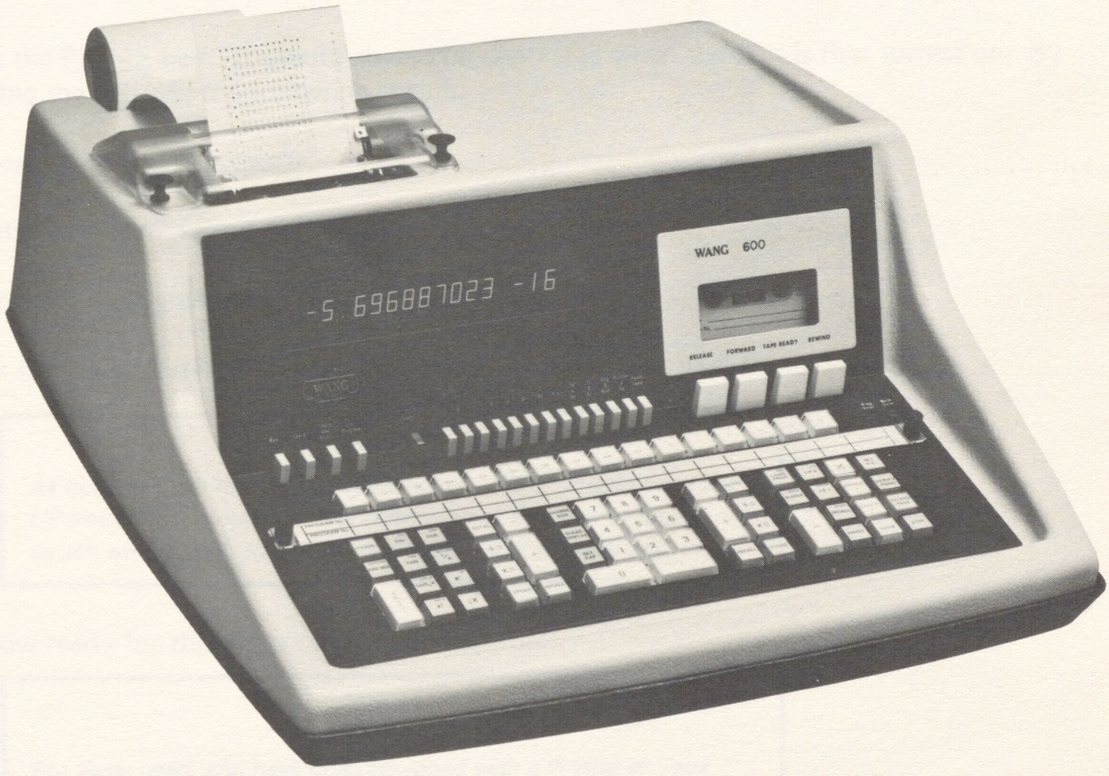
A list of machine codes for keystrokes and combinations of keys and switches for various functions.

INDEX 195

600 KEYBOARD FOLDOUT PAGE . . . . . Last Page

PART I

OPERATING YOUR WANG 600 MANUALLY



Chapter 1

PREPARING THE 600 CALCULATOR FOR MANUAL OPERATION

To prepare the machine for manual operation, the user's initial procedure should be as follows:

1. Plug the unit into power source.

2. Turn the "Power Switch" ON (located at lower left-hand corner of rear panel). (Allow machine to warm-up a few seconds.)

3. Depress the PRIME key. It should always be depressed when the unit is first turned on. By depressing the PRIME, the following occurs:

   A. Clears the Display Register.

   B. Sets the Program Counter to 0000.

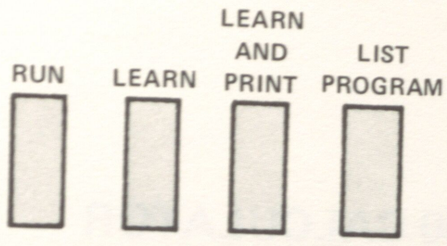
   C. Resets the Program Error Indicator and Machine Error Indicator.

4. Depress the red CLEAR switch to clear registers [00] thru [15] .

> Note:
>
> *At this time, the function selector switch* **(SP)** *should be in the UP position. A more elaborate discussion of this special function switch will be covered in Chapters 4 and 14 of this manual.*

The 600 is now ready for the mode of operation desired.

> Note:
>
> ***For those users who have units equipped with a Printer or Tape Cassette, see Chapters 13 and 14.***



(Push-down Lock-in switches)

The 600 has five modes of operation. The selection depends upon the application in process. Lock-in switches activate the different modes of operation and they are located on the top left-hand corner of the keyboard.

The RUN, LEARN, and RUN/LEARN MODES are used most extensively in manual keystroke operations; LEARN-PRINT and LIST-PROGRAM modes are used when hard-copy output is desirable.

RUN MODE

The RUN MODE switch is the most widely used Mode switch. All manual keyboard and program control calculations are done in the RUN MODE.

LEARN MODE

The LEARN MODE is used to program instructions into memory. *See page 60.*

RUN/LEARN MODE

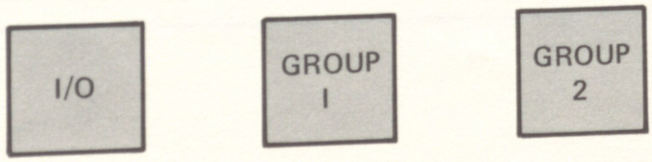
The RUN/LEARN mode is used primarily during the "debugging" of a program. *See page 97.*

LEARN-PRINT MODE (For printing models only)

The LEARN-PRINT mode is used when hard-copy output of the program is desired. *See page 146.*

LIST-PROGRAM MODE (For printing models only)

With the 600 in the LIST-PROGRAM mode, depressing the **PRIME** and **GO** keys will automatically list the program steps, the instruction codes, and operation symbols. *See page 147.*



Note:

The three keys, I/O, GROUP 1 and GROUP 2, are used only when addressing peripheral output and storage equipment.

Chapter 2

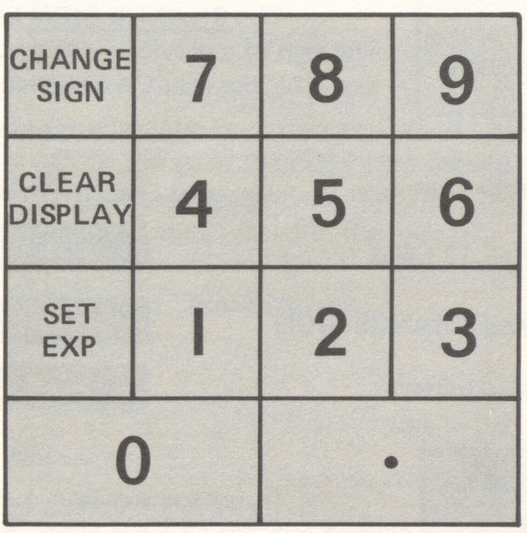
DATA ENTRY KEYS AND RIGHT AND LEFT REGISTERS

Entering numbers into the system via the ten-key keyboard is the same process used as writing numbers on a piece of paper. The same sequence of actions takes place with the decimal point keyed in where desired.

This section will discuss the procedure of entering numbers into the system and their associated arithmetic functions.

The 0 through 9 keys are used for entering data into the system. A depression of any one of these keys will generate the number into the display register. Such functions as changing the sign of a number, clearing the display, or setting the exponent are all one-keystroke operations. For numbers between .0000000001 and 1.000 000 000 the decimal point retains its natural position. However, for numbers indexed outside this range, the SET EXP key should be used. The use of the SET EXP key will automatically express the numbers in true scientific notation. By true scientific notation, the decimal point is placed after the first digit in the display.

TEN-KEY KEYBOARD



The use of the CHANGE SIGN, CLEAR DISPLAY, SET EXP keys and the 0 – 9 and decimal point will be illustrated by the following examples.

Unless otherwise specified, all operations will be performed in floating point mode with the Autoprint off. To turn Autoprint "OFF", key SHIFT, ALPHA (α), SHIFT, ALPHA (α).

SET EXP

The range of the exponent is –99 to +99. After depressing the SET EXP key, the next two keystrokes will determine the value of the exponent. If you depress more than two keys, the exponent will be determined by the last two keystroke depressions.

Keystroke	Display	Actual Printout
1. Key 45.67	+45.67	
2. Key SET EXP	+45.67 +00	
3. Key 2	+45.67 +02	
4. Key 3	+45.67 +23	
5. Key STORE RIGHT	+4.567000000 +24	
6. Key PRINT	+4.567000000 +24	4.567000000 + 24

CHANGE SIGN

The CHANGE SIGN key can be used before, during or after the entry of a number. The sign of a previously computed result may be changed by depressing the CHANGE SIGN key after the result is obtained.

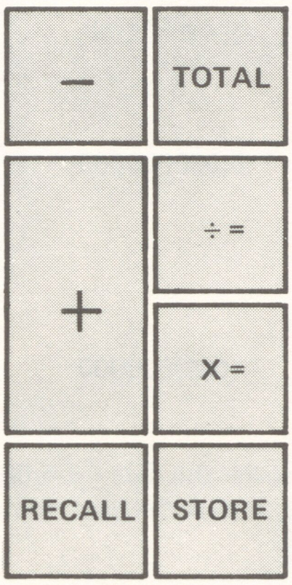
Keystroke Operation	Display	Actual Printout
1. Key 123.456	+123.456	
2. Key CHANGE SIGN	-123.456	
3. Key PRINT	-123.4560000	– 123.4560000

CLEAR DISPLAY

If, for some reason, a number being keyed into the system is incorrect and should be cleared out, the CLEAR DISPLAY key will clear the contents of the display register.

Keystroke Operation	Display	Actual Printout
1. Key 123.456	+123.456	
2. Key CLEAR DISPLAY	+	
3. Key PRINT	+0.000000000	0.000000000

Our discussion on the use of these keys will be confined to the floating mode position.



RIGHT [14] AND LEFT [15] REGISTERS

Seven keys are located on both sides of the ten-key keyboard. These keys are used to generate basic arithmetic operations. As such, they can be used as two individual calculators and/or two storage registers.

Examples will point out the useful features of the right- and left-hand calculators in conjunction with the ten-key keyboard.

See Special Register Chapter 4 for full explanation of the right and left registers with Registers [14] and [15].

STORE

Storing a number into the right or left register is a one-keystroke operation: key in the data and depress the STORE key. The data will be stored in the designated register (left or right) and will also remain in the Display Register. (Example storing data in the Right and Left Registers.) This will override any previous data placed in the register.

Keystroke Operation	Display	Actual Printout
(Right Register)		
1. Key **5**	+5	
2. Key **STORE RIGHT**	+5.000000000	
3. Key **PRINT**	+5.000000000	5.000000000
(Left Register)		
1. Key **10**	+10	
2. Key **STORE LEFT**	+10.00000000	
3. Key **PRINT**	+10.00000000	10.00000000

RECALL

To continue with our example, we would now like to recall the data from the right and left registers. To do this, one depression of the RECALL key is all that is needed.

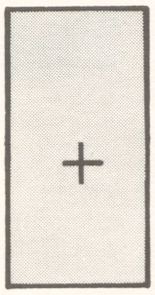
Keystroke Operation	Display	Actual Printout
1. Key **RECALL RIGHT**	+5.000000000	
2. Key **PRINT**	+5.000000000	5.000000000
1. Key **RECALL LEFT**	+10.00000000	
2. Key **PRINT**	+10.00000000	10.00000000



The TOTAL key has two major functions: To transfer the data from the right or left register to the display (depending on the choice), and to clear the data from the right or left register.

Keystroke Operation	Display	Actual Printout
1. Key **TOTAL RIGHT**	+5.000000000	
2. Key **PRINT**	+5.000000000	5.000000000
3. Key **RECALL RIGHT**	+0.000000000	
(Right Register now cleared)		
4. Key **PRINT**	+0.000000000	0.000000000
1. Key **TOTAL LEFT**	+10.00000000	
2. Key **PRINT**	+10.00000000	10.00000000
3. Key **RECALL LEFT**	+0.000000000	
(Left Register now cleared)		
4. Key **PRINT**	+0.000000000	0.000000000

The last example illustrated the use of the TOTAL key and how by using the TOTAL key the right and left registers were cleared. The following examples will cover the keystroke operation dealing with the +, −, ÷ =, and X = keys. We will start our discussion of these keys with no data in either the right or left register.



Addition is simply a matter of keying in your data and depressing the plus (+) key. To add up a list of numbers and to print the sum is obtained by the following keystroke operations:

Keystroke Operation	Display	Actual Printout
(Right Register)		
1. Key **5**	+5	
2. Key + or Key **STORE**	+5.000000000	
3. Key **6**	+6	
4. Key +	+11.00000000	
5. Key **7**	+7	
6. Key +	+18.00000000	
7. Key **PRINT**	+18.00000000	18.00000000

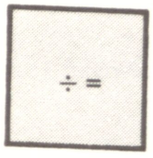
It is obvious at this point that this operation acts as a very efficient adding machine. Our next operation will discuss the procedure for subtracting.



From our previous example the right register now contains 18. To subtract 5 from the right register, the following keystroke operations are required:

Keystroke Operation	Display	Actual Printout
(Right Register)		
1. Key **5**	+5	
2. Key –	+13.00000000	
3. Key **PRINT**	+13.00000000	13.00000000

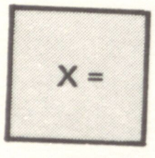
The right register now contains 13 and is also in the display.



To divide, it is just a matter of keying in your divisor and dividing.

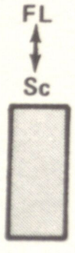
Keystroke Operation	Display	Actual Printout
(Right Register)		
To divide by 3		
1. Key **3**	+3	
2. Key ÷ =	+4.333333333	
3. Key **PRINT**	+4.333333333	4.333333333

The right register and the display now contain 4.333333333



To multiply the contents of the right register by 3, the following keystrokes are needed:

Keystroke Operation	Display	Actual Printout
(Right Register)		
1. Key **3**	+3	
2. Key **X** =	+12.99999999	
3. Key **PRINT**	+12.99999999	12.99999999



As stated in the beginning of this section, all operations were done in the floating decimal point mode. To initiate the unit for true scientific notation, the lock-in switch located on the top portion of the keyboard is used. An example will best illustrate the use of this key.

From the previous example, 12.99999999 now appears in the display. Lock in the scientific mode switch and the display will now read

+1.299999999 +01

Example 1 Floating-Point Mode/Scientific Notation

RUN MODE	DISPLAY
1. Key **1250500.7**	+1250500.7
2. Key **STORE LEFT**	+1.250500700+06
3. Key **1033.044**	+1033.044
4. Key **PLUS LEFT**	+1.251533744+06

Shift the switch to Floating-Point Mode. The decimal point will retain its correct position, +1251533.744

Example 2 Calculate the value of X in the following expression

$$X = \frac{5.168 + 20\left(6 \div 4.61 + 7 \times 8314 - \frac{61.7}{.516}\right)}{\frac{63.9}{.114}\left(8.2 \times 6.3 - \frac{4.11}{2} \times 83.1\right)}$$

Keystroke Operation	Comment
1. Key **6**	
2. Key **STORE RIGHT**	
3. Key **4.61**	
4. Key **÷ RIGHT**	6/4.61 is calculated in right register
5. Key **7**	
6. Key **STORE LEFT**	6/4.61 is stored in left
7. Key **8314**	
8. Key **X LEFT**	7 x 8314 is calculated in left
9. Key **+ RIGHT**	6/4.61 +7 x 8314 is now in right
10. Key **61.7**	
11. Key **STORE LEFT**	

Keystroke Operation	Comment
12. Key **.516**	
13. Key ÷ **LEFT**	61.7/.516 is calculated in left
14. Key – **RIGHT**	6/4.61 + 7 x 8314 – 61.7/.516 is now in right
15. Key **20**	
16. Key **X RIGHT**	20 is multiplied by everything in right yielding 20 (6/4.61 + 7 x 8314 – 61.7/.516)
17. Key **5.168**	
18. Key + **RIGHT**	The numerator 5.168 + 20 (6/4.61 + 7 x 8314 – 61.7/.516) is in right register.

In calculating the denominator of the expression we need to use both the left and right registers. If we use the right register we will destroy or lose the value of the numerator in Step 18.

Although the sixteen white colored registers numbered [00] [01] [02] [03] . . . [15] have not been explained yet, it will be necessary to use one of them for storage.

19. **Depress** [St] **Function Selector Switch on top of keyboard**

20. **Depress** [00] The numerator is now stored in [00] as well as the right register.

Now let us calculate the denominator.

21. Key **8.2**
22. Key **STORE RIGHT**
23. Key **6.3**
24. Key **X RIGHT** 8.2 x 6.3 is stored in right
25. Key **4.11**
26. Key **STORE LEFT**
27. Key **2**

Keystroke Operation	Comment
28. Key ÷ **LEFT**	4.11/2 is calculated
29. Key **83.1**	
30. Key **X LEFT**	4.11/2 x 83.1 is calculated
31. Key – **RIGHT**	8.2 x 6.3 – 4.11/2 x 83.1 is stored in right
32. Key **63.9**	
33. Key **X RIGHT**	
34. Key **.114**	
35. Key ÷ **RIGHT**	The denominator 63.9/.114 (8.2 x 6.3 – 4.11/2 x 83.1) is stored in right.

At this point the denominator is in the right register and the numerator is in register [00]

36. **Depress [Re] switch to prepare for recalling numerator**

37. [00]	Numerator is in display
38. Key **STORE LEFT**	Numerator is in left register
39. Key **RECALL RIGHT**	Denominator is in display
40. Key ÷ **LEFT**	Answer for X is displayed

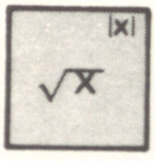
Answer: –17.39844546

Chapter 3

SCIENTIFIC AND ENGINEERING KEYS

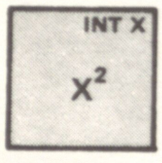
This chapter of the 600 Manual will cover the use of many technically oriented and sophisticated calculating applications. You will find as you study this chapter that some of the most powerful calculator operations in existence today are found on the 600 keyboard. Most of these operations require just one depression of the key. All of the operations discussed in this chapter affect only the display – and none of the 16 basic storage registers

UPPERCASE AND LOWERCASE FUNCTION KEYS



> Note:
>
> *The $\sqrt{X}$ key is a lowercase key; the |X| is the uppercase key. Examples and illustrations in this chapter and Chapter 5 will discuss in detail the use of these keys for both uppercase and lowercase functions.*

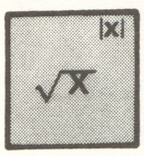
SQUARE



To square any number, it is simply a matter of keying in the data and depressing the $\underline{X^2}$ key.

Keystroke Operation	Display	Actual Printout
1. Key 5	+5	
2. Key X^2	+25.00000000	
3. Key PRINT	+25.00000000	25.00000000

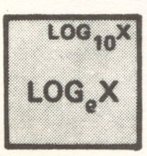
SQUARE ROOT



To obtain the square root of a number:

Keystroke Operation	Display	Actual Printout
1. Key **5**	+5	
2. Key $\sqrt{X}$	+2.236067977	
3. Key **PRINT**	+2.236067977	2.236067977

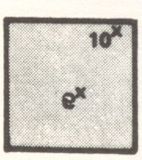
NATURAL LOG



To generate the natural log of a number:

Keystroke Operation	Display	Actual Printout
1. Key **5**	+5	
2. Key **Log$_e$ X**	+1.609437912	
3. Key **PRINT**	+1.609437912	1.609437912

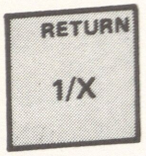
ANTI-LOG



To raise e to a power:

Keystroke Operation	Display	Actual Printout
1. Key **5**	+5	
2. Key **e^x**	+148.4131591	
3. Key **PRINT**	+148.4131591	148.4131591

RECIPROCAL



To generate the reciprocal of a number:

Keystroke Operation	Display	Actual Printout
1. Key 5	+5	
2. Key 1/X	+.2000000000	
3. Key PRINT	+.2000000000	.2000000000

RADIANS-TO-DEGREES CONVERSION



To convert radians to degrees:

Keystroke Operation	Display	Actual Printout
1. Key 5 Radians	+.5	
2. Key Rad→Deg	+286.4788975	
3. Key PRINT	+286.4788975	286.4788975



If the degree-to-radian switch is UP, the number that appears in the display will be interpreted as degrees.

If the degree-to-radian switch is DOWN, the number that appears in the display will be interpreted as radians.

When performing SIN, COS, and TAN functions, the RAD→DEG switch must be selected. The results of data keyed in must be less than 10 radians or no greater than 572°.

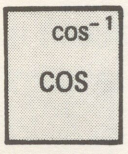
SINE



To generate the sine of a number:

Keystroke Operation	Display	Actual Printout
1. Key **30** (degrees)	+30	
2. Key **SIN**	+.5000000000	
3. Key **PRINT**	+.5000000000	.5000000000

COSINE



To generate the cosine of a number:

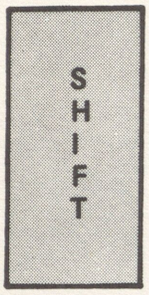
Keystroke Operation	Display	Actual Printout
1. Key **30** (degrees)	+30	
2. Key **COS**	+.8660254037	
3. Key **PRINT**	+.8660254037	.8660254037

TANGENT



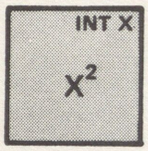
To generate the tangent of a number:

Keystroke Operation	Display	Actual Printout
1. Key **30** (degrees)	+30	
2. Key **TAN**	+.5773502691	
3. Key **PRINT**	+.5773502691	.5773502691



Keys illustrated thus far in this section are all lowercase function keys. To execute the dual function of uppercase functions, a key called SHIFT (located at the extreme lower left-hand corner of the keyboard) is needed. The light on the SHIFT key will indicate when the unit is in the SHIFT mode. All uppercase functions must be preceded by the SHIFT key.

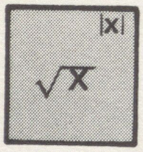
INTEGER



The integer function is similar to all others previously covered except that the SHIFT key must be depressed.

Keystroke Operation	Display	Actual Printout
1. Key **5.123**	+5.123	
2. Key **SHIFT**	+5.123	
3. Key **Int X**	+5.000000000	
4. Key **PRINT**	+5.000000000	5.000000000

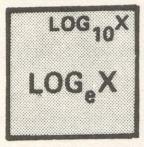
ABSOLUTE VALUE



To generate the absolute value of a number:

Keystroke Operation	Display	Actual Printout
1. Key **CHS 5.236**	- 5.236	
2. Key **SHIFT**	- 5.236	
3. Key **|X|**	+5.236000000	
4. Key **PRINT**	+5.236000000	5.236000000

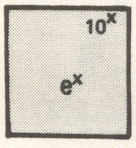
BASE 10 LOG



To generate base 10 logarithms:

Keystroke Operation	Display	Actual Printout
1. Key **5**	+5	
2. Key **SHIFT**	+5	
3. Key **$Log_{10}X$**	+.6989700043	
4. Key **PRINT**	+.6989700043	.6989700043

TEN TO A POWER



To raise 10 to a power:

Keystroke Operation	Display	Actual Printout
1. Key **5**	+5	
2. Key **SHIFT**	+5	
3. Key **10^x**	+100000.0000	
4. Key **PRINT**	+100000.0000	100000.0000

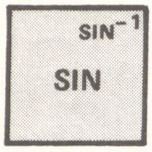
DEGREES-TO-RADIANS CONVERSION



To convert degrees to radians.

Keystroke Operation	Display	Actual Printout
1. Key **286.4788975**		
2. Key **SHIFT**	+286.4788975	
3. Key **Deg → Rad**	+4.999999998	
4. Key **PRINT**	+4.999999998	4.999999998

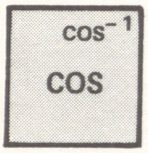
ARCSINE



To generate Arcsine:

Keystroke Operation	Display	Actual Printout
1. Key **.5**	+.5	
2. Key **SHIFT**	+.5	
3. Key **SIN⁻¹**	+30.00000000	
4. Key **PRINT**	+30.00000000	30.00000000

ARCCOSINE



To generate Arccosine:

Keystroke Operation	Display	Actual Printout
1. Key **.8660254037**	+.8660254037	
2. Key **SHIFT**	+.8660254037	
3. Key **COS⁻¹**	+30.00000000	
4. Key **PRINT**	+30.00000000	30.00000000

ARCTANGENT



To generate Arctangent:

Keystroke Operation	Display	Actual Printout
1. Key **.5773502961**	+.5773502961	
2. Key **SHIFT**	+.5773502961	
3. Key **TAN⁻¹**	+30.00000115	
4. Key **PRINT**	+30.00000115	30.00000115

FINDING ANGLES BY TRIGONOMETRIC RATIOS

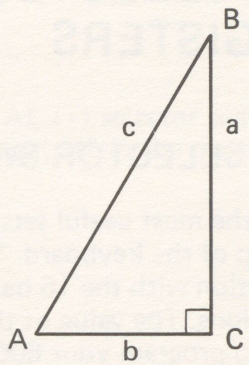
Find A in each of the following:

In a right triangle

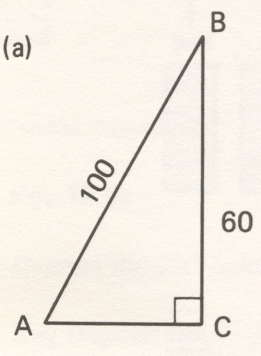
$$\sin A = \frac{a}{c}, \cos A = \frac{b}{c}, \tan A = \frac{a}{b}$$

or

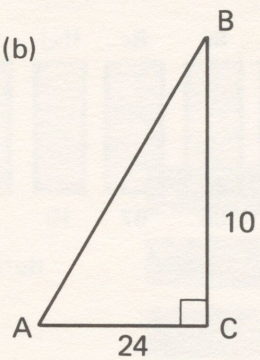
$$A = \sin^{-1}\frac{a}{c}, A = \cos^{-1}\frac{b}{c}, A = \tan^{-1}\frac{a}{b}$$



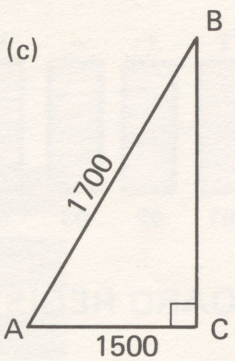
(a)



(b)



(c)



(a) SIN^{-1} of A

Key

1. **60**
2. **ST LEFT**
3. **100**
4. **÷ LEFT**
5. **SHIFT**
6. **SIN^{-1}**

A = 36.869

(b) TAN^{-1} of A

Key

1. **10**
2. **ST LEFT**
3. **24**
4. **÷ LEFT**
5. **SHIFT**
6. **TAN^{-1}**

A = 22.619

(c) COS^{-1} of A

Key

1. **1500**
2. **ST LEFT**
3. **1700**
4. **÷ LEFT**
5. **SHIFT**
6. **COS^{-1}**

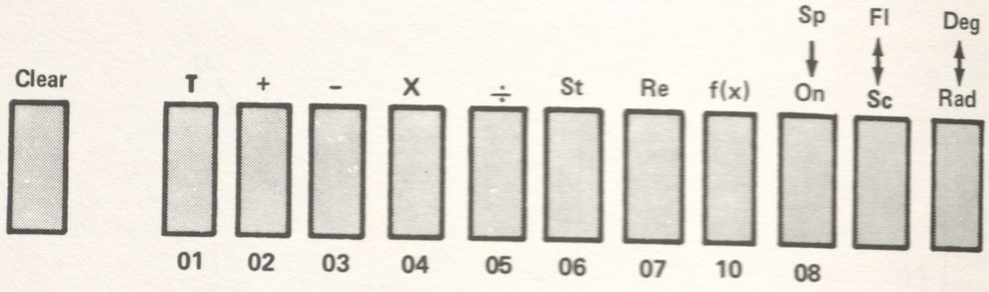
A = 28.072

Chapter 4

SPECIAL SELECTOR SWITCHES AND REGISTERS

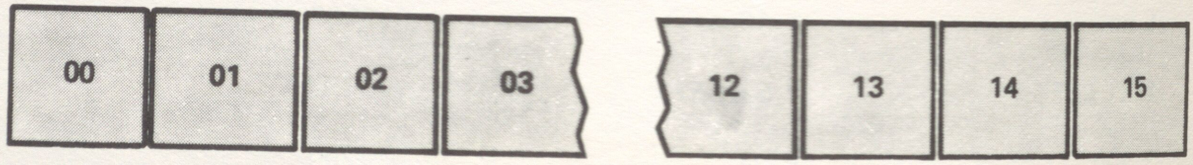
FUNCTION SELECTOR SWITCHES

One of the most useful sets of keys and switches on the 600 is the set of Function Selector Switches found at the top of the keyboard. You will find, in this and later chapters, that when these switches are used in conjunction with the 16 basic register keys, many procedures in your programs become simple one-step operations. The value of this feature will become evident in the second part of this manual, which describes how to program your 600 Calculator.



THE 16 BASIC KEYBOARD REGISTERS

Suppose you had a problem or equation which required you to repeatedly use several different numbers several times (Example 8 on page 76 is a good example of this.) Each time you come to the required step, you can key in each number again and again; or, you can use several of the 600's sixteen Basic Data Registers to store each of these numbers, ready to be used at the touch of a key. This chapter will show you how to save time and effort by making use of this keyboard feature.



Our work in the previous chapter dealt with entering data into the right- and left-hand registers. We will now discuss the use of all the basic sixteen register keys [00] – [15], located on the top of the keyboard, and their associated arithmetic functions.

This section of the keyboard has many calculating functions, but for the time being, we will consider the use of these keys as they are designated. Such functions as T, +, –, X, ÷, ST, and RE will be illustrated with the Function Selector Switch (Sp) covered last.

All operations will be performed in floating mode.

> **Note:**
>
> *This section of the 600 Manual will deal with the (T), (+), (–), (X), (÷), (St), (Re), and (Sp) Function Selector Switches, and their use with the 16 Basic Keyboard Registers only. For an explanation of the use these Function Selector Switches with any of the extended storage registers – i.e., registers [16] and beyond – see chapters 10 and 11, which explain data storage and Indirect Addressing.*

CLEAR

Cl



The red colored CLEAR switch (located to the left of the TOTAL (T) selector switch) will clear the contents of Registers [00] through [15] . It will not affect the contents of the display.

STORE

St



Storing data in Register [05] requires the following keystroke operations:

Keystroke Operation	Display	Actual Printout
1. Key **12.45**	+12.45	
2. Depress the **(St) Selector Switch**	+12.45	
3. Key **Register** [05]	+12.45000000	
4. Key **PRINT**	+12.45000000	12.45000000

12.45 is now stored in Register [05] and also appears in the display.

To store data in Register [08] , the following keystroke operations are required:

Keystroke Operation	Display	Actual Printout
1. Key **365.65**	+365.65	
(St) Selector Switch is in the down position from the previous example.		
2. Key **Register** [08]	+365.6500000	
3. Key **PRINT**	+365.6500000	365.6500000

(365.65 is now stored in Register [08] , and is also in the display.)

RECALL

Re



To recall the data that was previously stored in Registers 05 and 08 , the following procedure should be used:

Keystroke Operation	Display
Recall **Register** 05	
1. Depress (Re) Selector Switch	
2. Key **Register** 05	+12.45000000
Recall Register 08	
1. Key **Register** 08	+365.6500000

TOTAL

T

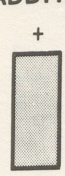


To recall and clear any register , the (T) Selector Switch is used.

Keystroke Operation	Display
Recall and Clear 05	
1. Depress (T) Selector Switch	
2. Key **Register** 05	+12.45000000
Recall and Clear 08	
1. Key **Register** 08	+365.6500000

Both Registers 05 and 08 should now be cleared. To check this, depress the (Re) Selector Switch and key Register 05 and Register 08 . Both operations should display zeros.

ADDITION



To add numbers using the Function Selector Switch is simply a matter of keying in the data, depressing the (+) Function Selector Switch and designating the desired register.

It is evident that 16 individual calculators are available through the use of registers [00] through [15] . Illustrations will show the use of (+), (−), (X) and (÷) Function Selector Switches in a calculating routine.

Example: To add 3 + 4 + 5 in Register [06].

Keystroke Operation	Display	Actual Printout
1. Depress **CLEAR**		
2. Depress (+) **Selector Switch**	+	
3. Key **3**	+3	
4. Key **Register [06]**	+3.000000000	
5. Key **4**	+4	
6. Key **Register [06]**	+7.000000000	
7. Key **5**	+5	
8. Key **Register [06]**	+12.00000000	
9. Key **PRINT**	+12.00000000	12.00000000

Register [06] now contains 12 and the display also contains 12.00000000

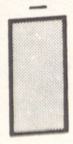
MULTIPLICATION



To multiply the contents of Register [06] by 5:

Keystroke Operation	Display	Actual Printout
1. Depress the (X) **Selector Switch**		
2. Key **5**	+5	
3. Key **Register [06]**	+60.00000000	
4. Key **PRINT**	+60.00000000	60.00000000

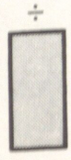
SUBTRACTION



To subtract 28 from the contents of Register 06:

Keystroke Operation	Display	Actual Printout
1. Depress the (—) Selector Switch		
2. Key 28	+28	
3. Key Register 06	+32.00000000	
4. Key PRINT	+32.00000000	32.00000000

DIVISION



From the previous example, Register 06 now contains 32. To divide 2 into 32, the operation would be as follows:

Keystroke Operation	Display	Actual Printout
1. Depress the (÷) Selector Switch		
2. Key 2	+2.	
3. Key Register 06	+16.00000000	
4. Key PRINT	+16.00000000	16.00000000

The answer will now appear in the Display and also in Register 06.

SPECIAL SWITCH (EXCHANGING DATA)

Sp



The Special Switch located beside the f(x) key gives a user the feature of exchanging the data in any register with the Display. The example below illustrates the procedure used to perform the exchange operation.

Display reads 3.00000000
Contents of Register [04] are 4.000000000.

Keystroke Operation

1. [Sp] Selector Switch down
2. [St] Selector Switch down
3. Key Reg [04]

The above example will exchange the contents of the display with register [04].

> *CAUTION:*
>
> *The (Sp) switch should be put back in the up position when not being used.*

f(x)



The main function of the f(x) selector switch is for the execution of subroutines; however, other functions can be performed by using the f(x) switch. (See part II Programming.) For the time being, we will concern ourselves with these functions and will cover the execution of subroutines later.

In this section, three main areas will be discussed. They are generating Pi (π), moving the decimal to the left, and moving the decimal to the right.

To generate Pi (π), the keystrokes would be:

GENERATING π

Keystroke Operation	Display	Actual Printout
1. Key SHIFT		
2. Key α (ALPHA)		
3. Key Register [00]	3.141592653	
4. Key PRINT	3.141592653	3.141592653

SHIFTING DECIMAL POINT TO THE RIGHT OR LEFT

SHIFTING RIGHT

f(x) down

SHIFT
α
01 } 1 place

SHIFT
α
02 } 2 places

SHIFT
α
03 } 3 places

.
.
.

SHIFT
α
15 } 15 places

SHIFTING LEFT

f(x) down

SHIFT
α
SHIFT
01 } 1 place

SHIFT
α
SHIFT
02 } 2 places

SHIFT
α
SHIFT
03 } 3 places

.
.
.

SHIFT
α
SHIFT
15 } 15 places

To continue with the data that was generated from the last operation, we can now proceed to move the decimal point up to a maximum of 15 places to the right (and force it into scientific notation).

We will use as our example the data in the display 3.141592653. (Remember f(x) switch DOWN.)

Example 1

Move the decimal point five places to the right.

Keystroke Operation

1. Key **SHIFT**
2. Key **α(ALPHA)**
3. Key **Register** 05
4. Key **PRINT**

Actual Printout

314159.2653

Example 2

Move the decimal point back 5 places to the left.

Keystroke Operation	Actual Printout
1. Key **SHIFT**	
2. Key **α (ALPHA)**	
3. Key **SHIFT**	
4. Key **Register** 05	
5. Key **PRINT**	3.141592653

The procedures just illustrated can move the decimal point to a maximum of 15 places, either to the right or left.

Example 3

Keystroke Operation	Display
1. Key **1.2345**	
2. Key **SHIFT**	
3. Key **α (ALPHA)**	
4. Key **Register** 03	1234.500000

Example 4

Keystroke Operation	Display
1. Key **SHIFT**	
2. Key **α (ALPHA)**	
3. Key **SHIFT**	
4. Key **Register** 03	1.234500000

SUMMARY

As you have seen, the Special Selector Switches and Data Register Keys are very useful in handling data while the 600 is used under manual control. However, you will find that these keys have their greatest use under program control, where they will save you programming steps time and time again. The purpose of introducing these features at this time is to enable you to familiarize yourself with their operation.

Chapter 5

REVIEW BY EXAMPLES

The examples which follow with accompanying comments will serve to provide a review of keyboard operations explained in the first four chapters and to provide various techniques for intermixing the proper keys to solve realistic user problems.

For programming examples and techniques refer to Part II of this manual.

Example 1

Calculate $c = \sqrt{a^2 + b^2}$ for a = 3, b = 4 and a = 6, b = 7.

For a = 3, b = 4

Keystroke	Comment
1. 3	
2. x^2	
3. STORE RIGHT	3^2
4. 4	
5. x^2	4^2
6. + RIGHT	$3^2 + 4^2$
7. $\sqrt{x}$	$\sqrt{3^2 + 4^2}$

Answer: c = 5.000000000

For a = 6, b = 7

Keystroke	Comment
1. 6	
2. x^2	6^2
3. STORE LEFT	6^2
4. 7	
5. x^2	7^2
6. + LEFT	$6^2 + 7^2$
7. $\sqrt{x}$	$\sqrt{6^2 + 7^2}$

Answer: c = 9.219544457

Example 2

Calculate $c = \sqrt{a^2 + b^2 - 2ab\,CosC}$ for $a = 3.1$, $b = 6.8$, $C = 83^o$.

Keystroke	Comment
1. 3.1	
2. x^2	
3. STORE RIGHT	a^2
4. 6.8	
5. x^2	
6. + RIGHT	$a^2 + b^2$
7. RAD ⟶ Degree switch up	
8. 83	
9. CosX	
10. STORE LEFT	CosC
11. 2	
12. X LEFT	2 CosC
13. 3.1	a
14. X LEFT	2a CosC
15. 6.8	b
16. X LEFT	2ab CosC
17. − RIGHT	$a^2 + b^2 - 2ab\,CosC$
18. $\sqrt{x}$	$c = \sqrt{a^2 + b^2 - 2ab\,CosC}$

Answer $c = 7.121235039$

Example 3

Determine y for x = 34.1158 in the formula $y = (x - 1.105)^5 + \frac{31x^3}{e^x} - 4(x+e^{x-1})^5$

Keystroke	Comment
1. 34.1158	x
2. STORE RIGHT	x
3. 1.105	
4. – RIGHT	$x - 1.105$
5. Log_e x	
6. STORE RIGHT	$Log_e (x - 1.105)$
7. 5	
8. X RIGHT	$5Log_e (x - 1.105)$
9. e^x	$(x - 1.105)^5$
10. (St) SWITCH DOWN	
11. 00	$(x - 1.105)^5$ stored in 00
12. 34.1158	
13. Log_e x	$Log_e x$
14. ST RIGHT	
15. 3	
16. X RIGHT	$3Log_e x$
17. e^x	x^3
18. ST RIGHT	x^3 stored in right register
19. 31	
20. X RIGHT	$31x^3$ stored in right register
21. 34.1158	

Keystroke	Comment
22. e^x	e^x
23. ÷ RIGHT	$31x^3$ is divided by e^x and stored in right register and appears in display
24. (+) SWITCH DOWN	Prepares registers for addition
25. 00	$31x^3/e^x$ is added to $(x - 1.105)^5$
26. 34.1158	
27. ST RIGHT	
28. 1	
29. – RIGHT	$x - 1$
30. e^x	e^{x-1}
31. ST RIGHT	e^{x-1} stored in right register
32. 34.1158	
33. + RIGHT	$x + e^{x-1}$
34. $Log_e x$	
35 ST RIGHT	$Log_e (x + e^{x-1})$
36. 5	
37. X RIGHT	$5Log (x + e^{x-1})$
38. e^x	$(x + e^{x-1})^5$
39. ST RIGHT	
40. 4	
41. X RIGHT	$4(x + e^{x-1})^5$
42. (–) SWITCH DOWN	Prepares registers for subtraction
43. 00	$4(x + e^{x-1})^5$ is subtracted from $(x - 1.105)^5 + \frac{31x^3}{e^x}$

Answer: y = –3.251666586 + 72

Example 4

$$A = \sqrt{5.1^2 + 6.1\sqrt{82}} + \frac{\text{Sin}^2\,(31.8^\circ)}{|\text{Tan } 121.4^\circ|}$$

Keystroke	Comment
1. Key **82**	
2. Key $\sqrt{x}$	
3. Key **STORE RIGHT**	$\sqrt{82}$ in right register
4. Key **6.1**	
5. Key **x RIGHT**	$6.1\sqrt{82}$
6. Key **5.1**	
7. Key x^2	5.1^2 in display
8. Key **+ RIGHT**	$5.1^2 + 6.1\sqrt{82}$ in right register
9. Key $\sqrt{x}$	$\sqrt{5.1^2 + 6.1\sqrt{82}}$ only in display
10. Key **STORE RIGHT**	$\sqrt{5.1^2 + 6.1\sqrt{82}}$ is stored in right register
11. Key **31.8**	
12. Key **SinX**	with Rad → Degree switch up key SinX
13. Key x^2	Sin^2 31.8
14. Key **STORE LEFT**	Sin^2 31.8 in left register
15. Key **121.4**	
16. Key **TanX**	Tan 121.4
17. **SHIFT**	

Keystroke	Comment
18. Key $\lvert x \rvert$	Absolute value of Tan 121.4 is a positive number
19. Key ÷ LEFT	Sin^2 (31.8) is divided by $\lvert \text{Tan } 121.4 \rvert$
20. Key + RIGHT	and added to $\sqrt{5.1^2 + 6.1\sqrt{82}}$

Answer: A = 9.183256958

Example 5

$$B = \frac{\text{Sin } (1 - .006)^5}{\sqrt{\text{Cos}^2\ 8.1 + \text{Sin } 4.3}} - \frac{\lvert 3.11^2 - 15.005 \rvert^2}{(1 - .0078)^3}$$

Keystroke	Comment
1. Key 1	
2. Key STORE RIGHT	
3. Key .006	
4. Key – RIGHT	1 – .006 in right register
5. Key $\text{Log}_e x$	Log_e (1 – .006)
6. Key STORE RIGHT	
7. Key 5	
8. Key X RIGHT	5 Log_e (1 – .006) in right register
9. Key e^x	$(1 - .006)^5$ in right register
10. Key SinX	
11. Key STORE RIGHT	Sin $(1 - .006)^5$ in right register
12. Key 4.3	
13. Key SinX	Sin 4.3
14. Key STORE LEFT	Sin 4.3 in left register
15. Key 8.1	

Keystroke	Comment
16. Key **CosX**	
17. Key x^2	Cos^2 8.1 is added
18. Key **+ LEFT**	to Sin 4.3
19. Key $\sqrt{x}$	$\sqrt{\text{Cos}^2\ 8.1\ +\ \text{Sin}\ 4.3}$
20. Key **÷ RIGHT**	First term is now stored in right register after division
21. Key **3.11**	
22. Key x^2	
23. Key **STORE LEFT**	3.11^2 in left register
24. Key **15.005**	
25. Key **– LEFT**	3.11 – 15.005
26. Key **SHIFT**	
27. Key | x |	
28. Key x^2	$\lvert 3.11 - 15.005 \rvert^2$
29. Key **STORE LEFT**	$\lvert 3.11 - 15.005 \rvert^2$ in left register
30. **1**	
31. Key (St) 00	Since left and right registers are in use we must use new register 00 . Remember (St) Switch is down.
32. **.0078**	
33. – 00	1 – .0078 in 00 (–) SWITCH DOWN
34. $\text{Log}_e x$	Log_e (1 – .0078)
35. St 00	(St) SWITCH DOWN
36. **3**	
37. (x) 00	3Log_e (1 – .0078) (X) SWITCH DOWN
38. e^x	$(1 - .0078)^3$ in display

Keystroke	Comment
39. ÷ LEFT	$\lvert 3.11^2 - 15.005 \rvert^2$ is divided by $(1 - .0078)^3$
40. − RIGHT	and subtracted from first term of expression

Answer: B = −29.09934568

Example 6

For $x_1 = 40$, x_2 45, x_3 43.5, $x_4 = 51.7$, $x_5 = 47.8$, $x_6 = 46.61$, and $x_7 = 52$, multiply each x by 7.5 and divide by 12. Print the results. If your 600 is not equipped to print, omit PRINT command and read result in the display.

Keystroke	Comment
1. (St) Selector Switch Down	
2. 40	
3. 01	40 is in Register 01
4. 45	
5. 02	45 is in Register 02
6. 43.5	
7. 03	43.5 is in Register 03
8. 51.7	
9. 04	51.7 is in Register 04
10. 47.8	
11. 05	47.8 is in Register 05
12. 46.61	
13. 06	46.61 is in Register 06
14. 52	
15. 07	52 is in Register 07
16. 7.5	

Keystroke	Comment
17. STORE LEFT	
18. 12	
19. ÷ LEFT	7.5/12 is in Left Register
20. (X) SWITCH DOWN	Whenever [00] – [15] is pressed number in display will be multiplied by contents of register.
21. [01]	40 was previously stored here
22. PRINT	40 x 7.5/12 = 25.00000000 (Answer) 25 is now in [01]
23. RECALL LEFT	7.5/12 is in display
24. [02]	45 is multiplied by 7.5/12, since [X] is still down.
25. PRINT	45 x 7.5/12 = 28.12500000 (Answer) 28.125 in [02]
26. RECALL LEFT	
27. [03]	
28. PRINT	43.5 x 7.5/12 = 27.18750000 (Answer)
29. RECALL LEFT	
30. [04]	
31. PRINT	51.7 x 7.5/12 = 32.31250000 (Answer)
32. RECALL LEFT	
33. [05]	
34. PRINT	47.8 x 7.5/12 = 29.87500000 (Answer)
35. RECALL LEFT	
36. [06]	
37. PRINT	46.61 x 7.5/12 = 29.13125000 (Answer)
38. RECALL LEFT	
39. [07]	
40. PRINT	52 x 7.5/12 = 32.5000000 (Answer)

Example 7

Show that $\text{Sin}^2\theta + \text{Cos}^2\theta = 1$ for $\theta = 1^\circ, 15^\circ, 30^\circ, 60^\circ, 120^\circ, 170^\circ, 210^\circ, 315^\circ$

Keystroke	Comment
1. (St) SWITCH DOWN	
2. **1**	
3. [00]	1 is in [00]
4. **15**	
5. [01]	15 is in [01]
6. **30**	
7. [02]	30 is in [02]
8. **60**	
9. [03]	60 is in [03]
10. **120**	
11. [04]	120 is in [04]
12. **170**	
13. [05]	170 is in [05]
14. **210**	
15. [06]	210 is in [06]
16. **315**	
17. [07]	315 is in [07]
18. (Re) SWITCH DOWN	Each time one of the [00] – [15] keys is pressed the number in the register will be recalled to the display.
19. [00]	Recalls 1 to display
20. SinX	Sin 1
21. x^2	
22. STORE RIGHT	$\text{Sin}^2 1$

Keystroke	Comment
23. 00	Recalls 1°
24. CosX	
25. x^2	$\cos^2 1$
26. + RIGHT	$\sin^2 1 + \cos^2 1 = 1.000000000$ (Answer)
27. 01	15 is recalled
28. SinX	
29. x^2	$\sin^2 15$
30. STORE RIGHT	
31. 01	
32. CosX	
33. x^2	$\cos^2 15$
34. + RIGHT	$\sin^2 15 + \cos^2 15 = 1.000000000$ (Answer)
35. 02	30 is recalled
36. SinX	
37. x^2	$\sin^2 30$
38. STORE RIGHT	
39. 02	
40. CosX	
41. x^2	$\cos^2 30$
42. + RIGHT	$\sin^2 30 + \cos^2 30 = 1.000000000$ (Answer)
43. 03	60 is recalled
44. SinX	
45. x^2	
46. STORE RIGHT	
47. 03	
48. CosX	

Keystroke	Comment
49. x^2	
50. + RIGHT	$\sin^2 60 + \cos^2 60 = 1.000000000$ (Answer)
51. 04	120 is recalled
52. SinX	
53. x^2	$\sin^2 120$
54. STORE RIGHT	
55. 04	
56. CosX	
57. x^2	$\cos^2 120$
58. + RIGHT	$\sin^2 120 + \cos^2 120 = 1.000000000$ (Answer)
59. 05	170 is recalled
60. SinX	
61. x^2	$\sin^2 170$
62. STORE RIGHT	
63. 05	Recalls 170
64. CosX	
65. x^2	$\cos^2 170$
66. + RIGHT	$\sin^2 170 + \cos^2 170 = 1.000000000$ (Answer)
67. 06	210 is recalled
68. SinX	
69. x^2	$\sin^2 210$
70. STORE RIGHT	
71. 06	
72. CosX	
73. x^2	$\cos^2 210$
74. + RIGHT	$\sin^2 210 + \cos^2 210 = 1.000000000$ (Answer)

Keystroke	Comment
75. 07	315 is recalled
76. SinX	
77. x^2	$\sin^2 315$
78. STORE RIGHT	
79. 07	
80. CosX	
81. x^2	$\cos^2 315$
82. + RIGHT	$\sin^2 315 + \cos^2 315 = .9999999999$ (Answer)

Example 8

$$u = \sqrt{\left|\frac{2(1-Y-X)}{(1-Y)(1+\cos^2\theta)}\right|}\left[1 - \frac{\epsilon_1 \tan^2\theta}{4(1-Y)(1+\sec^2\theta)}\left(\frac{3+\sec^2\theta}{Y} + \tan^2\theta\right)\right]$$

This formula was taken from a Physics text on the theory of wave propagation.

Calculate u for y = .002, x = 1.048, ϵ_1 = .0034, and θ = 23.6°

Keystroke	Comment
1. (St) Selector Switch Down	Number in display will be stored whenever 00 - 15 keys are pressed.
2. .002	
3. 00	Y is stored in 00
4. 1.048	
5. 01	X is stored in 01
6. .0034	
7. 02	ϵ_1 is stored in 02
8. 23.6	
9. 03	θ is stored in 03
10. 1	

Keystroke	Comment
11. STORE RIGHT	
12. STORE LEFT	
13. (Re) Selector Switch Down	
14. 00	Y in display
15. – LEFT	1 – Y in left register
16. 03	Recalls θ to display
17. Cos	
18. X^2	$\cos^2 \theta$ in display
19. + RIGHT	$1 + \cos^2 \theta$
20. X LEFT	$(1 - Y)(1 + \cos^2 \theta)$ in left register
21. 1	
22. STORE RIGHT	1 in right register
23. 00	Recalls Y to display
24. – RIGHT	1 – Y in right register and in display
25. 01	X in display
26. – RIGHT	1 – Y – X in right register and display
27. 2	
28. X RIGHT	2 (1–Y–X) in display and right register
29. RECALL LEFT	$(1 - Y)(1 + \cos^2 \theta)$ in display
30. ÷ RIGHT	$\frac{2(1 - Y - X)}{(1 - Y)(1 + \cos^2 \theta)}$ in display and right register
31. |X|	$\left\lvert \frac{2(1 - Y - X)}{(1 - Y)(1 + \cos^2 \theta)} \right\rvert$ in display . Use SHIFT key for | X |
32. $\sqrt{X}$	
33. (St) Selector Switch Down	$\sqrt{\left\lvert \frac{2(1 - Y - X)}{(1 - Y)(1 + \cos^2 \theta)} \right\rvert}$ in register 04
34. 04	

Keystroke	Comment
35. 1	
36. STORE LEFT	1 is in left
37. STORE RIGHT	1 is in right
38. (Re) Selector Switch Down	
39. [00]	Recalls Y to display
40. – LEFT	$1 - Y$ in left register
41. [03]	θ
42. Cos	
43. X^2	$\text{Cos}^2\theta$
44. 1/X	$\text{Sec}^2\theta = \frac{1}{\text{Cos}^2\theta}$
45. + RIGHT	$(1 + \text{Sec}^2\theta)$ in right
46. X LEFT	$(1 - Y)(1 + \text{Sec}^2\theta)$ in left register
47. 4	
48. X LEFT	$4(1 - Y)(1 + \text{Sec}^2\theta)$ in left register
49. [02]	Recalls ϵ_1 to display
50. STORE RIGHT	ϵ_1 in right register
51. [03]	Recalls θ to display
52. Tan	
53. X^2	$\text{Tan}^2\theta$
54. X RIGHT	$\epsilon_1 \text{Tan}^2\theta$ in right register
55. RECALL LEFT	Recalls $4(1 - Y)(1 + \text{Sec}^2\theta)$ to display
56. ÷ RIGHT	
57. (St) Selector Switch Down	
58. [05]	$\frac{\epsilon_1 \text{Tan}^2\theta}{4(1 - Y)(1 + \text{Sec}^2\theta)}$ in Register [05]
59. (Re) Selector Switch Down	

Keystroke	Comment
60. 03	Recalls θ to display
61. Cos	
62. X^2	
63. 1/X	$\text{Sec}^2\,\theta = \dfrac{1}{\text{Cos}^2\,\theta}$
64. STORE LEFT	
65. 3	
66. + LEFT	$3 + \text{Sec}^2\,\theta$ in left register
67. 00	Recalls Y to display
68. ÷ LEFT	$\dfrac{3 + \text{Sec}^2\theta}{Y}$ in left and display
69. 03	Recalls θ to display
70. Tan	
71. X^2	
72. + LEFT	$\dfrac{3 + \text{Sec}^2\,\theta}{Y} + \text{Tan}^2\theta$ in left and display
73. (X) Selector Switch Down	
74. 05	$\dfrac{\epsilon_1\ \text{Tan}^2\theta}{4\,(1 - Y)\,(1 + \text{Sec}^2\,\theta)}$ $\left(\dfrac{3 + \text{Sec}^2\theta}{Y} + \text{Tan}^2\,\theta\right)$ in 05
75. 1	
76. STORE LEFT	
77. (Re) Selector Switch Down	
78. 05	
79. − LEFT	$1 - \dfrac{\epsilon_1\ \text{Tan}^2\theta}{4(1 - Y)\,(1 + \text{Sec}^2\,\theta)}$ $\left(\dfrac{3 + \text{Sec}^2\,\theta}{Y} + \text{Tan}^2\right)$ in display and left register

Keystroke	Comment

80. (X) Selector Switch Down

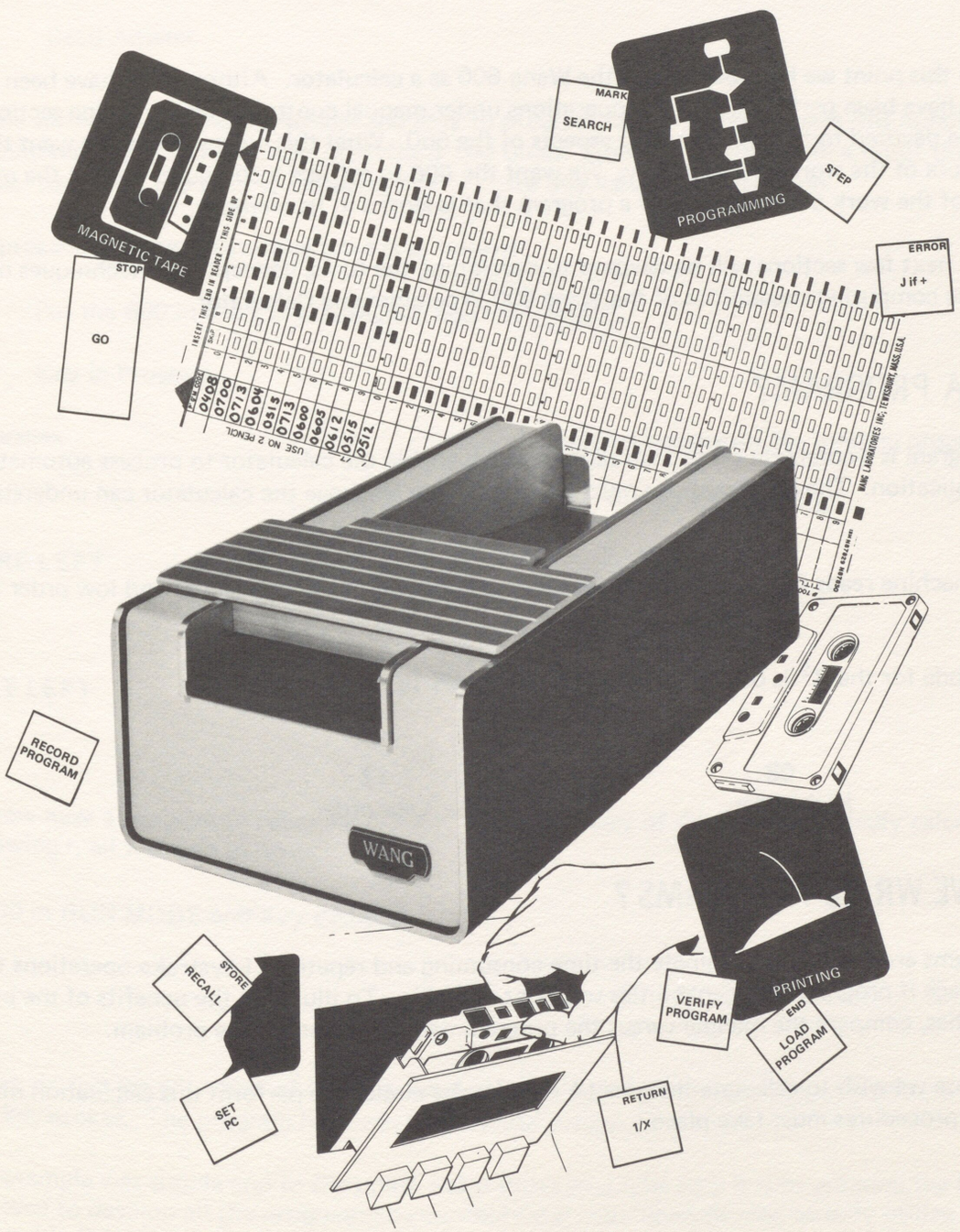
81. 04

$$\mu = \sqrt{\left| \frac{2\,(1 - Y - X)}{(1 - Y)\,(1 + \cos^2\theta)} \right|} \left[1 - \frac{\epsilon_1 \tan^2\theta}{4(1-Y)(1+\sec^2\theta)} \left(\frac{3 + \sec^2\theta}{Y} + \tan^2\theta \right) \right]$$

answer μ = .1970874057

PART II

PROGRAMMING YOUR WANG 600



INTRODUCTION TO PROGRAMMING

Up to this point we have been using the Wang 600 as a calculator. Although we have been writing programs, we have been performing all the operations under manual control. The remaining sections of this manual will be devoted to the programming aspects of the 600. What this means is that we want the 600 to start doing more of the work automatically. We want the 600 to act like a computer so that the machine will do most of the work under control of a program that is stored in its memory.

These next few sections will be devoted to developing the tools, concepts and techniques needed to best utilize the computing capabilities of the Wang 600 Programmable Calculator.

WHAT IS A PROGRAM?

A program is a complete set of instructions which enable the calculator to process automatically a particular application. These instructions must be written in a language the calculator can understand and interpret.

The machine reads four digit code numbers in a high order (first two digits) and low order (last two digits) form.

The code for the X^2 is 08 12, the code for $\sqrt{X}$ is 08 13, and so on.

08	12
High order	Low order

WHY DO WE WRITE PROGRAMS?

Programs are written to eliminate the time-consuming and repetitive keystroke operations that would take place if programming capabilities were not available. To illustrate the benefits of the programming capabilities, compare the manual versus the programmable calculating of a problem.

Suppose we wish to calculate the area ($A = \pi r^2$) of a circle. To perform this calculation manually, the following procedures must take place.

For this example, let r = 8.3
let π = 3.14

Keystroke Operation

1. Key **8.3**
2. Key **x^2**
3. Key **STORE LEFT**
4. Key **3.14**
5. Key **X LEFT**

Read Answer

On the surface this looks too simple and easy to program, but suppose we have one-hundred values for r. We would have to key those same operations 100 times. How can we eliminate a lot of this work? The answer is: to write a program to calculate $A = \pi r^2$ for any value that r takes on.

The procedure to perform these tasks is as follows:

Put the 600 in the LEARN MODE and key **PRIME**.

Key in Program.

Keystroke Operation

1. Key **X^2**
2. Key **STORE LEFT**
3. Key **3.14**
4. Key **MULT LEFT**
5. Key **Stop**

Operating Instructions for running program

1. Key a value for r
2. Key GO

We now have a program to calculate $A = \pi r^2$ in the memory of the 600. To actually calculate $A = \pi r^2$ knowing r, all we need to do is:

1. Put the 600 in RUN MODE and Key **PRIME**.
2. Key in r
3. Key GO

We can do this for any value of r. Thus, what have we done? We have eliminated the manual indexing of repetitive keystrokes. The program has been learned into memory.

This example was simple and to the point, yet it illustrated how easy it is to program the 600. We will now proceed to develop all the programming concepts and techniques needed to fully utilize the Wang 600 Programmable Calculator.

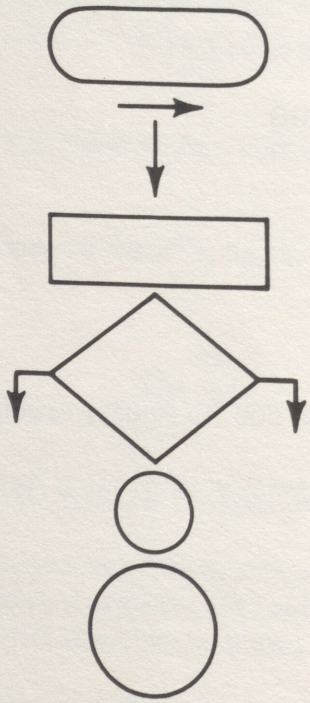
Chapter 6

FLOW DIAGRAMS

To write a program it is necessary to plan the sequence of operations that the computer is to perform. The relationships between the operations may become complex and difficult to keep clearly in mind. For this reason programmers usually sketch out an outline of the program. The outline, which is called a flow diagram or block diagram, gives the programmer a visual representation of the relationships involved.

A block diagram is made up of a set of blocks, the shapes of which indicate the nature of the computer operations described, along with lines and arrows that show the flow of control between the various operations.

Later sections of this text will invoke the following symbols:



An oval box indicates starting or stopping operation.

Arrows indicate the direction of flow through the diagram. Every connecting line should have an arrow on it.

A rectangular box indicates any processing operation except a decision.

A diamond indicates a decision. The lines leaving the decision box indicate the possible path the program can follow.

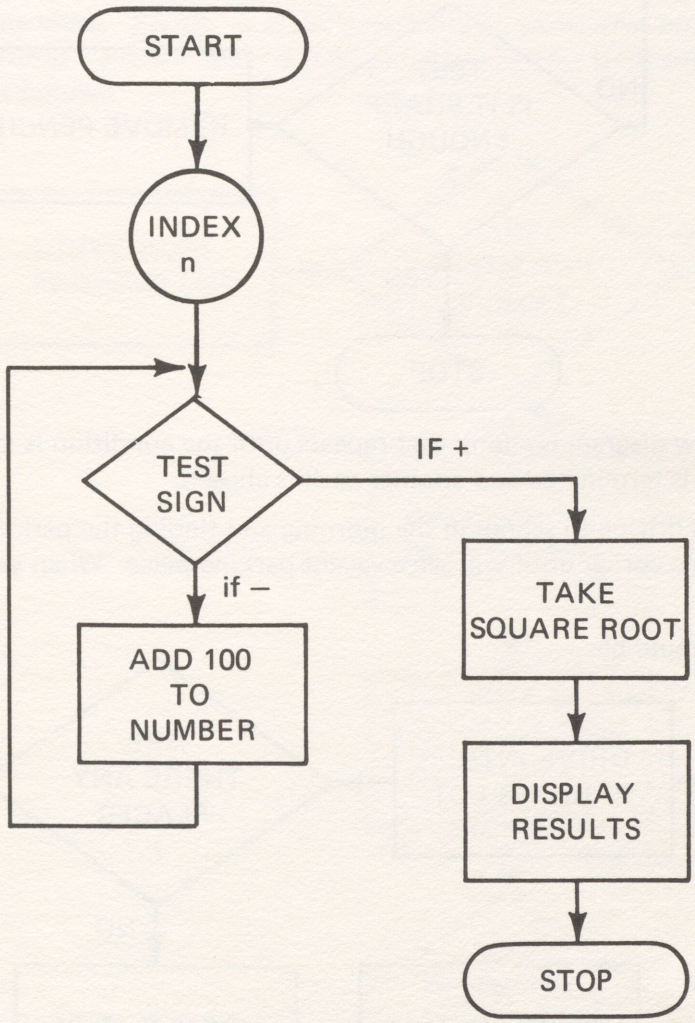
A small circle indicates a connection between two points in a block diagram, where a connecting line would be too clumsy.

A large circle indicates a connection in a program where a number may be indexed.

To illustrate how a block diagram may be used, we shall work out a problem involving a test for the sign of a number before taking the square root of that number. If the number is positive

we will perform the square root operation, but if it is negative we will add 100 to the number and test it again. If the new result is positive, the square root operation will be performed.

The flow diagram would be:

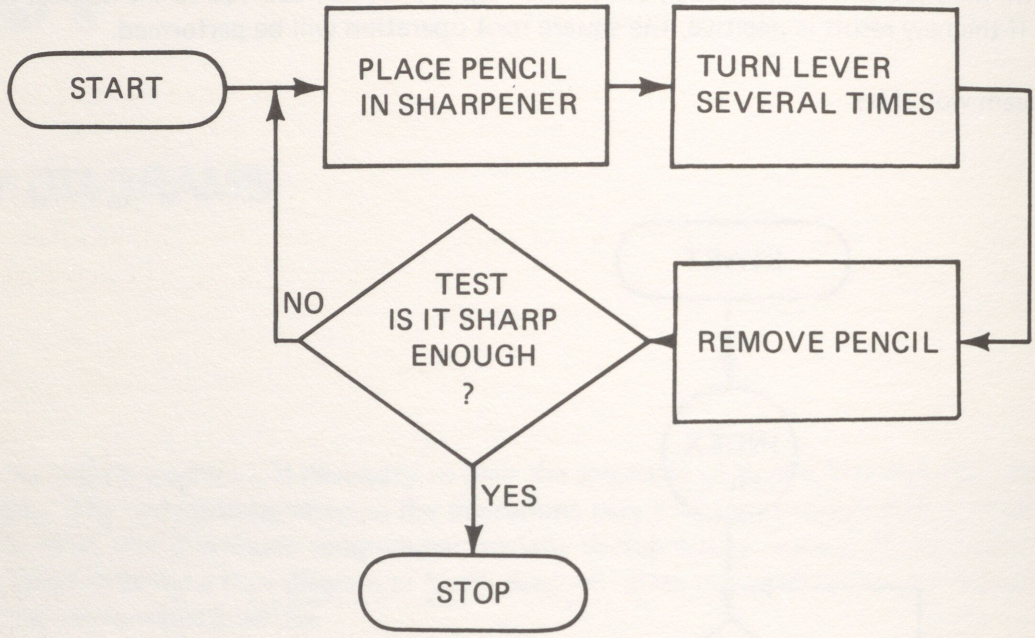


You will find it helpful to sketch out a flow diagram before you attempt to write the programs in the later sections.

In addition to using flow diagrams to represent mathematical processes, we may use them to represent everyday occurrences. The pencil needs to be sharpened and we receive permission to walk to the sharpener. Our first action, once we get to the pencil sharpener, is to put the end of the pencil into the sharpener. We turn the handle several times and remove the pencil. Then, we examine the tip of the pencil. We are required to make a decision based on the condition of the pencil point. Is it sharp enough or not?

If the pencil is not sharp enough, we put it back into the sharpener and continue to turn the handle, and again we test the condition. We keep repeating the same steps until the test indicates the pencil is sharp enough for us to stop.

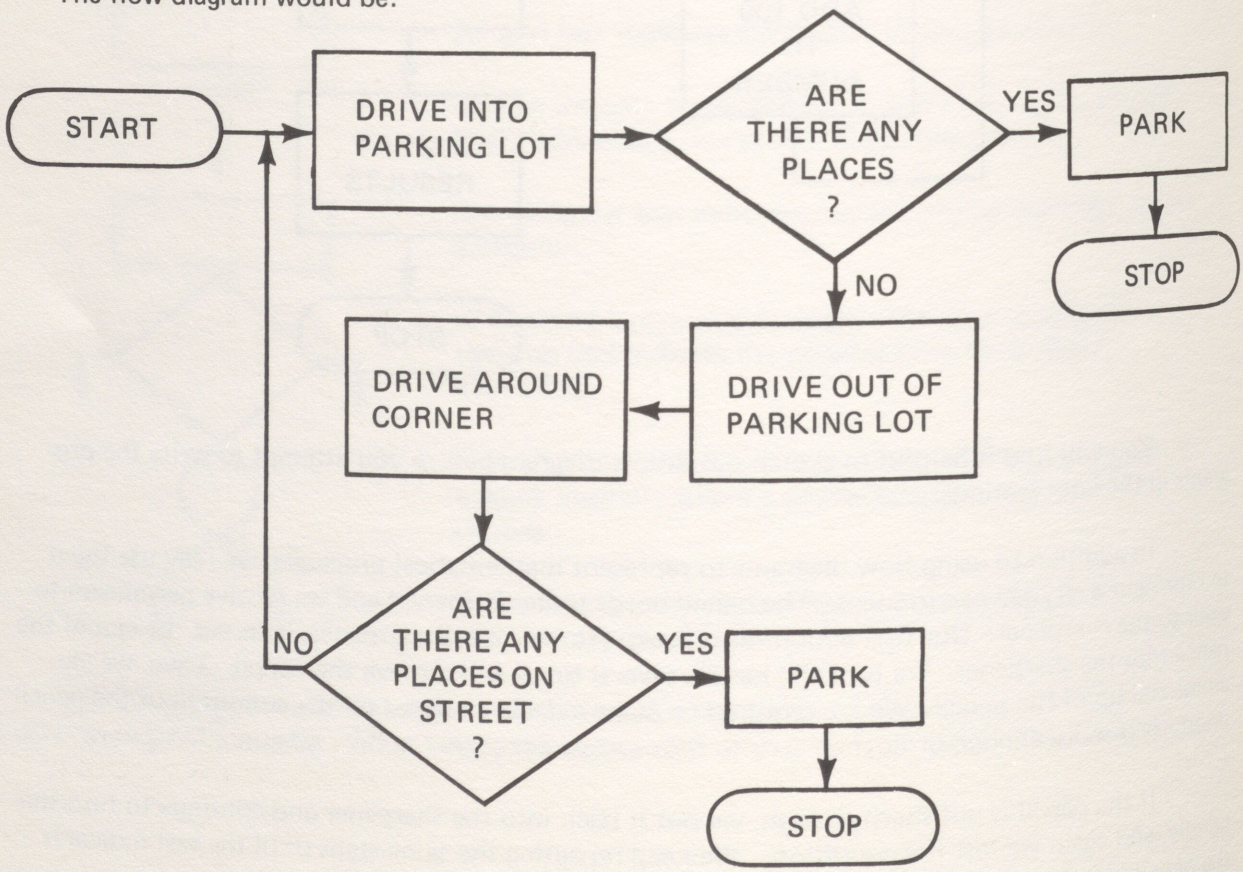
The flow diagram would be:



The path of the flow diagram is a loop that repeats until the condition is met (pencil sharp enough) and then the loop is terminated and another path is chosen.

Another example is driving to school in the morning and finding the parking lot full. You then keep driving around the corner until you see a vacant parking place. When you do, you pull over and park.

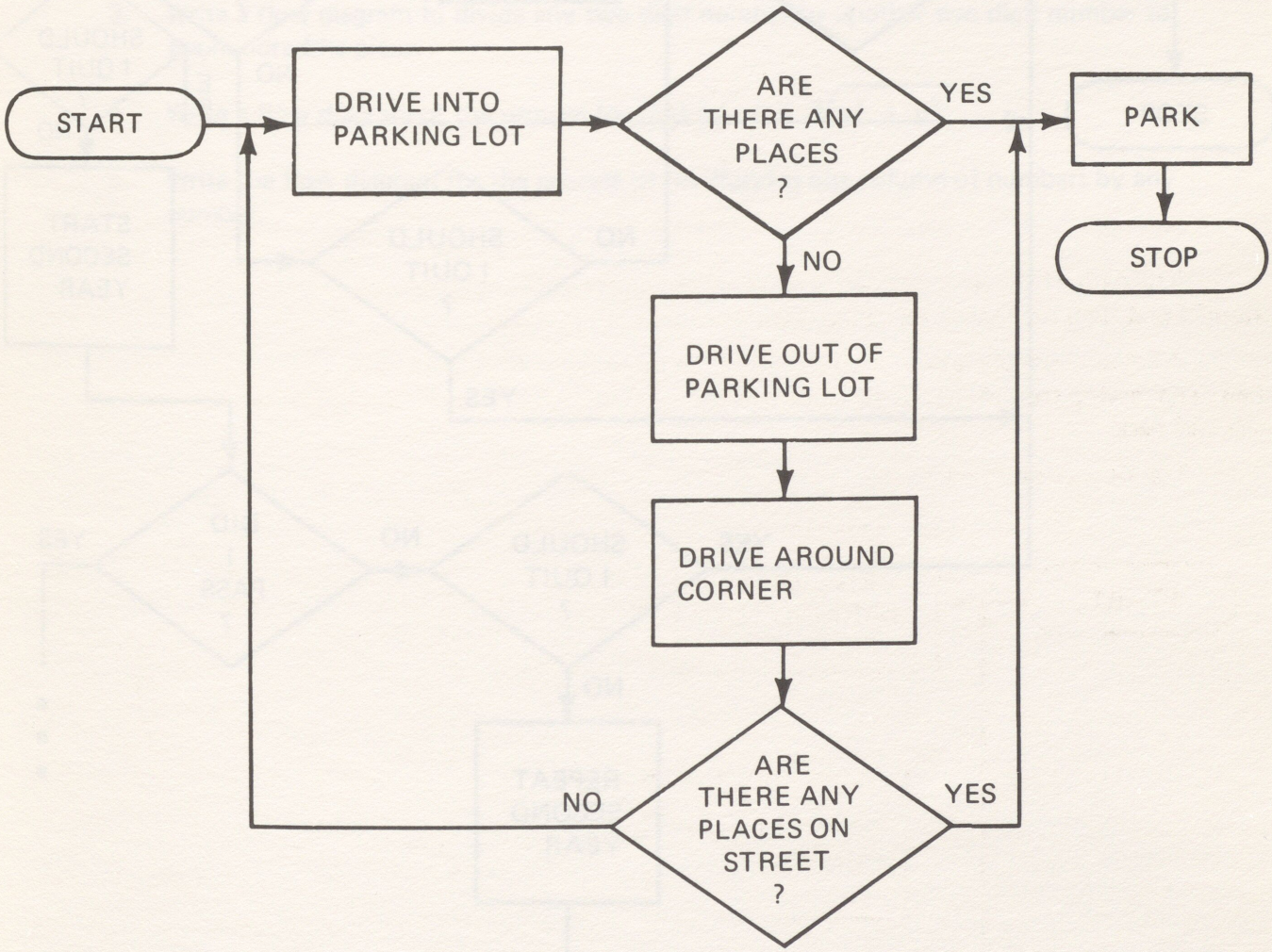
The flow diagram would be:



Notice the upper diamond-shaped decision block, "ARE THERE ANY PLACES?" Follow the "YES" path to the block "PARK" and the oval "STOP"

Now notice the lower diamond-shaped decision block "ARE THERE ANY PLACES ON STREET?" and the "YES" path that also leads to a block "PARK" and an oval "STOP"

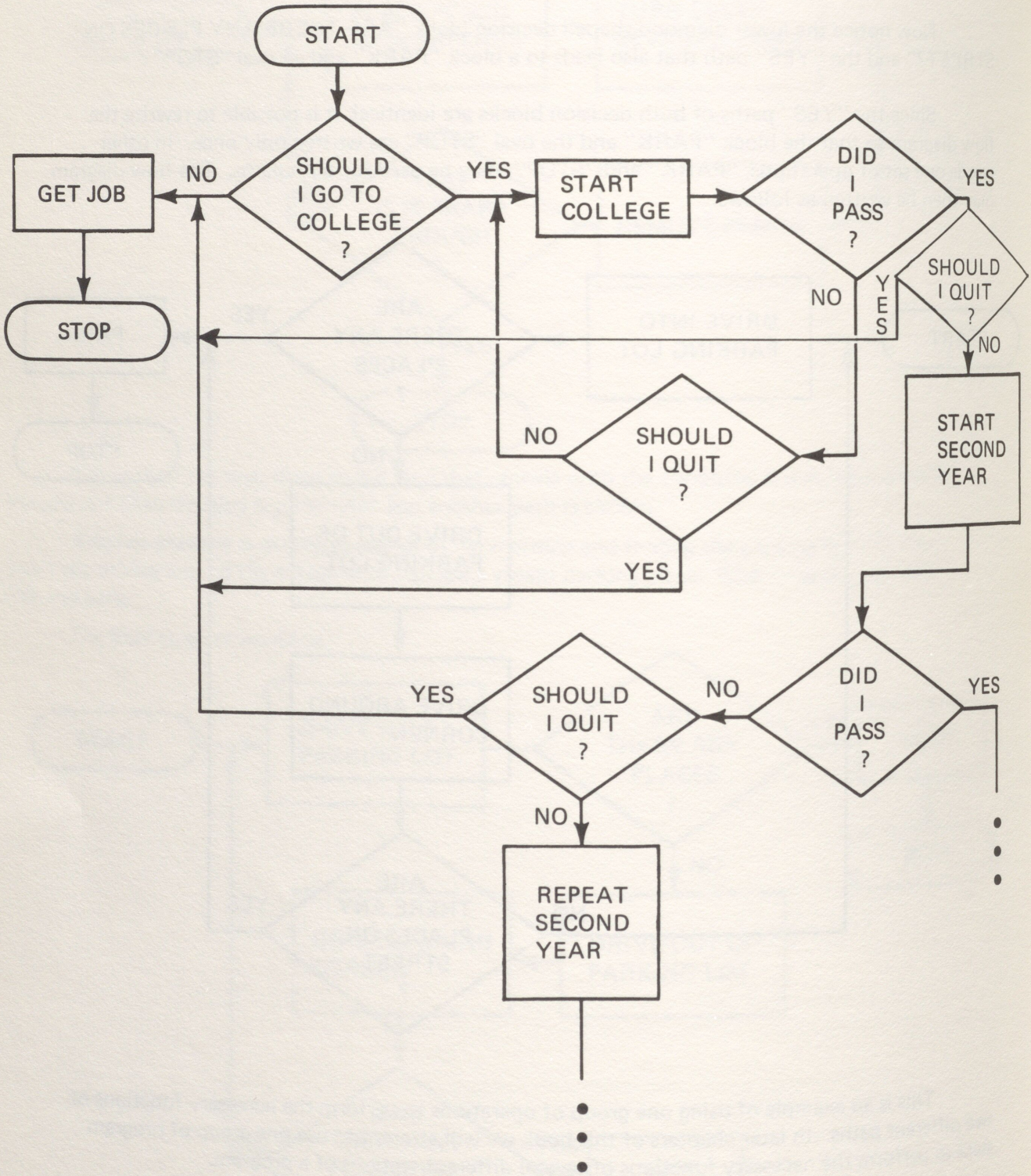
Since the "YES" paths of both decision blocks are identical, it is possible to rewrite the flow diagram so that the block "PARK" and the oval "STOP" are written only once. In other words one set of operations, "PARK" and "STOP", may be used for both paths. The flow diagram may then be written as follows:



This is an example of using one group of operations to perform the necessary functions of two different paths. In later chapters of this book we will attempt to use one group of program steps to perform the necessary functions of several different sections of a program.

Here is an important flow diagram for you to consider. It is a flow diagram that shows several possible directions you may take when you graduate from high school. You may decide to go to college. After your Freshman year at college you have to make another decision: If I have passed, should I quit and get a job or continue to my Sophomore year? Or, if I have failed, should I quit or should I repeat my Freshman year?

The flow diagram would be:



Also, as an exercise you may wish to complete the flow diagram to encompass the third year and the fourth year and the possibility of joining the Armed Forces. You might also want to elaborate on the path "GET JOB" at all levels of the program.

EXERCISES

1. You are informed by your teacher that you are to read a chapter in your Physics book and prepare to take a test the next day. The test will consist of modification of the Practice Problem at the end of the chapter. Write the flow diagram of the process you would use to prepare for the test.

2. Write a flow diagram for multiplying any two digit number by another two digit number. (Be sure to make a provision for carry.)

3. Write a flow diagram to divide any two digit number by another two digit number to the hundredths place.

4. Write a flow diagram of the process for finding $r = \sqrt{x^2 + y^2}$.

5. Write the flow diagram for the process of multiplying one column of numbers by any number.

Chapter 7

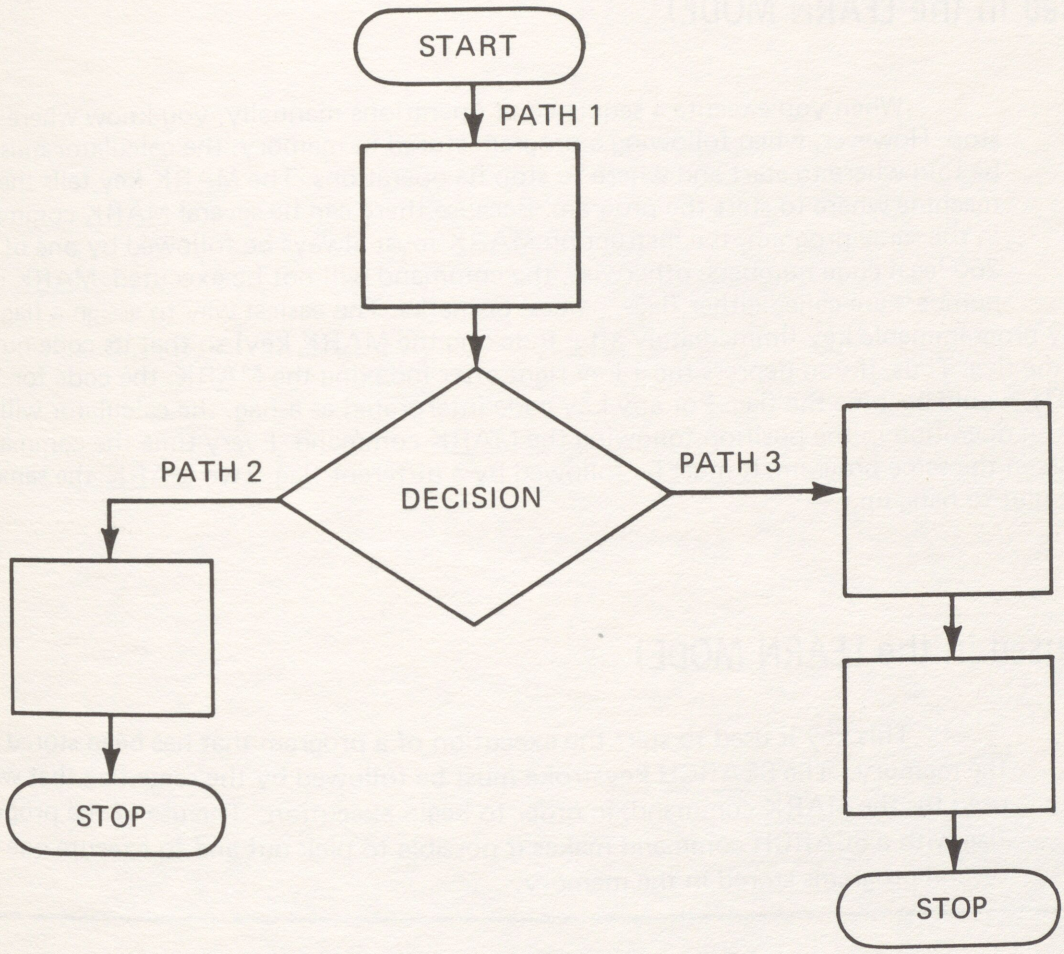
RELATING FLOW DIAGRAMS TO COMPUTER PROGRAMMING

Chapter 6 developed the idea of pictorially representing, with a flow diagram, the interrelationships that are involved in an overall process. A flow diagram was written to represent the process of sharpening a pencil. Another flow diagram was written to represent the process of deciding whether or not a number under the radical is positive or negative. If the number is positive, the square root of the number can be extracted. If the number is negative, the flow diagram indicates that 100 is to be added to the number.

There is a twofold purpose for developing the idea of flow diagrams. The first purpose is to structure our thinking in terms of the logical sequence of steps in a process. Hopefully, we may now better organize the sequence of steps to follow in obtaining our everyday objectives. The second purpose for developing flow diagrams is to lay the foundation for writing computer programs.

Recall that a flow diagram is a pictorial representation of a process. This pictorial representation must now be related to instructions that are meaningful to the computer. Chapter 7 will relate the flow diagram to computer programming.

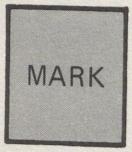
Consider the following flow diagram.



Notice that PATH 1 leads to the diamond shaped decision block. At this point in the diagram a decision is necessary. The outcome of the decision will determine if PATH 2 is to be taken or if PATH 3 is to be taken. Notice that we have given each path a name or label, PATH 1, PATH 2, PATH 3, to distinguish one path from another. Likewise in a computer program we must distinguish one path of program instructions from another path of program instructions. Instead of using PATH 1, PATH 2, and PATH 3, we label the paths of the program MARK 1, MARK 2, and MARK 3.

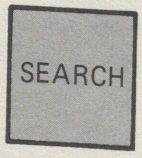
A program needs some sort of command to tell the calculator where to start and where to end its calculation. This is the purpose of the SEARCH and MARK commands.

MARK (used in the LEARN MODE)



When you execute a sequence of operations manually, you know where to stop. However, when following a program stored in memory, the calculator must be told where to start and where to stop its operations. The MARK key tells the machine where to start the program. Because there can be several MARK commands in the same program, the instruction MARK must always be followed by one of the 260 legal code numbers; otherwise, the command will not be executed. MARK code numbers are called either flags, names, or marks. The easiest way to assign a flag is to depress any programmable key (immediately after indexing the MARK key) so that its code number becomes the flag. Thus, if you depress the 1 key right after indexing the MARK, the code for 1, which is (0001), would become the flag. For any key code interpreted as a flag, the calculator will not interpret it as an operation in the position following the MARK command. Every time the command MARK appears in the same program, it must be followed by a different flag. Two MARK the same will cause the unit to hang-up.

SEARCH (used in the LEARN MODE)



This key is used to start the execution of a program that has been stored in the memory. The SEARCH keystroke must be followed by the same flag that was used for the MARK command, in order to begin execution. The use of the proper flag with a SEARCH command makes it possible to pick out and to execute one of several programs stored in the memory.

NOTE:

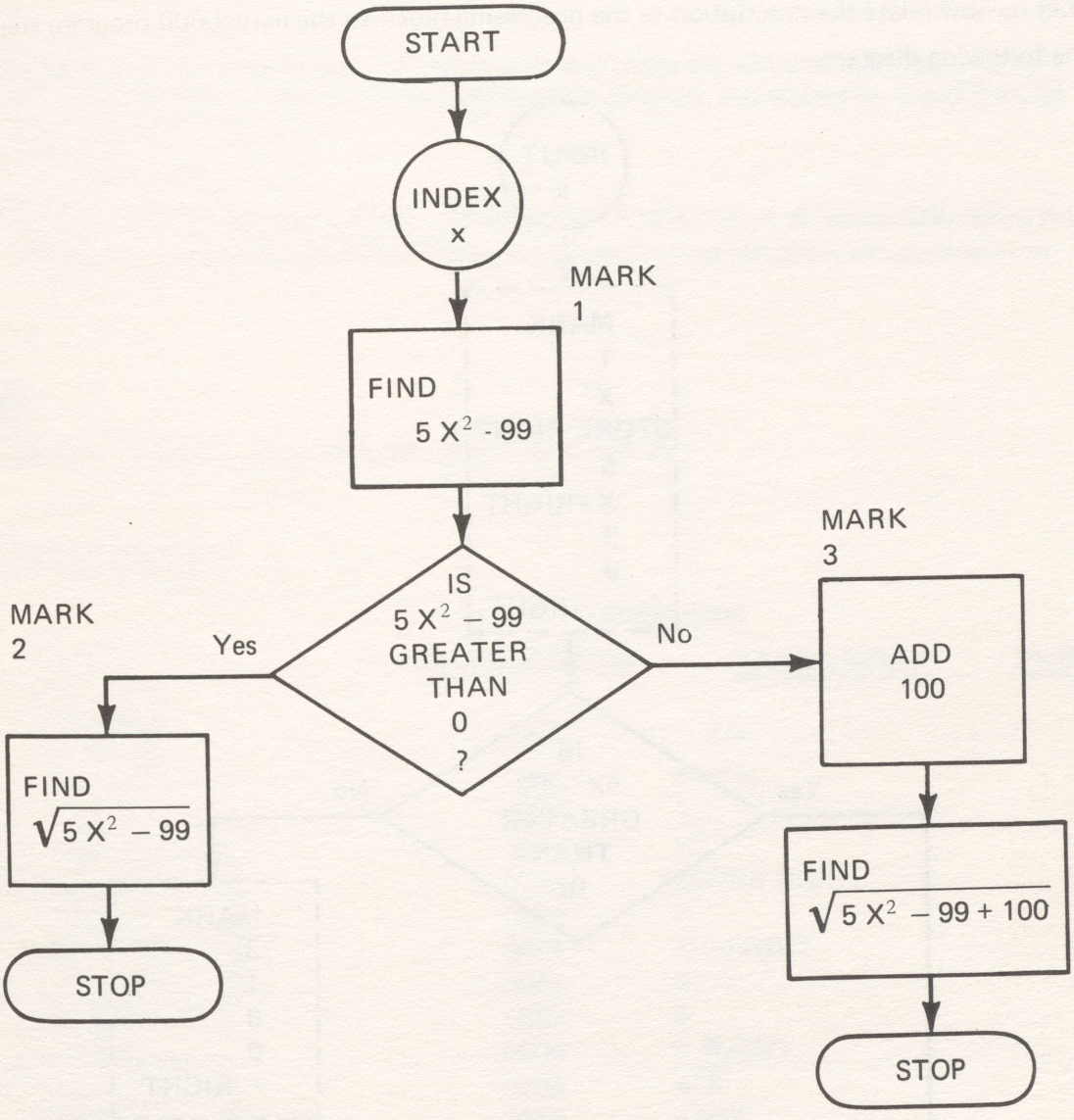
Although almost any key on the 600 can be used as a MARK or SEARCH code (ex. MARK 3, SEARCH 3, or MARK SIN, SEARCH SIN), the END PROG key should not be used as a MARK in a program if you intend to record the program on a tape cassette.

If MARK END PROG is used, the tape will record only the steps preceding the END PROG code (09 14).

Chapter 6 defined the rectangular block in the flow diagram to represent any processing operation except a decision. What do we mean by a processing operation?

Let us answer this question by giving several examples.

Notice the flow diagram on the next page. Write a program to evaluate $\sqrt{5X^2 - 99}$, taking into consideration that $5X^2 - 99$ may be less than 0. If $5X^2 - 99$ is less than zero, add 100 to that value before taking the square root.



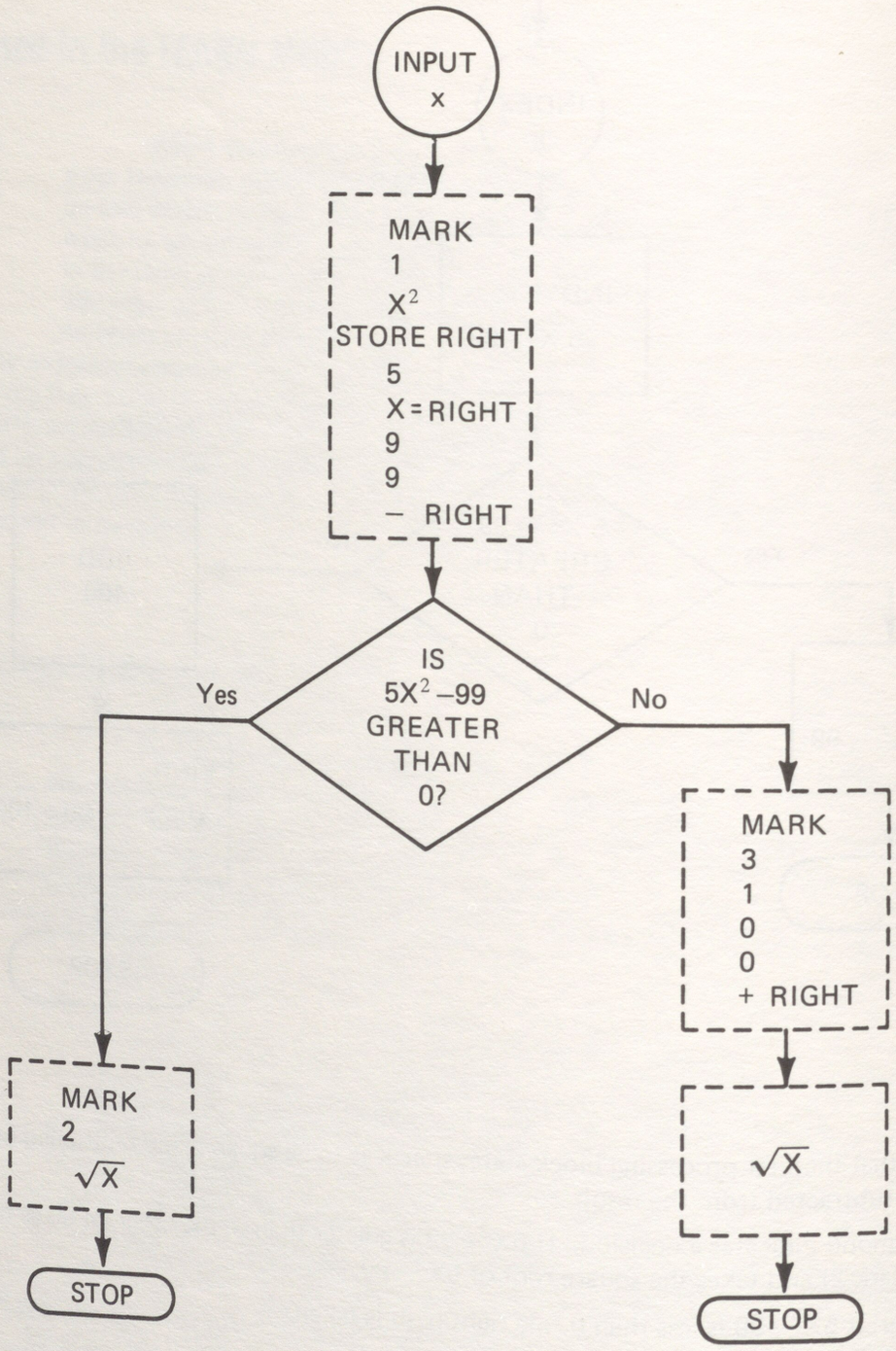
Notice that the first processing block states that X is to be squared and multiplied by 5, and 99 is to be subtracted from the result.

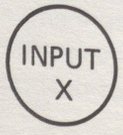
The diamond indicates a decision. If $5X^2-99$ is greater than 0, the program takes the left path (named Mark 2) and takes the square root of $5X^2-99$.

However, if $5X^2-99$ is less than 0, the right path is taken.

It should be clear that a rectangular processing block represents an operation or a series of operations. The description inside the block briefly explains the operations to be performed. The extent of detail in a block usually depends on the complexity of the program.

Let us now relate the description in the processing block to the actual 600 program steps. Study the following diagram.



Note that the symbol 

has been added to the diagram. This symbol indicates that the operator will key a value of x into the display.

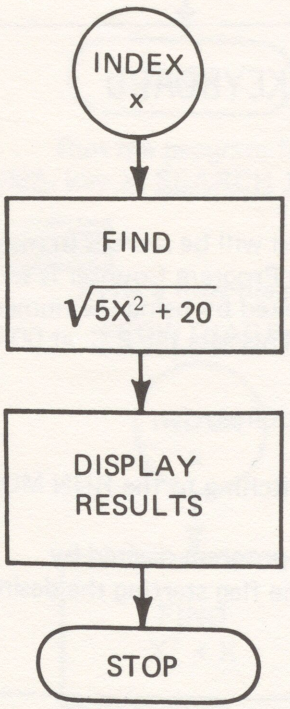
The program steps of the first processing block will cause the value in the display to be squared. This value will then be stored in the right Scratch Register, multiplied by 5 and then 99 will be subtracted from the result.

Later in this chapter we will transform the decision making block to actual 600 Series Program steps. Meanwhile, let us get some practice writing flow diagrams and relatively simple programs.

Example 1

Write a flow diagram and a program for $y = \sqrt{5X^2 + 20}$

FLOW DIAGRAM



PROGRAM

INSTRUCTIONS LEARN MODE PRIME

Step	Key
0000	MARK
0001	1
0002	X^2
0003	STORE RIGHT
0004	5
0005	X= RIGHT
0006	2
0007	0
0008	+ RIGHT
0009	$\sqrt{X}$
0010	STOP
0011	END PROG

RUN the program for X = 10, X = 20, and X = 30. Remember to reset the 600 to RUN MODE, then PRIME, key 10 SEARCH 1, and read the result.

Note:

Notice the END PROG command at step 0011.

LEARN MODE

The LEARN MODE is used when programming instructions into memory. When the LEARN MODE switch is engaged, the display shows two sets of numbers: the number on the left will display 4 digits indicating the step number; i.e., 0012, Step 12: the number on the right is a four-digit number indicating the instruction code or the function being processed; i.e., 05 15 (the operation being performed is division in Register [15]). The codes are listed on the pull-out cards on the bottom of the calculator.

0012 05 15

In the LEARN MODE, a user can step through his program for such things as correcting an error, debugging his program or adding additional steps. This feature is very useful when preparing a program for calculating work.

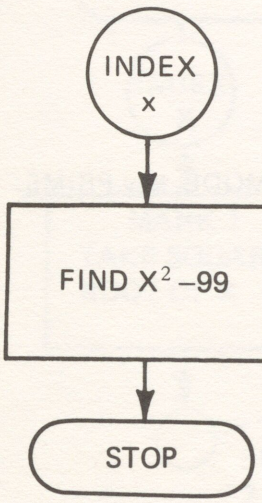
HOW TO WRITE A PROGRAM INTO MEMORY FROM THE KEYBOARD

1. Depress the LEARN MODE button.
2. Set the Program Counter at the step where the first program command will be stored. In most cases it will be at Step 0000 if no other programs are in memory. The Program Counter is set by depressing the SET P. C. key followed by a four-digit number indexed by using the numeric 0-9 keys. If you want to start at Step 0000, depress the PRIME key. This sets the P.C. at 0000.
3. Introduce the program via the sequence of keystrokes the program requires.
4. The program is now stored in the memory and can be executed by switching to the RUN MODE.
5. If several programs are stored in memory, you can execute whichever program desired by indexing the SEARCH command and the flag which corresponds to the flag starting the desired program.

Example 2

Write a flow diagram and a program for $y = X^2 - 99$.

FLOW DIAGRAM



PROGRAM

INSTRUCTIONS: In LEARN MODE, key PRIME.

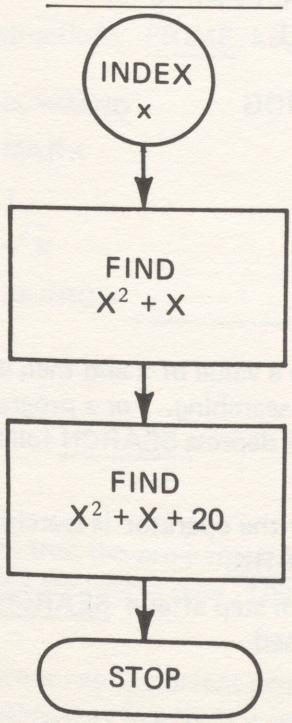
Step	Key
0000	MARK
0001	1
0002	X^2
0003	STORE RIGHT
0004	9
0005	9
0006	– RIGHT
0007	STOP
0008	END PROG

Run the program for X = 1, 2, and 3. Remember to reset the 600 to RUN MODE, then PRIME, key X, SEARCH 1, and read the answer.

Example 3

Write a flow diagram and a program for $y = X^2 + X + 20$.

FLOW DIAGRAM



PROGRAM

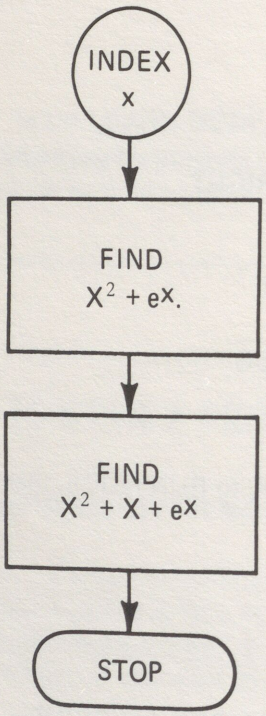
INSTRUCTIONS: In LEARN MODE, key PRIME.

Step	Key
0000	MARK
0001	3
0002	STORE RIGHT
0003	X^2
0004	+ RIGHT
0005	2
0006	0
0007	+ RIGHT
0008	STOP
0009	END PROG

Run the program for x = 1, 2 and 3. Key X, SEARCH 3.

Example 4

Write a flow diagram and the program for $y = X^2 + X + e^x$.



PROGRAM

INSTRUCTIONS: In LEARN MODE, key PRIME.
Key **X, SEARCH 7**

Step	Key
0000	MARK
0001	7
0002	STORE LEFT
0003	X^2
0004	STORE RIGHT
0005	RECALL LEFT
0006	e^x
0007	+ RIGHT
0008	RECALL LEFT
0009	+ RIGHT
0010	STOP
0011	END PROG

To run any of the programs just written, the operator keyed in a value of x and then depressed SEARCH followed by the key that indicated the MARK that we were searching. For a program labeled MARK 1 we depress SEARCH 1; or for a program labeled MARK 7 we depress SEARCH followed by a 7.

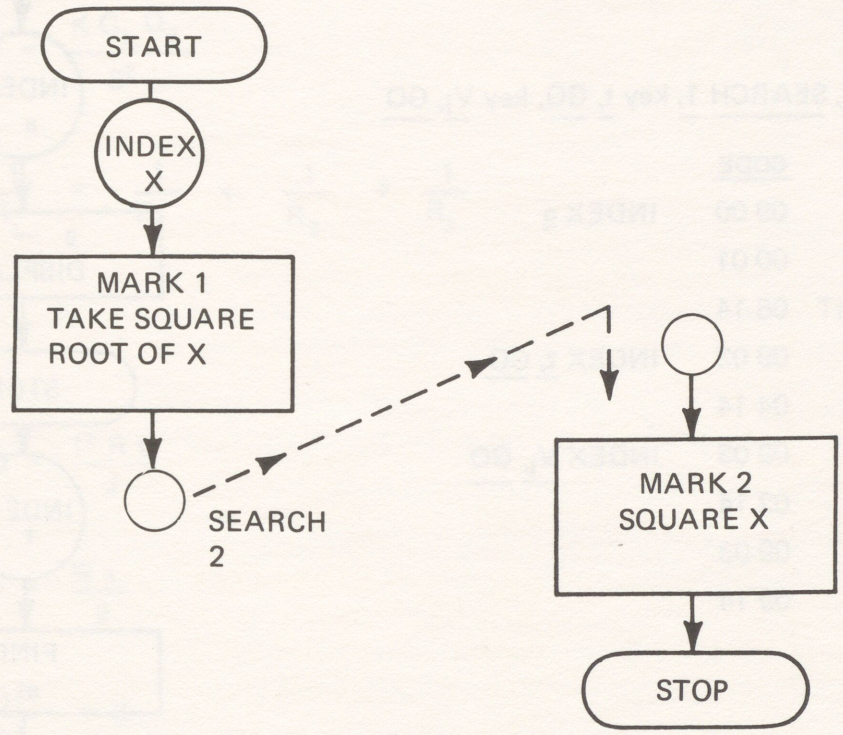
The procedure is usually always the same. This is true whether the operator is searching a particular MARK or whether the program is searching a particular MARK.

Yes, it is true. A program can SEARCH a MARK. The program step after a SEARCH command must contain the code for the particular MARK being searched.

Example 5

Write a program for $y = \sqrt{x}$; then have the program SEARCH a second program that performs the x^2 operation and stops.

The flow diagram with the appropriate MARK and SEARCH commands would be:



The actual calculator program would be:

Operating instructions: PRIME, key x, SEARCH 1.

STEP	COMMAND	STEP	COMMAND
0000	MARK		
0001	1		
0002	$\sqrt{x}$		
0003	SEARCH	0043	MARK
0004	2	0044	2
0005		0045	x^2
0006		0046	STOP
		0047	END PROG

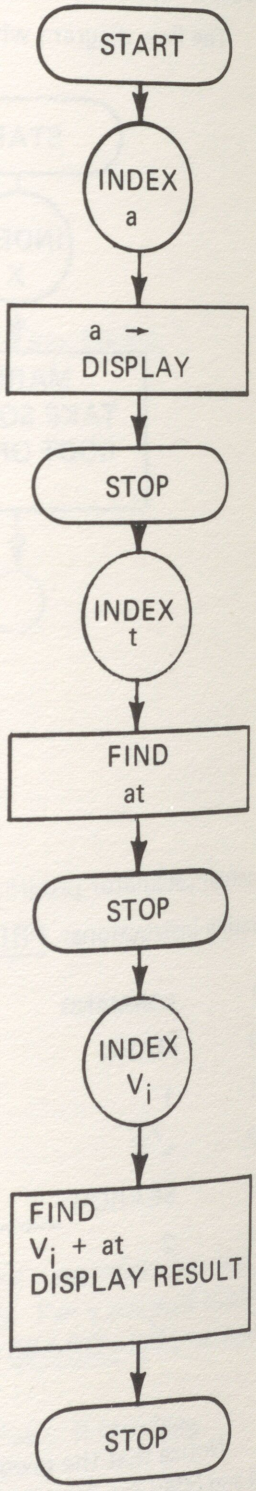
Notice that the program suddenly branches from Step 0004 to Step 0043. The steps between 0004 and 0043 are passed over. The calculator starts executing the commands at Step 0043.

An extremely important programming concept has just been introduced. Later it will become more evident as to why it is important for a programmable calculating, or computing device, to be able to branch from one group of program steps to another group of program steps.

EXERCISES

1. Write a flow diagram and a program for each of the following formulas:

a. $V_f = V_i + at$



Operating Instructions:

PRIME, key a, SEARCH 1, key t, GO, key V_i, GO

STEP	COMMAND	CODE	
0000	MARK	09 00	INDEX a
0001	1	00 01	
0002	STORE RIGHT	06 14	
0003	STOP	09 03	INDEX t, GO
0004	X RIGHT	04 14	
0005	STOP	09 03	INDEX V_i, GO
0006	+ RIGHT	02 14	
0007	STOP	09 03	
0008	END PROG	09 14	

b. $S = V_i t + \frac{1}{2} a t^2$

c. $V_f = \sqrt{V_i^2 + 2as}$

d. $F = \frac{G M_1 M_2}{S^2}$

e. $F_i = \frac{M V^2}{r}$

f. $T = 2\pi \sqrt{\frac{L}{g}}$ where $g = 9.8$

g. $K.E. = \frac{1}{2} M V^2$

h. $E = Mgh + \frac{1}{2} M V^2$

i. $S_i = \frac{f S_c}{S_c - f}$

j. $F = \frac{K Q_1 Q_2}{S^2}$

k. $\frac{1}{R_t} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$

l. $Q = \frac{I^2 R t}{J}$

m. $F = \frac{W a}{g}$

n. $\frac{R^3}{T^3} = K$

o. $a = \frac{4 \pi^2 R}{T^2}$

p. $F = \frac{k Q_1 Q_2}{r^2}$

2. What is meant by branching?

3. Write a short program that indicates how a program branches from one point in a program to another point.

Chapter 8

PROGRAMMING CONCEPTS LOOPS, DECISIONS, SUBROUTINES

Chapter 7 developed the idea of writing simple 600 programs. These programs contained program steps that were executed in a sequential manner, i.e., Step 0000, 0001, 0002 to the last step of the program. Only at the end of the chapter did we indicate that a programmable SEARCH command enabled the programmer to write programs to branch from one point to another.

In this chapter we will realize why the ability to branch is such an important programming feature. This chapter will also teach us how to loop, make decisions, and address subroutines.

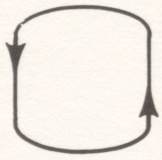
LOOPS

A loop is generally understood to be a section of a program which may be performed repeatedly. Two simple examples follow.

Example 1

This program never stops; it circles or loops continuously between steps 0000 and 0003.

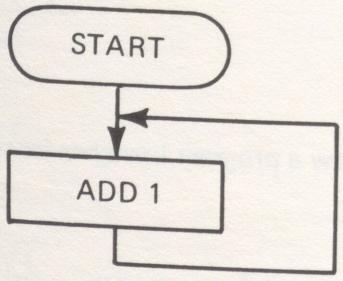
STEP	COMMAND	CODE
0000	MARK	0900
0001	1	0001
0002	SEARCH	0800
0003	1	0001



Example 2

Write a program that keeps adding by 1.

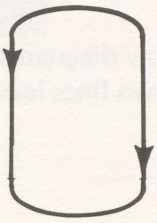
The flow diagram would be:



The program would be:

Operating instructions: PRIME, SEARCH 2

STEP	COMMAND	CODE
0000	MARK	0900
0001	2	0002
0002	1	0001
0003	+ RIGHT	0214
0004	SEARCH	0800
0005	2	0002



Neither program ever stops unless interrupted by a manual operation like PRIME or STEP, or a power failure.

It is interesting to note that the above program counts. Appropriately, a program loop that either decrements or increments, is called a counter. We will soon see how a counter is used as a basic building block for making decisions.

USING THE HALF SECOND PAUSE

The 600 enables you to visually monitor your program by displaying the results in the window as the program flow progresses. Results can be displayed in half second increments for any length of time desired. Each half second pause requires that the two-step command ALPHA STOP be inserted after the step in the program you wish to display.

For example, in the above program, if you wanted to display the contents of the right keyboard register for one second each time it is incremented, when keying in the program you would include two half-second pauses:

Operating instructions: PRIME, SEARCH 2

STEP	COMMAND	CODE	
0000	MARK	0900	
0001	2	0002	
0002	1	0001	
0003	+ RIGHT	0204	
0004	ALPHA	0902	} 1/2 Sec. Pause
0005	STOP	0903	
0006	ALPHA	0902	} 1/2 Sec. Pause
0007	STOP	0903	
0008	SEARCH	0800	
0009	2	0002	

When the program is executed, the DISPLAY will show the contents of the Right keyboard register for 1 second, each time it is incremented by 1.

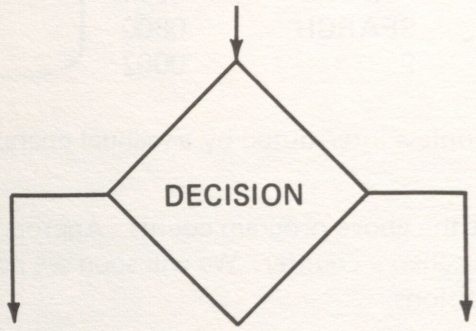
TERMINATING LOOPS

Think of a terminating loop as the action of someone who keeps trying to cross the street. Everytime he puts his foot off the curb, a car comes around the corner, and he has to step back onto the curb. He keeps looping back and forth until a certain condition exists (no cars coming) and then he terminates the loop and crosses the street.

In a program, certain conditions must exist for the computer to make the decision to stop looping and go on to perform other instructions. The condition, of course, cannot be "no card coming", but instead, it must be based on the occurrence of certain numbers because that is all the calculator has to work with. What better place to examine numbers than in the display.

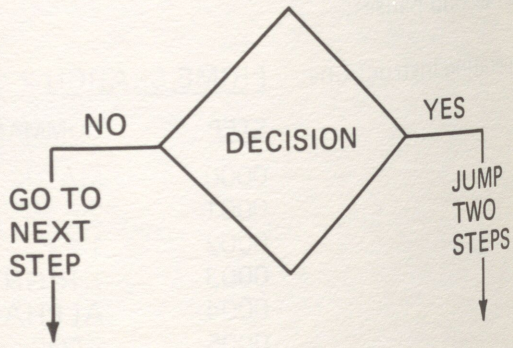
DECISIONS

Remember that the flow diagram symbol for a decision function consists of a line leading into a diamond shaped block and two lines leaving. The lines leaving represent the two possible paths that the program can follow.



There are seven decision-making commands on your Wang Model 600. They are divided into two groups. Each of the four in the first group allows the program to examine the contents of the display, and then branch accordingly.

Keys	Description	Flow Diagram Form
J if +	Jump if Positive	
J if 0	Jump if Zero	
J if ≠ 0	Jump if not equal zero	
J if ERROR	Jump if Error	
J if +		

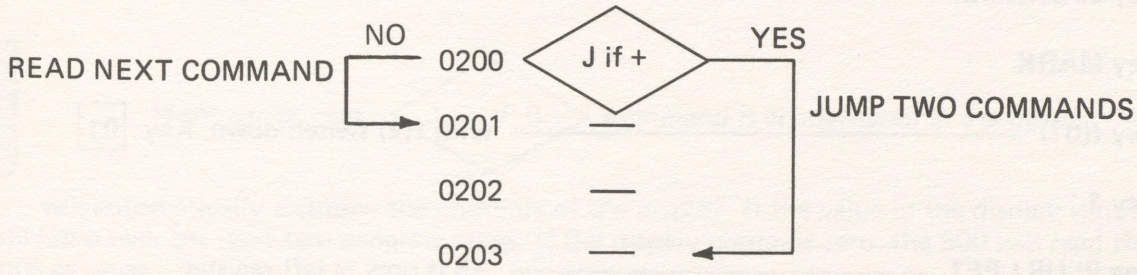


If the J if + command is encountered in a program, the contents of the display are automatically examined. If the value in the display contains a positive sign, the next two commands in the program will be bypassed or jumped over.

Consider the following diagram.

Display





At step 0200 the program examines the contents in the display and sees a +.7085. Because this value contains a "+" the program jumps from step 0200 to step 0203 without reading or executing the commands at Steps 0201 and 0202. This represents one path that the program can take.

If the value in the display were not a positive value, but a negative value, then the next two commands immediately following the J IF + command, at Steps 0201 and 0202, would have been read as usual. A SEARCH XX XX command could branch the program to a second path.

Example 3

Write a program to read negative numbers adding one to the negative number each time it goes through a loop. The calculator will continue looping until it reads its first positive number and will then stop with the positive number in the display.

Program steps eight, nine, ten and eleven are the instructions for a 1 second pause. The calculator will display, -5, -4, -3, -2, -1, and will interpret zero as a positive number, stopping the calculator and displaying the zero in the window.

OPERATING INSTRUCTIONS

Key PRIME, GO, in RUN MODE

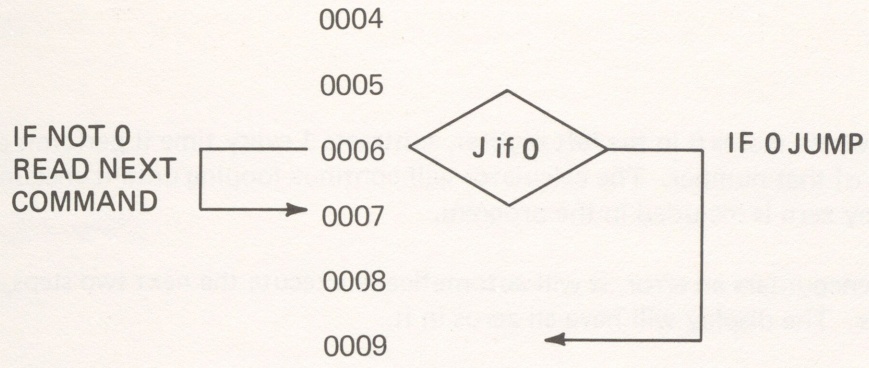
Keystroke Operations | **Comment**

LEARN MODE

1. Key **6**
2. Key **CHANGE SIGN**
3. Key **STORE LEFT** — –6 is now in left register
4. Key **MARK**
5. Key **f(01)** — With **f(x)** switch down, Key **01**
6. Key **1**
7. Key **PLUS LEFT** — –5 is now in left register
8. Key **WRITE ALPHA (α)**
9. Key **STOP** — 1 second pause
10. Key **WRITE ALPHA (α)**
11. Key **STOP**
12. Key **JUMP IF +**
13. Key **SEARCH**
14. Key **f(01)** — Since –5 is not +, program searches Mark **f(01)** whereupon 1 is added to –5 giving –4. This is repeated until zero is displayed. Then program stops at step 16.
15. Key **RECALL LEFT**
16. Key **STOP**
17. Key **END PROGRAM**

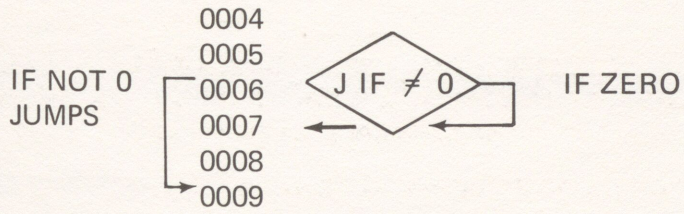
J if 0

The J IF 0 decision-making command tests to see if the left most digit in the display contains zero. If the display contains zero, the next two program commands are bypassed or jumped over. If the value is not zero, the next command following the J IF 0 command will be executed as usual.



J if ≠ 0

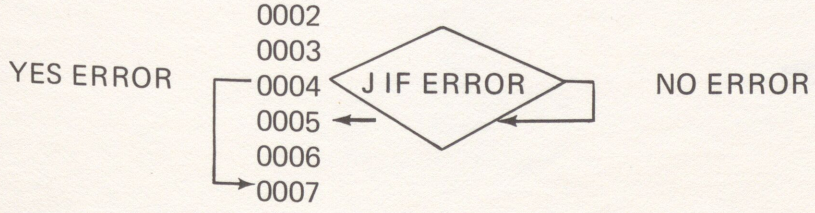
If the upper case J IF ≠ 0 command is encountered in a program, the 600 will automatically examine the contents of the display. If the value in the display is not zero, the 600 will jump over the next two program steps. If the display contains zero, the 600 will read the next program step as usual.



J if ERROR

If the upper case command J IF ERROR is encountered in a program, the 600 will check if an error condition exists. If an error condition does exist, the 600 will jump over the next two programs, the error light will automatically be turned off, and the display will stop pulsating.

The program error light is turned on whenever a non-functional operation is performed (i.e., taking the logarithm of the square root of a negative number, or dividing by zero). Also, if a calculation result is greater than 10^{99} or smaller than 10^{-99} the Program Error indicator will be turned on.



Example 4

Write a program which stores 6 in the left register, subtracts 1 every time it performs a loop and then takes the reciprocal of that number. The calculator will continue looping until it encounters a zero. A provision for division by zero is included in the program.

If the calculator encounters an error, it will automatically execute the next two steps, recalling the data from the left register. The display will have all zeros in it.

Keystroke Operation	Operating Instructions
LEARN MODE	RUN MODE
1. Key **6**	1. PRIME
2. Key **STORE LEFT**	2. GO
3. Key **MARK**	
4. Key **f(00)**	
5. Key **1**	
6. Key **MINUS LEFT**	
7. Key **WRITE ALPHA**	
8. Key **STOP**	
9. Key **WRITE ALPHA**	
10. Key **STOP**	
11. Key **1/X**	
12. Key **WRITE ALPHA**	
13. Key **STOP**	
14. Key **WRITE ALPHA**	
15. Key **STOP**	
16. Key **JUMP IF ERROR**	
17. Key **SEARCH**	

18. Key **f(00)**

19. Key **RECALL LEFT**

20. Key **STOP**

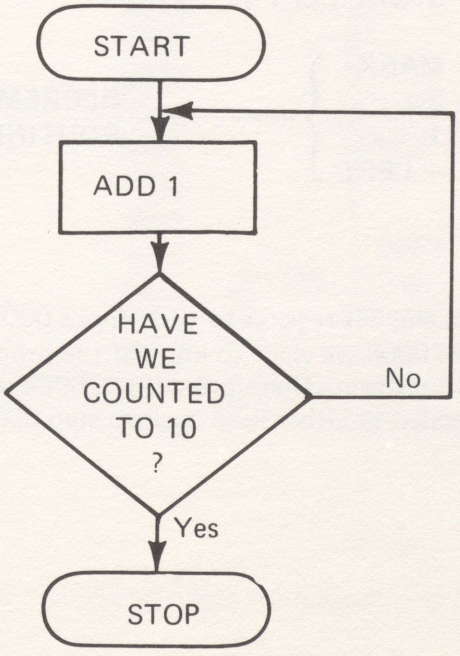
Earlier we stated that looping is an important technique in computer programming. We also stated that without the capability of terminating the loop, this technique is useless. It is like the senselessness of the small boy who runs away from home and who is not allowed to cross the street, he keeps walking around and around the block getting nowhere. Decision making commands enable the small boy to cross the street.

The decision commands make it possible to terminate or exit from a loop and to branch to other programming routines. Let us now practice writing simple programs that make decisions to terminate loops.

Example 5

Write a flow diagram and program that loops 10 times and then stops.

The flow diagram would be:



Before we write the program let us discuss what is involved.

The program is going to loop 10 times. Therefore, we need a routine that will set the loop reference value to 10.

```
MARK
2
1
0
STORE LEFT
```

SETTING LOOP REFERENCE VALUE TO 10

Another group of steps is needed to decrement the loop reference value by 1 each time the program loops. Those steps could be

```
1
– LEFT
```

Because these steps are going to be repeated over and over we need a means of locating these steps. For this reason we will insert a MARK, 3 in front of these steps. Later we will SEARCH MARK, 3.

To this point the program would be:

```
0000    MARK
0001    2
0002    1
0003    0
0004    STORE LEFT

0005    MARK
0006    3
0007    1
0008    – LEFT
```

SETTING LOOP REFERENCE VALUE TO 10

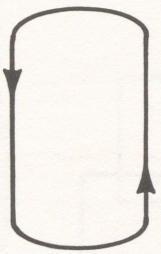
DECREMENTING ROUTINE

Steps 0000 to 0004 set the left register to +10. Steps 0005 to 0008 decrease this value by 1. However, after we execute step 0008 we want to know if the program should loop back to step 0005 and execute the decrementing routine again. Therefore, at step 0009 we include a decision-making command followed by the commands needed to either loop back to step 0005 or to end the program.

The completed program would be:

```
0000    MARK
0001    2
0002    1
0003    0
0004    STORE LEFT
0005    MARK
0006    3
0007    1
0009    – LEFT
0010    J IF 0          (NO)
0011    SEARCH
0012    3
0013    STOP
0014    END PROG
```

(YES) → 0013



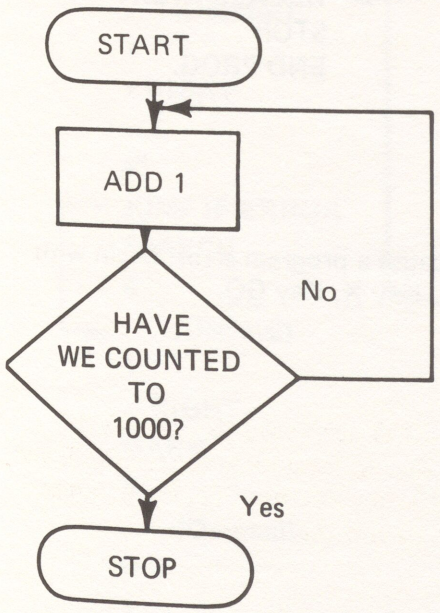
LOOP

Press the STEP key repeatedly to visualize the decrementing and decision routines.

Example 6

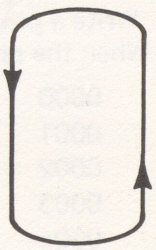
Write the same program as requested in Example 5, except loop 1000 times

The flow diagram and program would be:




```
0000    MARK
0001    2
0002    1
0003    0
0004    0
0005    0
0006    STORE LEFT
0007    MARK
0008    3
0009    1
0010    – LEFT
0011    J IF 0          No
0012    SEARCH
0013    3
0014    STOP
0015    END PROG
```

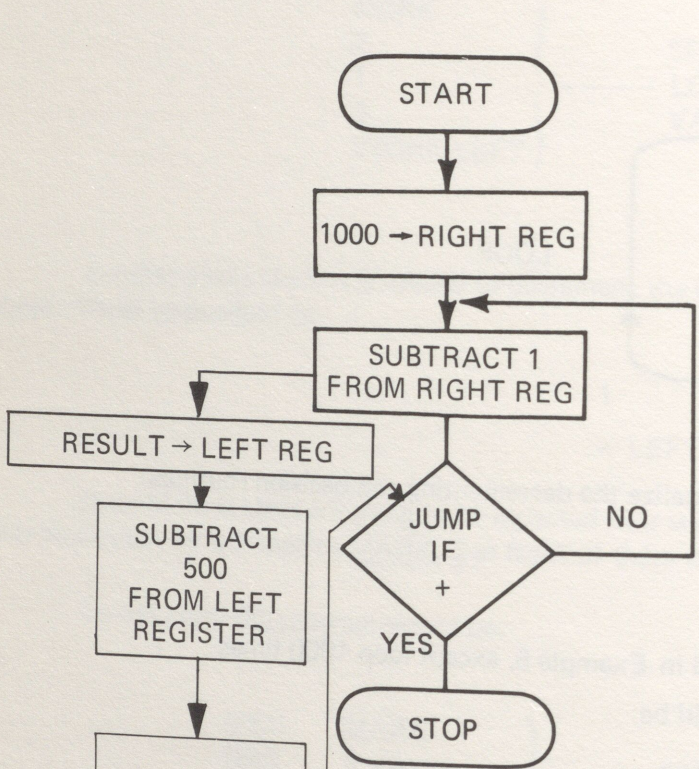
Yes → 0014



This program executes Steps 0007 to 0013 a total of 1000 times before terminating the loop.

Example 7

Write a flow diagram and program that counts from 1000 to 500, by 1. Use the decision command JUMP IF +.




```
0000    MARK
0001      5
0002      1
0003      0
0004      0
0005      0
0006    ST RIGHT
0007    MARK
0008      6
0009      1
0010    – RIGHT
0011    ST LEFT
0012      5
0013      0
0014      0
0015    – LEFT
0016    CH SIGN
0017    J IF +
0018    SEARCH
0019      6
0020    RECALL RIGHT
0021    STOP
0022    END PROG.
```

Example 8

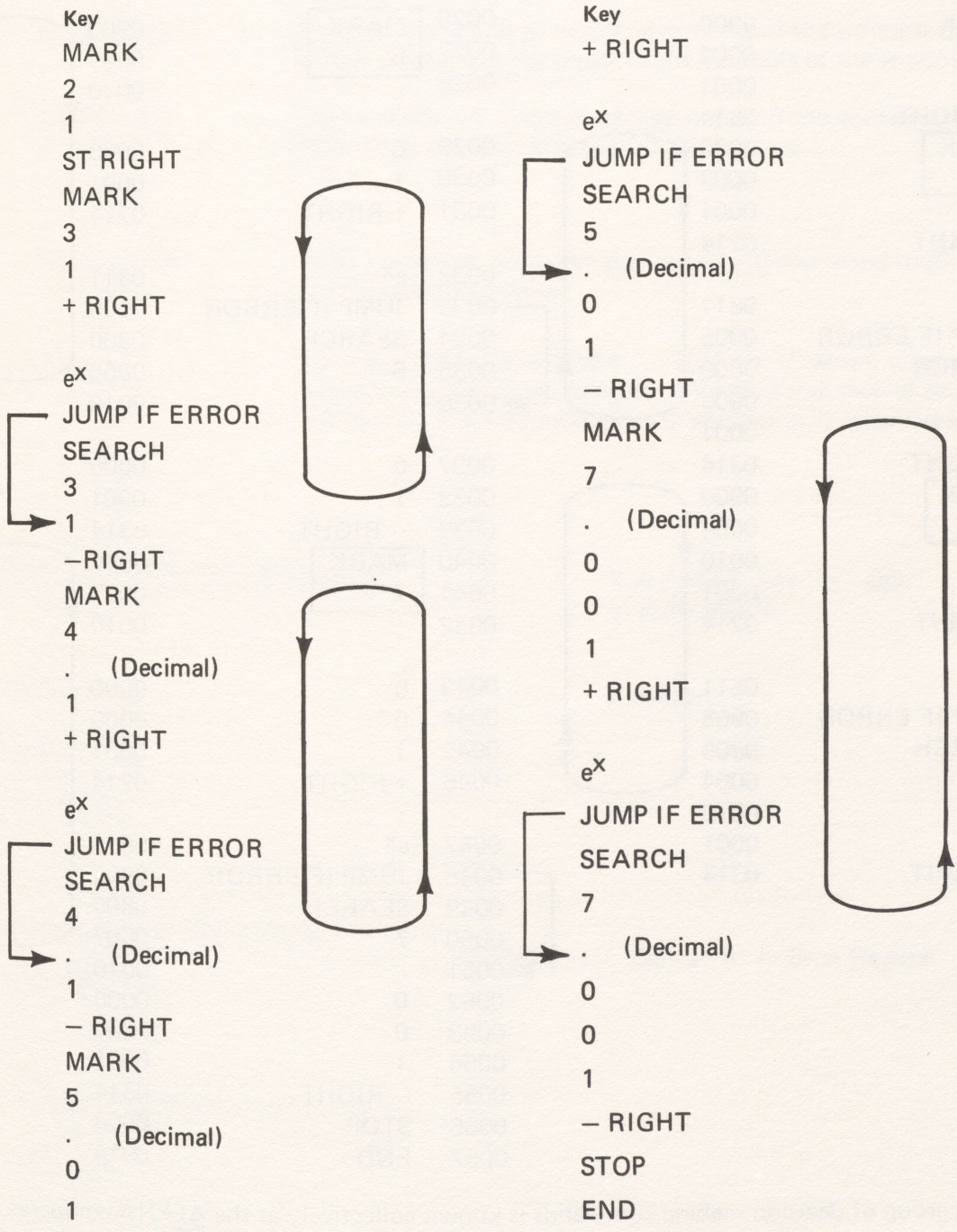
Write a program to increment the value of x by 1 until e^x will cause a program error. Begin with X = 2. When the program stops, the value of x should be in the display. Key X, Key GO.

```
0000    MARK
0001      2
0002    ST RIGHT
0003    MARK
0004      3
0005      1
0006    + RIGHT
0007    e^x
0008    JUMP IF ERROR
0009    SEARCH
0010      3
0011    RECALL RIGHT
0012    STOP
0013    END PROG.
```

Notice that the 231 causes a program error. Does 230 cause a program error? (Hint: Key X, Key STEP repeatedly.) What is the greatest value of x that does not cause a program error?

Example 9

Write a program to find the first value of X that causes a program error for the e^X function. Be sure that the value of X is carried to three place accuracy.



The coded program appears on the next page.

Example 9 (Continued)

Write a program to find the first value of x that causes a program error for the e^x function. Be sure that the value of x is carried to three-place accuracy. The coded program is given below.

Step	Key	Code
0000	MARK	0900
0001	2	0002
0002	1	0001
0003	ST RIGHT	0614
0004	MARK	0900
0005	3	0003
0006	1	0001
0007	+ RIGHT	0214
0008	e^x	0811
0009	JUMP IF ERROR	0905
0010	SEARCH	0800
0011	3	0003
0012	1	0001
0013	– RIGHT	0314
0014	MARK	0900
0015	4	0004
0016		0010
0017	1	0001
0018	+ RIGHT	0214
0019	e^x	0811
0020	JUMP IF ERROR	0905
0021	SEARCH	0800
0022	4	0004
0023	.	0010
0024	1	0001
0025	– RIGHT	0314
0026	MARK	0900
0027	5	0005
0028	.	0010
0029	0	0000
0030	1	0001
0031	+ RIGHT	0214
0032	e^x	0811
0033	JUMP IF ERROR	0905
0034	SEARCH	0800
0035	5	0005
0036	.	0010
0037	0	0000
0038	1	0001
0039	– RIGHT	0314
0040	MARK	0900
0041	7	0007
0042		0010
0043	0	0000
0044	0	0000
0045	1	0001
0046	+ RIGHT	0214
0047	e^x	0811
0048	JUMP IF ERROR	0905
0049	SEARCH	0800
0050	7	0007
0051	.	0010
0052	0	0000
0053	0	0000
0054	1	0001
0055	– RIGHT	0314
0056	STOP	0903
0057	END	0914

The second group of decision-making commands is known collectively as the ALPHA compares.

THE ALPHA COMPARE COMMANDS

The ALPHA compare commands are so called because they involve the use of the ALPHA key, as the first part of a two-part command. You will find these three commands to be useful whenever you wish to compare the contents of two separate data registers, without changing their values.

The three ALPHA Jump commands, which compare the contents of Data Register [00] with the contents of the window are:

ALPHA J IF 0 – The machine will skip two program steps if the contents of Data Register [00] equals the contents of the window.

ALPHA J IF + – The machine will skip two program steps if the contents of Data Register [00] are greater than or equal to the contents of the window.

ALPHA SIN – The machine will skip two program steps if the contents of Data Register [00] are less than the contents of the window.

For example: Write a program, using the ALPHA J IF 0 command that will find the sum of the converging series:

$1 + 1/2 + 1/4 + 1/8 + 1/16 + \cdots$ for "n + 1" terms, where "n" is specified in the beginning of the program. Include a one second pause feature which will display the summation in the window as it progresses.

OPERATING INSTRUCTIONS:

Key PRIME, key "n", GO, in RUN MODE

Keystroke Operations

Comments

LEARN MODE

1. Key **MARK**
2. Key **1**
3. **(St) Selector Switch Down**
4. Key [00]

} Stores "n" in Data Register [00]

5. Key **1**
6. **(St) Selector Switch Down**
7. Key [01]
8. Key **TOTAL RIGHT**
9. Key **1**
10. Key **STORE LEFT**
11. Key **MARK**

12. Key **2**
13. Key **.5**
14. Key **X LEFT**
15. **+ SWITCH DOWN**
16. Key **01**
17. Key **ALPHA (α)**
18. Key **STOP**
19. Key **ALPHA (α)**
20. Key **STOP**

1 Sec. Pause shows sum on screen

21. Key **1**
22. Key **+ RIGHT**
23. Key **ALPHA (α)**
24. Key **J IF 0**

IF contents of Data Register **00** equal the window contents, program skips to

25. Key **SEARCH**
26. Key **2**
27. **(Re) Selector Switch Down**
28. Key **01**
29. Key **STOP**
30. Key **END PROGRAM**

Example 10

Write a program using the **ALPHA SIN** command which will compute and print out the value of all angles θ (in 5-degree increments) less than 360° which satisfy the following relationship:

$$\text{SIN}\,\theta + \text{COS}\,\theta > n, -\sqrt{2} \leqslant n \leqslant +\sqrt{2}\ .$$

"n" is specified in the beginning of the program.

Key PRIME, key "n", GO,

Printer ON, RUN MODE,

"n" will be labeled as "X" by Printer.

"θ" will be labeled as "A" by Printer.

(SIN θ + COS θ) will be labeled as "Y" by Printer.

Keystroke Operations	Comments
LEARN MODE	
1. Key **MARK**	
2. Key **1**	
3. **(St) Selector Switch Down**	Stores "n" in Data Register [00].
4. Key [00]	
5. Key **PRINT**	Prints "n" in floating point format, with I.D. Code "X".
6. **ALL SWITCHES UP**	
7. Key [10]	
8. Key **PRINT**	Double spaces Printer
9. Key [15]	
10. Key **PRINT**	
11. Key [15]	
12. Key **355**	Degree counter stored in [01]
13. **(St) Selector Switch Down**	
14. Key [01]	
15. Key **TOTAL RIGHT**	

16. Key **MARK**
17. Key **2**
18. Key **RECALL RIGHT**
19. Key **SIN**
20. Key **STORE LEFT**
21. Key **RECALL RIGHT**
22. Key **COS**
23. Key **+ LEFT**

Calculates SINθ + COS θ

24. Key **ALPHA** (α)
25. Key **SIN**

*ALPHA Compare Command compares calculation with input "n" in Data Register [00].

26. Key **SEARCH**
27. Key **3**
28. Key **RECALL RIGHT**
29. Key **PRINT**
30. **(–) Selector Switch Down**
31. Key [00]

Prints "θ" with no following zeros.
Identify code = "A".

32. Key **RECALL LEFT**
33. Key **PRINT**
34. **(+) Selector Switch Down**
35. Key [10]

Prints computed value, to 10 decimal places,
I. D. = "Y".

36. Key **PRINT**
37. Key [15]

Single space skip on Printer

38. Key **MARK**
39. Key **3**
40. Key **5**
41. **(–) Selector Switch Down**
42. Key [01]

Subtracts 5° from degree counter.

43. Key **J IF +** ——— If degree counter is still positive, jump to
44. Key **SEARCH**
45. Key **4**
46. Key **5**
47. Key **+ RIGHT** ——— Adds 5° to "θ" and returns to beginning of program for next calculation.
48. Key **SEARCH**
49. Key **2**
50. Key **MARK**
51. Key **4**
52. Key **STOP**
53. Key **END PROGRAM**

Below is a sample output for n = 1.40:

```
1.400000000          X

          40.        A
1.408832052          Z

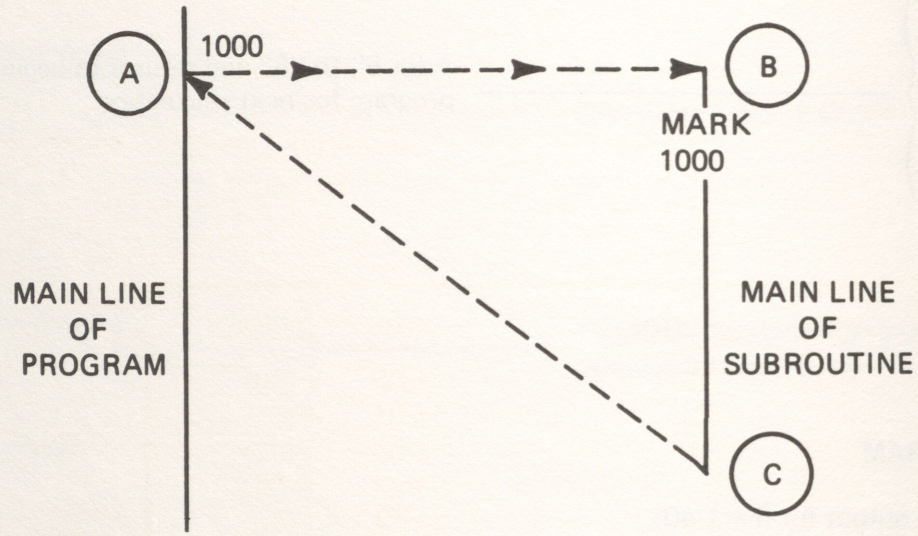
          45.        A
1.414213562          Z

          50.        A
1.408832052          Z
```

SUBROUTINE

Another word for "subroutine" is "subprogram." This word implies that a lesser program is part of a main program. It is conceivable that a program could be made up of several lesser programs. The lesser programs would be called upon by the main program, whenever necessary, to perform a series of intermediate calculations. After a series of intermediate calculations is performed, the subroutine returns control to the main program.

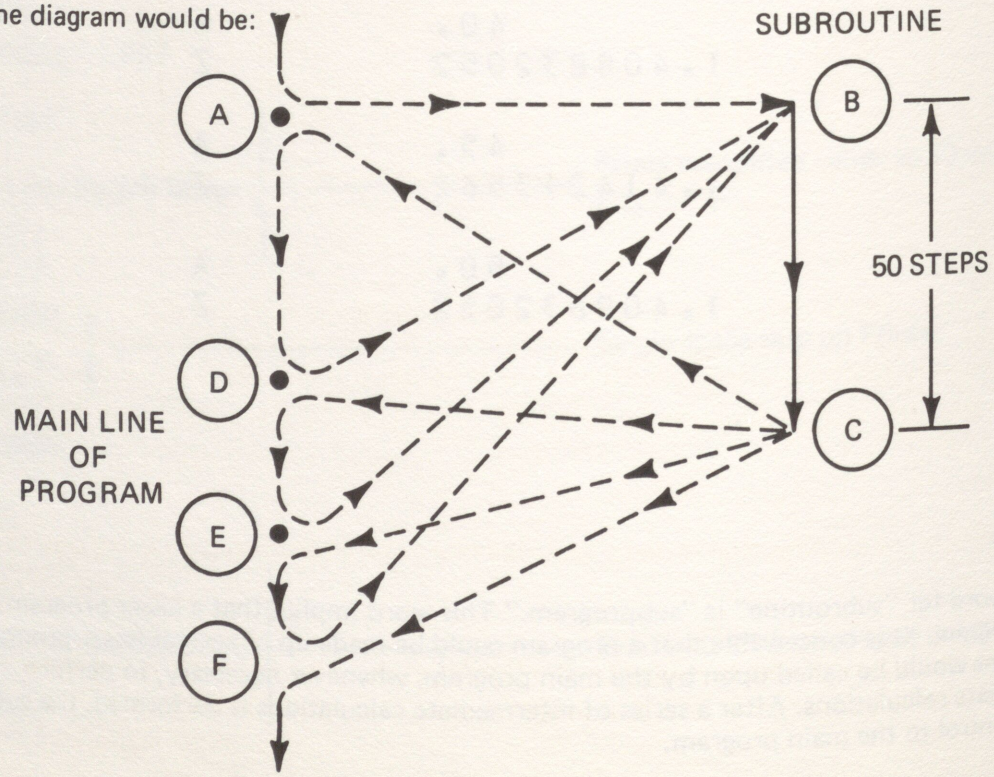
The diagram below represents a program that branches from point (A) to the subroutine beginning at point (B), and then, at the end of the subroutine point (C), returns to the main program at point (A)



Would you venture a guess, at this time, why subroutines are important?

Suppose there are 50 program steps between points (B) and (C) in the above subroutine; and suppose that this subroutine is not only called upon at point (A) to execute the 50 steps, but also at points (D), (E) and (F).

The diagram would be:



The diagram may confuse you more than the concept, but the important thing to notice is that this 50-step subroutine is used 4 times by the main program. The subroutine takes only 50 steps of memory, but executes 200 steps of program operations. With skillful use of subroutines, more effective use of the hardware is possible.

What is the method for writing programs that utilize the subroutine capability of the calculator; What is the coding system needed?

Most programs you write will use no more than a few subroutines. However, to assure an adequate number, almost every key on the keyboard can be used to address a subroutine. The basic set of 32 subroutine codes is given below. The others are called the "15 07 Subroutine Codes", and are described in Appendix B.

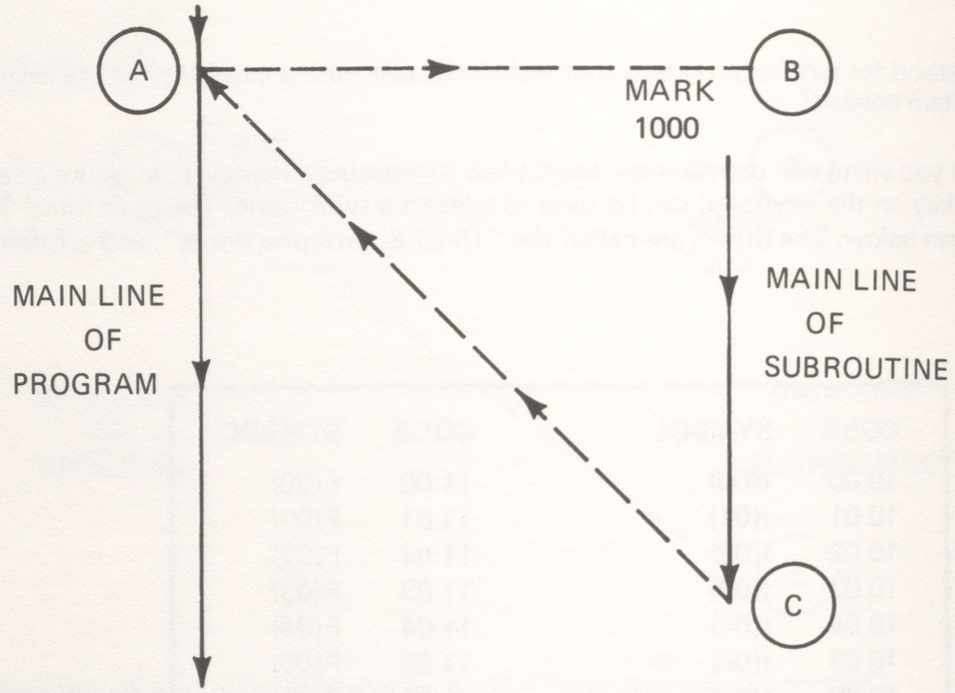
CODE	SYMBOL	CODE	SYMBOL
10 00	f(00)	11 00	F(00)
10 01	f(01)	11 01	F(01)
10 02	f(02)	11 02	F(02)
10 03	f(03)	11 03	F(03)
10 04	f(04)	11 04	F(04)
10 05	f(05)	11 05	F(05)
10 06	f(06)	11 06	F(06)
10 07	f(07)	11 07	F(07)
10 08	f(08)	11 08	F(08)
10 09	f(09)	11 09	F(09)
10 10	f(10)	11 10	F(10)
10 11	f(11)	11 11	F(11)
10 12	f(12)	11 12	F(12)
10 13	f(13)	11 13	F(13)
10 14	f(14)	11 14	F(14)
10 15	f(15)	11 15	F(15)

These subroutine address codes are easy to remember. Notice the first two digits in each column. They are respectively, 10 and 11. Now notice that the second two digits in each column run from 00 to 15. The first two digits of any subroutine code are 10 or 11 and the second two digits range from 00 to 15. [f(x) Switch down and [04] will generate the code 10 04. SHIFT, f(x) Switch down and [04] will generate the code 11 04.]

How would you use a subroutine code to label the main line of a subroutine using one of these codes?

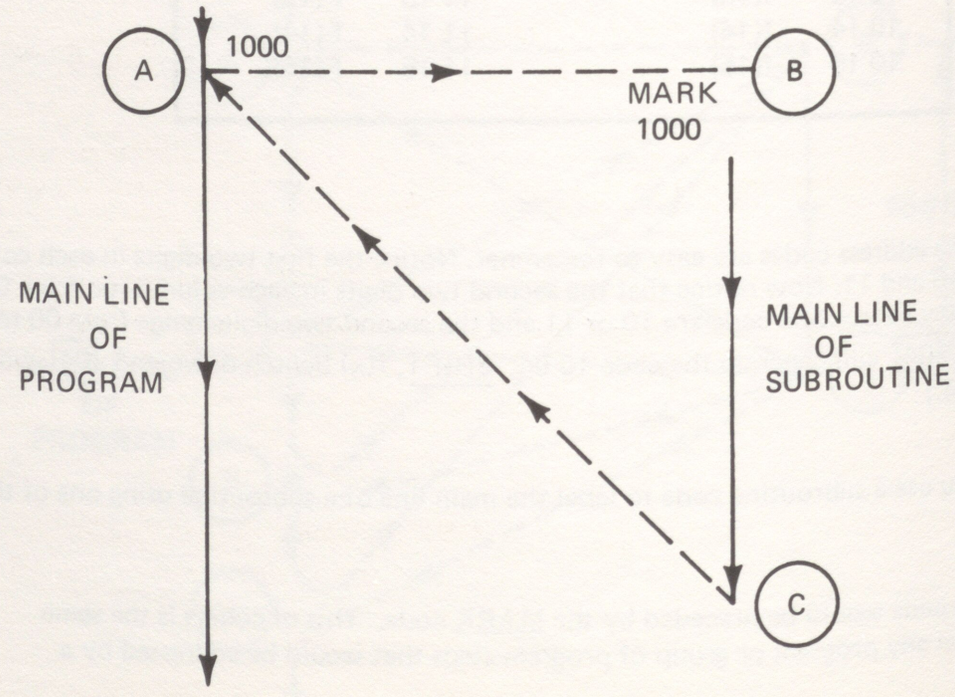
The subroutine code would be preceded by the MARK code. This of course is the same convention used to label any program or group of program steps that would be addressed by a SEARCH command.

The following diagram indicates the use of a subroutine code 1000 to label the main line of a subroutine.



A SEARCH command does not precede the 1000 command to cause the program to branch to subroutine MARK 1000. If the code 1000 is encountered in the main program it will cause the program to automatically branch to the subroutine labeled MARK 1000.

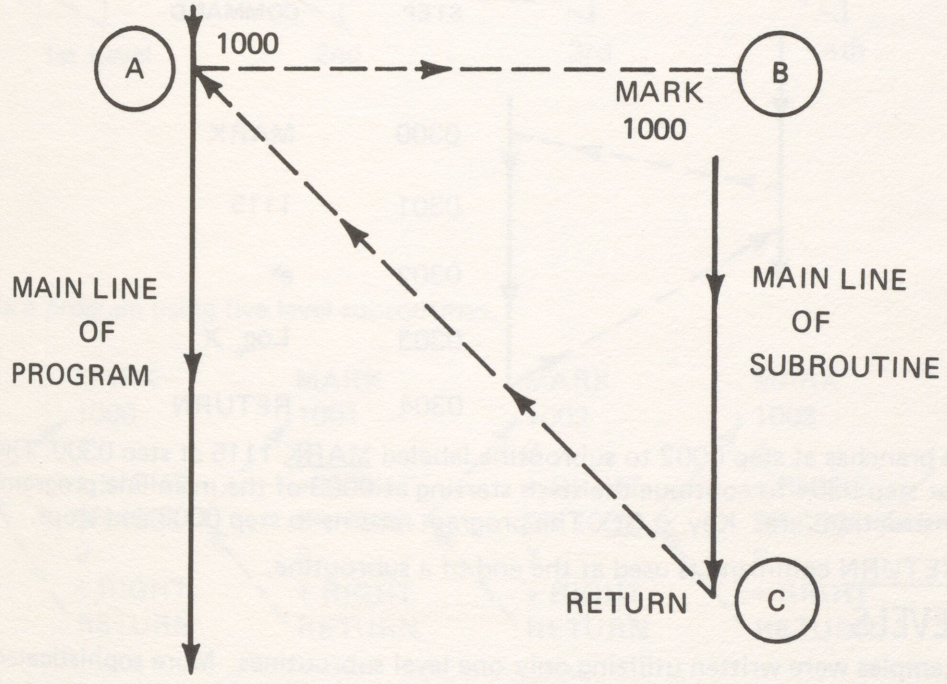
The diagram below indicates the use of the subroutine code 1000 to search a subroutine labeled MARK 1000.



The method of addressing a subroutine and the method of labeling a subroutine has been explained. There is still one question that has to be answered. After a subroutine is executed what is the command needed to return to the main program?

The one-step command RETURN is placed at the end of the subroutine and does just this. The execution of this code will cause the program to branch back to the point in the program from where the subroutine was initially addressed.

The diagram would be:



We will now explain how to write programs using subroutines.

Study the following:

Example 11 To generate the 1013 code, f(x) down and key 13

STEP	COMMAND	CODE
.	.	.
.	.	.
0030	SR1013	1013
0031	.	.
0032	.	.

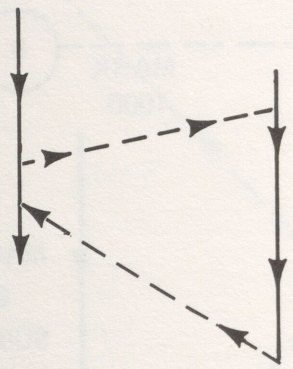
STEP	COMMAND	CODE
0045	MARK	0900
0046	1013	1013
0047	X^2	0812
0048	RETURN	0915
0049	.	.

Note that at step 0030, the program reads a subroutine SEARCH command SR1013. At this step, the program branches to the MARK command that precedes the 1013 command at step 0045. The program then executes the steps in the subroutine until the RETURN command is encountered at step 0048. The program then branches back to step 0031.

Note at step 0030 the subroutine address command 1013 is written when SR preceeding the code. The letters SR are a mnemonic for SEARCH and RETURN; i.e., SEARCH for a subroutine and RETURN to the main line of the program when you see a RETURN command.

Example 12 To generate the 1115 code, Key SHIFT, f(x) down, 15

STEP	COMMAND
0000	MARK
0001	1
0002	SR 1115
0003	STOP
0004	–
–	–



STEP	COMMAND
0300	MARK
0301	1115
0302	e^x
0303	Log_e X
0304	RETURN

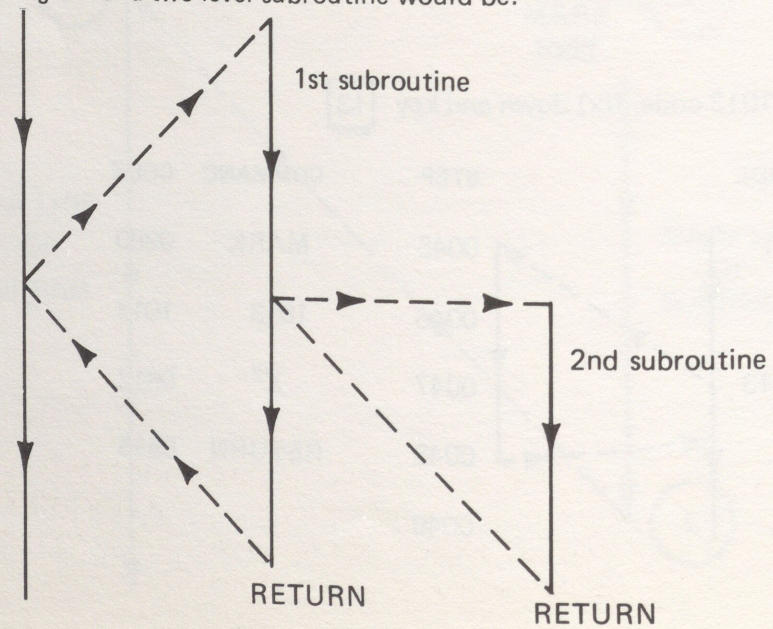
The program branches at step 0002 to subroutine labeled MARK 1115 at step 0300. The program then branches back at step 0304 to continue the steps starting at 0003 of the main line program. In this case, the operating instructions are: Key x, GO. The program returns to step 0003 and stops.

Notice the RETURN command is used at the end of a subroutine.

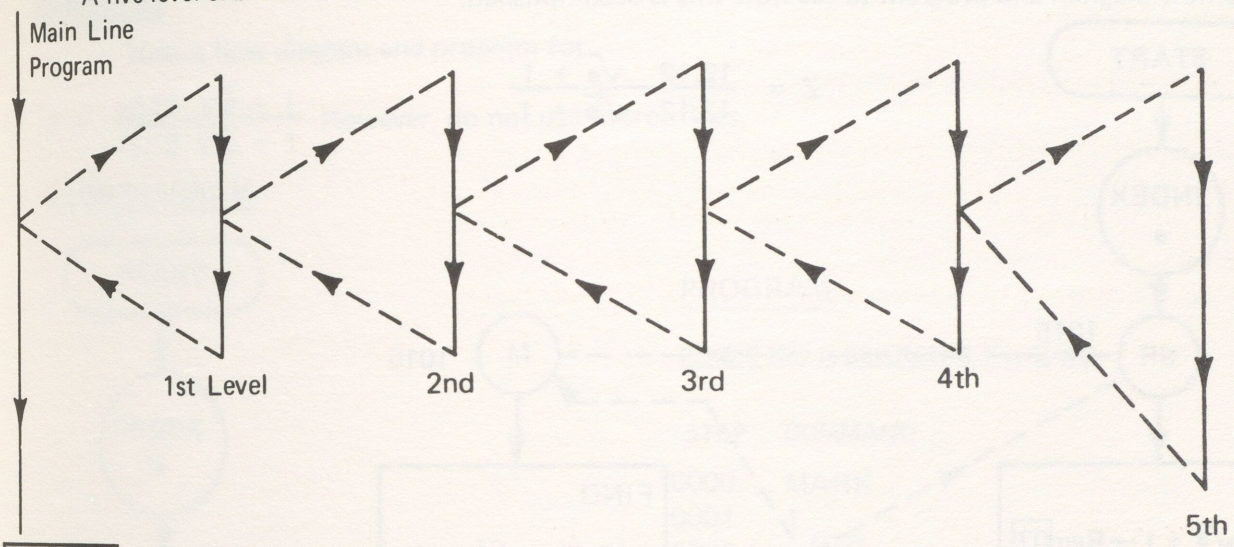
SUBROUTINE LEVELS

The above examples were written utilizing only one level subroutines. More sophisticated programs may include two or more level subroutines. A two level subroutine is a subroutine that addresses another subroutine.

The diagram of a two level subroutine would be:



A five level subroutine would be:



Example 13

Write a program using five level subroutines.

```
MARK          MARK          MARK          MARK          MARK          MARK
1             1000          1001          1002          1003          1004
CLEAR         1             2             3             4             5
SR 1000       + RIGHT       + RIGHT       + RIGHT       + RIGHT       + RIGHT
STOP          SR 1001       SR 1002       SR 1003       SR 1004       RETURN
              9             8             7             6
              + RIGHT       + RIGHT       + RIGHT       + RIGHT
              RETURN        RETURN        RETURN        RETURN
```

What does the above five level subroutine program do? Can this application be more efficiently written without the use of subroutines? Do you know of an application that would require a five level subroutine program?

Example 14

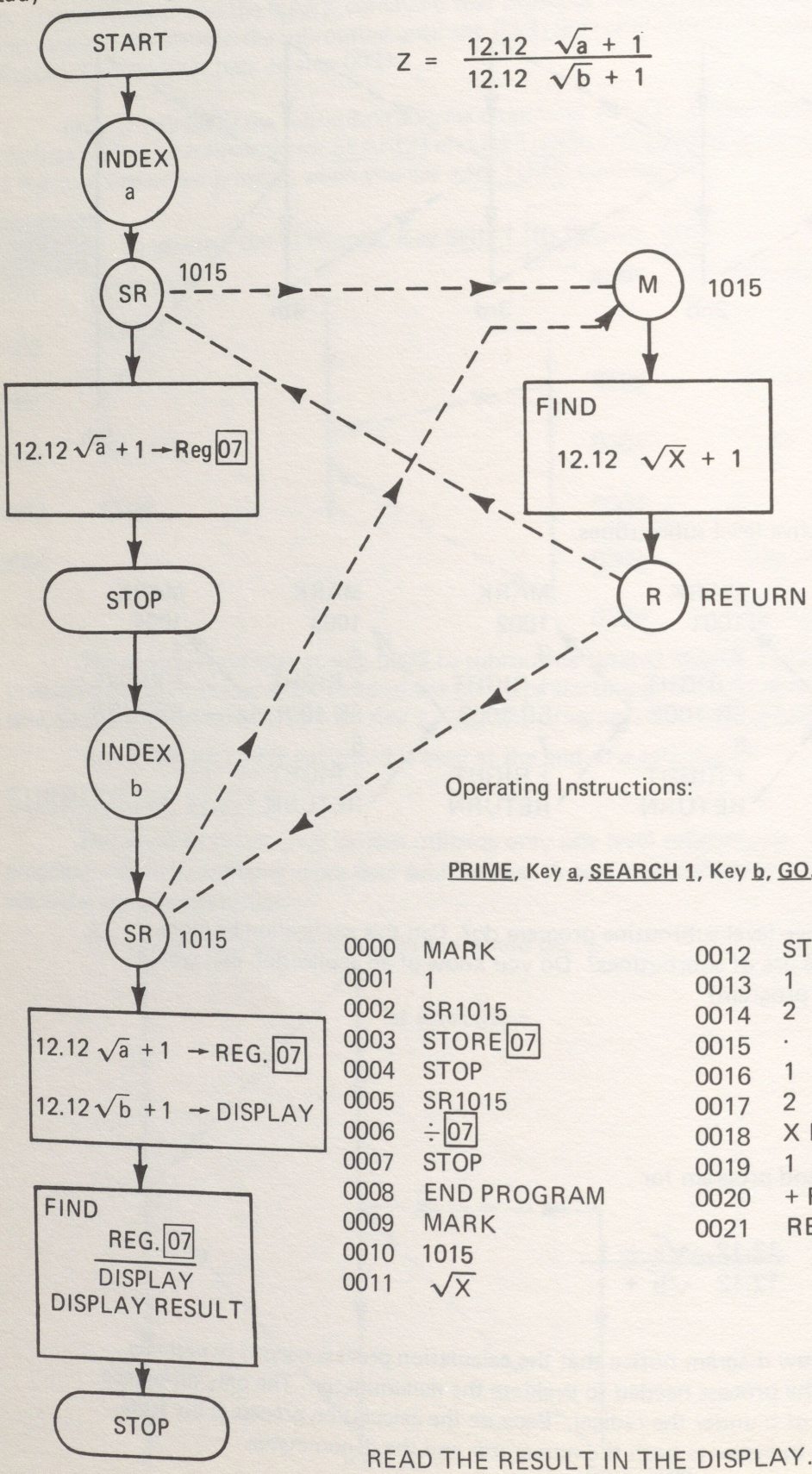
Write a flow diagram and program for

$$Z = \frac{12.12\ \sqrt{a} + 1}{12.12\ \sqrt{b} + 1}$$

Before you start the flow diagram notice that the calculation process needed to evaluate the numerator is identical to the process needed to evaluate the denominator. The only difference is the value of a and the value of b under the radical. Because the calculation process is the same, why not write one subroutine to evaluate both the numerator and the denominator.

Study the flow diagram and program to see how this is accomplished.

$$Z = \frac{12.12\ \sqrt{a} + 1}{12.12\ \sqrt{b} + 1}$$



Operating Instructions:

PRIME, Key **a**, **SEARCH** **1**, Key **b**, **GO**.

0000	MARK	0012	STORE RIGHT
0001	1	0013	1
0002	SR1015	0014	2
0003	STORE 07	0015	·
0004	STOP	0016	1
0005	SR1015	0017	2
0006	÷ 07	0018	X RIGHT
0007	STOP	0019	1
0008	END PROGRAM	0020	+ RIGHT
0009	MARK	0021	RETURN
0010	1015		
0011	√X		

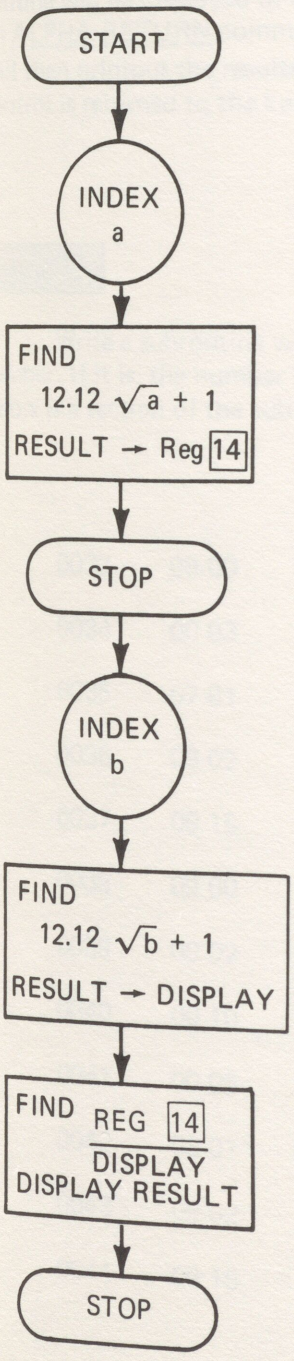
READ THE RESULT IN THE DISPLAY.

Example 15

Write a flow diagram and program for

$$Z = \frac{12.12\ \sqrt{a} + 1}{12.12\ \sqrt{b} + 1}$$ However, do not use subroutines.

FLOW DIAGRAM



PROGRAM

PRIME Key a, SEARCH 1, Key b, GO

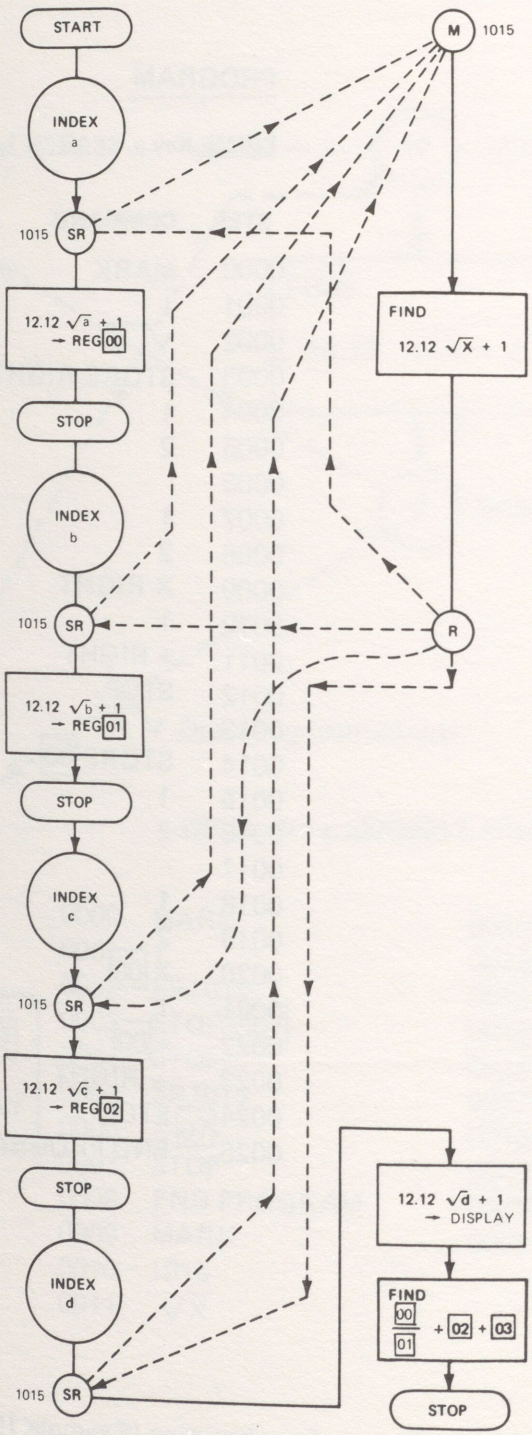
STEP	COMMAND
0000	MARK
0001	1
0002	$\sqrt{X}$
0003	STORE RIGHT
0004	1
0005	2
0006	.
0007	1
0008	2
0009	X RIGHT
0010	1
0011	+ RIGHT
0012	STOP
0013	$\sqrt{X}$
0014	STORE 00
0015	1
0016	2
0017	.
0018	1
0019	2
0020	X 00
0021	1
0022	+ 00
0023	÷ RIGHT
0024	STOP
0025	END PROGRAM

Notice that the program without the use of a subroutine (Example 15) requires 26 steps of core. With a subroutine, the program (Example 14) requires 22 steps of core. Example 16 will exemplify how subroutines can reduce the number of program steps in a program.

Example 16

Write a flow diagram and program for

$$Z = \frac{12.12\ \sqrt{a} + 1}{12.12\ \sqrt{b} + 1} + 12.12\ \sqrt{c} + 1 + 12.12\ \sqrt{d} + 1$$



As an exercise complete this example by writing the program with and without subroutines. Then compare the number of programs steps needed for both methods.

THE ALPHA RETURN AND PRINT FEATURE (AUTO-PRINT ONLY)

As we have seen, one of the advantages of using the Wang 600 is the convenience of being able to call many subroutines in memory by depressing a single key on the keyboard; the results of the subroutine will be displayed in the window. If a printout of the results is wanted in addition to the display, an ALPHA RETURN command should be used in place of the normal RETURN command. The 600 will then printout the results of the subroutine, if the Auto Print Feature has been turned ON, and control is returned to the keyboard.

Example 17

Write a subroutine which will determine whether or not a keyed-in number "n", is a prime number. If it is, the number itself should be printed. If it isn't, .5 should be added to it, and then printed. Below is a section of the subroutine, listed in Appendix D, which does just this.

```
.        .        .
.        .        .
.        .        .
0033     09 00    * MARK
0034     00 03    3
0035     07 01    Recall [01]   }
0036     09 02    * Alpha (α)   }  ——— Returns control to keyboard and initiates a printing
0037     09 15    * Return      }      of the keyed number.
0038     09 00    * MARK
0039     00 02    2
0040     00 10    Decimal Point }
0041     00 05    5             }  ——— Adds .5 to keyed number stored in [01] .
0042     02 01    Plus [01]     }
0043     09 02    * Alpha (α)   }  ——— Returns control to keyboard and initiates a printing.
0044     09 15    * Return      }
```

As you can see, the subroutine has two ALPHA RETURN commands. The first causes the printing of the number keyed in, if the keyed in number is a prime. The second returns control to the keyboard after .5 has been added to the number.

By including an ALPHA RETURN command in your subroutine, you will be assured of a permanent record of your results. Just be sure you turn the Auto-Print feature ON, before you start.

NOTE:

The ALPHA RETURN feature will only work when control is returned to the keyboard. Results will not be printed if the subroutine is called by a program, and control is returned to that program.

EXERCISES

1. What is a subroutine?
2. What is a conditional loop?
3. What is a conditional branch?
4. a. Write a program that will sum the squares of the first 50 counting numbers,

   that is: $1^2 + 2^2 + 3^2 + \ldots + 50^2$.

   b. Write a program that will loop to sum the first 12 factorials, that is:

   $1 + 1 \cdot 2 + 1 \cdot 2 \cdot 3 + \ldots + 1 \cdot 2 \cdot 3 \cdot \ldots \cdot 11 \cdot 12$

   c. Write a program that will loop to sum the reciprocals of the first 12 factorials, as:

   $$\frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \ldots + \frac{1}{1 \cdot 2 \ldots 11 \cdot 12}$$

5. a. Write a program that will sum the squares of N positive integers, that is:

   $1^2 + 2^2 + 3^2 + \ldots N^2$

   b. Write a program that will loop to sum N factorials, that is:

   $1 + 1 \cdot 2 + 1 \cdot 2 \cdot 3 + \ldots + 1 \cdot 2 \cdot 3 \cdot \ldots \cdot N$, or $1! + 2! + 3! + 4! + \ldots + N!$
6. Sum the first N terms of the arithmetic progression a, a + d, a + 2d, . . . , a + (N − 1) d using arbitrary selected values of a, d, and N. For this exercise, do not use the formula $S_N = \frac{N}{2}(a + L)$ where L stands for the last term in the series.
7. Find the least positive value of x (greater than zero) that will yield a real number for

   $$f(x) = \sqrt{x - \frac{1}{4}\sqrt{x}}$$

8. Write a short flow diagram and the program to calculate

   $(x + 1)^{1/3} + (x - 1)^{1/3}$, $x > 1$, where x is keyed in. Write two versions, one version without a subroutine and one version with a subroutine.
9. What is the value of subroutines?
10. Write the flow diagram and the program for:

    $$\frac{e^x - e^{-x}}{2} + \frac{e^{2.1x} - e^{-2.1x}}{2} - x\left(\frac{e^{3x} \quad e^{-3x}}{2}\right)$$

    Use subroutines wherever possible.
11. What is meant by branching?
12. How would you define an integer counter?
13. What is the purpose of a counter?

EXERCISE

THE BIRTHDAY PROBLEM

If there are 20 students in your class, what is the probability of someone else in the class having the same birthdate as you?

The probability of such an event may be expressed as:

$$P = 1 - \frac{365 \cdot 364 \cdot 363 \ldots (365 - K + 1)}{365^K}$$

Where P is the probability of the event
K is the number of students in the class

The probability that someone out of 20 students has the same birthdate is:

$$P = 1 - \frac{365 \cdot 364 \cdot 363 \ldots (365 - 20 + 1)}{365^{20}}$$

The probability that two students will have the same birthday is 0.4114384002.

As the number of students in a class increases, so does the probability of common birthdates. The probability is 0.994 for 60 students in a class.

14. Show the flow diagram for the expression

$$P = 1 - \frac{365 \cdot 364 \cdot 363 \ldots (365 - K + 1)}{365^K}$$

15. What is the probability that two students of a class of a. 10, b. 20, c. 30, d. 40, e. 50, f. 60, g. 70, h. 80, i. 90, and j. 100, have the same birthday?

16. What number of students in a class of 20 students will give a 50% probability that two students have the same birthday?

Chapter 9

DEBUGGING

The most frustrating thing to a programmer, next to going up to a computer and finding that the electricity has been turned off, is to discover that the program he has spent a week writing does not work.

He may stand there and stare at the computer or slowly pace back and forth trying to figure out what is wrong. But, as soon as he starts to scratch his head and to pull at his hair..Stand back .. He has bugs!

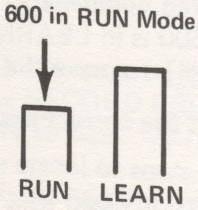
The bugs of course are not the type of bugs that are easily gotten rid of by purchasing a can of Raid or Black Flagg. Instead, these are the type of bugs that keep a program from working. It may take hours, days and sometimes even weeks of analysis to debug the program of an error.

These errors can be in the form of an incorrectly keyed program or they could be in the form of incorrect logic on the part of the programmer.

At any rate, programs attract bugs and cause programmers undue frustrations. To help the programmer "debug" his programs, the Wang 600 Computing System has several debugging features. Take a few minutes now to learn how to use these features.

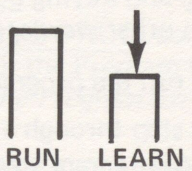
NOTE:

To use most of the keys on the 600, either the RUN Mode or LEARN Mode Switch is depressed. Many of the features described in this section, however, make use of the RUN/LEARN Mode. To put the 600 in the RUN/LEARN Mode, just depress both the RUN and LEARN Switches simultaneously.



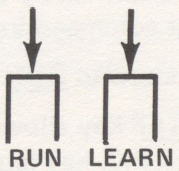
600 in RUN Mode

RUN Switch Depressed



600 in LEARN Mode

LEARN Switch Depressed



600 in RUN/LEARN Mode

RUN and LEARN Switches both Depressed

Example 1

Key the following program for the solution of the Pythagorean Theorem $c = \sqrt{a^2 + b^2}$ into memory.

Step	Key	Code
0000	MARK	0900
0001	0	0000
0002	X^2	0812
0003	STORE RIGHT	0614
0004	STOP	0903
0005	X^2	0812
0006	− RIGHT	0314
0008	$\sqrt{X}$	0813
0009	STOP	0903
0010	END PROGRAM	0914

Note

Do not forget to set the 600 to LEARN MODE

The program is written to find $c = \sqrt{a^2 + b^2}$. Now key in a = 4, SEARCH O, and then b = 3, GO. Notice that we get the incorrect answer of 2.64575. We know that c should be 5. Something is wrong with the program.

We will now attempt to debug this program by stepping through the program to see how each individual program step affects the overall process.

Each time the green colored STEP key is depressed, the calculator will execute one program step. If the command at that program step causes a change in the display, you will be able to see that change.

Put the 600 in the RUN mode. Key PRIME, key 4, and depress the STEP key three times. We have just executed the first three program steps. Note that the display contains 16, which is 4^2.

Depressing the STEP key again also moves 16 to the display.

At this point in the program, we must key in our second variable b = 3, and depress the STEP key twice. We now have stepped through steps 0004 and 0005. Step 0005 performed the X^2 operation on b.

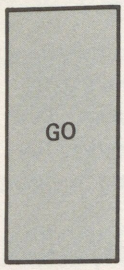
When we depress the STEP key again, we notice that the sum $a^2 + b^2$ does not appear in the display, instead, we see the difference, $a^2 - b^2$. This immediately tells us that the error is a minus command in place of a plus command.

At step 0006 the minus command is to be changed to a plus command. This is easily accomplished by setting the 600 to LEARN mode. Now set the Program Count (P.C.) to the step number that has the error. This is performed by depressing SET P.C. and keying 0006. Because the 600 is in LEARN mode, key the correct command into the computer. In other words, depress the + RIGHT to make the correction.

Now return to RUN mode, PRIME and run the program to see if you get the correct result.

The STEP key allows us to sequentially step through a series of program steps to locate an error. The SET P.C. key lets us bypass a series of program steps and go directly to the step that contains the error. The LEARN mode feature does two things. First, it lets us see the step number and the code in the display. Second, it allows us to make a correction.

It is a good practice to test a program before it is used. As in the example just given, key in numbers for which the solution is known. Check the result in the display and compare it to what is expected. If they don't agree, you should then take a closer look.



Using the GO command as part of a program can be another useful technique in debugging programs. In addition, if we have a STOP in the program, we can continue to the next step after the STOP command by indexing the GO key. At times it is advantageous to use the programmable key GO as a "do nothing" or "no-operation" command. Such commands may be put almost anywhere (except between two-step commands) in a program. The command may also be used to resume automatic execution after the STEP key has been used to perform one operation at a time.



USING THE B.S. (BACKSTEP) KEY

Suppose we were examining a program and found ourselves in the following situation:

The screen shows program step No. 0155, but we want to see step No. 0154. We could press SET P.C. and then key in the 0154, or we could do the following:

a. Depress both the RUN and LEARN Mode Switches.

b. Depress the B. S. (BACKSPACE) Key.

Result: Program Step 0154 will appear in the window.

We could BACKSTEP all the way to Step 0000 if we needed to. All we'd have to do is press the B. S. key again and again. With the 600, it's as easy to BACKSTEP as it is to STEP forward.

PROGRAM BLOCKS

A program block consists of any part of a program which can be loaded into memory at one time.

The block must be terminated by an END PROG. command. If the END PROG command is not given and the RECORD PROG. key is indexed, the Program Error Indicator will go on after transferring all of memory to tape. *See the discussion on the Program Error Indicator.*

In Model 600-2 a program block can have a maximum of 312 steps.

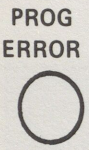
In Model 600-6, a program block can have a maximum of 824 steps.

In Model 600-14, a program block can have a maximum of 1848 steps.

PROGRAM ERROR INDICATOR

The 600 automatically checks for certain types of errors in a program. If there is a non-functional mathematical operation, like division by zero, or if the capacity of the machine is exceeded, the PROG ERROR indicator will be turned on. Besides these two types of errors, the indicator will be turned on if (1) you SEARCH for a NON-EXISTENT MARK, or (2) if an END PROG code has been omitted at the end of a program block in which you are using the INSERT or DELETE key, or are attempting to record on tape.

By locating the program step that activates the PROG ERROR indicator you can quickly determine the program bug.



In summary, the PROGRAM ERROR indicator is located in the upper right-hand corner of the keyboard. If a number is indexed which is greater than 12 digits, the PROGRAM ERROR indicator will light up and the entire display will flash off and on indicating an illegal operation. To correct this condition, depress the PRIME key or JUMP IF ERROR key. The PRIME key will clear the display. Jump if Error will show the most significant digit.

The PROGRAM ERROR INDICATOR will also light up for any of the following operations:

1. Calculated results exceeding 10^{99}. (This is what is known as an overflow condition.)
2. Dividing by Zero.
3. Square root of a negative number.
4. Natural log or the common log of a non-positive number.
5. Searching for a non-existent mark.
6. Exceeding capacity of memory with tape.
7. No "END PROG" command under VERIFY or RECORD.
8. SIN, COS, TAN of an angle greater than 10 radians or 572^{o}.
9. SIN^{-1}, COS^{-1} of a number whose absolute value is greater than 1.

VERY IMPORTANT

Note:

If an END PROGRAM instruction is at Step 0311 for Model 600-2, Step 0823 on Model 600-6, and Step 1847 on Model 600-14, the program will load properly, but the Error Indicator will go on. Even though the END PROG command is not missing, the Program Error Indicator will go on when this program is transferred to tape. If this is the case, simply PRIME and ignore the Program Error Indicator.

VERY IMPORTANT

Besides having an automatic PROGRAM ERROR indicator, the 600 has an indicator that automatically checks certain machine operations.

MACHINE ERROR INDICATOR

MACH ERROR

A MACHINE ERROR indicator is located just to the right of the PROGRAM ERROR indicator. If data is not transferred properly from or to the tape, the light will go on and the sign of the display flashes. This indicates that the information has not been transferred properly and the operation should be repeated. This flashing light should NOT be confused with the PROGRAM ERROR indicator located just to its left. The Prime key will turn both error indicators off.

An operator may have loaded the wrong program into the 600 and, for this reason, it would appear there is a bug. The VERIFY PROG key will help you identify, numerically, what program is in the machine.

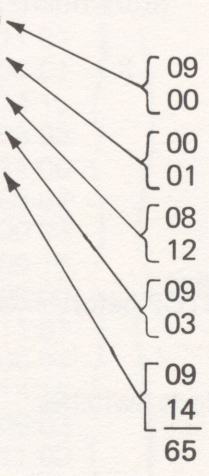
VERIFY PROGRAM

The VERIFY PROG key decimally adds the first two digits and the second two digits of the program codes into core beginning at step 0000 until it encounters an END code. The sum is displayed in the left most digits of the display.

Example 2

Step	Key	Code
0000	MARK	0900
0001	1	0001
0002	x^2	0812
0003	STOP	0903
0004	END PROG	0914

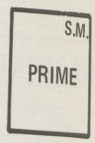
If this program is located in core and the VERIFY PROG key is depressed, the sum displayed is 65.



After the execution of the VERIFY PROGRAM operation the Program Counter is set at the END PROG. step.

After the performance of this operation, the Program Counter is set at the step where the END command is located. (Step 0004 in this example.)

THE MARK SEARCH AND STOP – S. M. KEY



As you already know, you can go directly to a routine within a program, and the machine will automatically execute the commands associated with that routine. To do this, all that is necessary is to key SEARCH, and then key the number corresponding to the MARK you are searching for (see page 56). However, during some debugging procedures, you may want to locate a MARK in a program and then stop before executing. You can do this easily on the 600 by using the RUN/LEARN Mode, the S. M. key, and the numerical keyboard.

Example 3

The following diagram shows a MARK within a program:

	.	.	.
	.	.	.
	.	.	.
	0215	00 01	1
	0216	08 15	1/X
	0217	09 00	MARK
After SEARCH MARK procedure, display will show →	0218	00 04	4
	0219	06 14	Store Right
	0220	08 12	X^2
	.	.	.
	.	.	.
	.	.	.

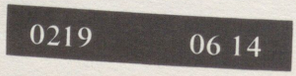
If we want to SEARCH for this MARK and STOP before executing the commands, we can do the following:

1.) Depress both the RUN and LEARN Mode Switches.

2.) Key S. M. (PRIME)

3.) Key 4

Result: The window will then show the first program command after the MARK number (after MARK 4 above.)



As you can see, this two-step MARK SEARCH and STOP function is very useful in long programs, or when we must search for a MARK in a program, but don't know its step number. This could result from several things, such as INSERT STEP or DELETE STEP, as described in the following sections. Either of these could change the step number associated with the program MARK.

USING THE DEL KEY TO DELETE STEPS IN YOUR 600 PROGRAM



As you know, one of the biggest headaches a programmer has is debugging the program he has worked so hard to write. Not only must he often re-arrange entire sections of his program, taking a section out here and putting it back there, but he must delete and insert individual steps as well. Depending on the complexity and length of the program, this can be a major task - - on any machine except the WANG 600, that is. Let's consider the STEP DELETION capability.

Example 4

J. P. Grammer has just entered the first part of his lengthy and complex program, and after standing back and admiring his work, notices he has entered a 7-step subroutine not once, but twice. (A printout would show the following):

```
.        .
.        .
.        .
0146     02 15
0147     09 15
0148     09 00
0149     11 01
0150     06 01
0151     14 15
0152     06 03
0153     11 00
0154     08 02
0155     00 03
0156     09 15
0157     09 00
0158     11 01
0159     06 01
0160     14 15
0161     06 03
0162     11 00
0163     08 02
0164     00 03
0165     09 15
0166     09 00
0167     10 07
0168     11 01
.        .
.        .
.        .
```

= Identical Subroutines

Fortunately, J. P. realized he had one too many subroutines. To correct the situation, J. P. coolly calculated that a DELETE STEP procedure was called for. He proceeded as follows:

First, he keyed SET P. C. and 0157.
The window showed

0157	09 00

Next he depressed both the RUN and LEARN Mode Switches, and keyed DEL (the DELETE STEP Key) once.

The window then showed

0157	11 01

That is, program command (09 00) vanished from the program, and command (11 01) took its place as Step No. 0157.

A printout of the program confirmed his suspicions:

BEFORE

0154	08 02
0155	00 03
0156	09 15
0157	09 00
0158	11 01
0159	06 01
0160	14 15
0161	06 03
0162	11 00
0163	08 02
0164	00 03
0165	09 15
0166	09 00
0167	10 07

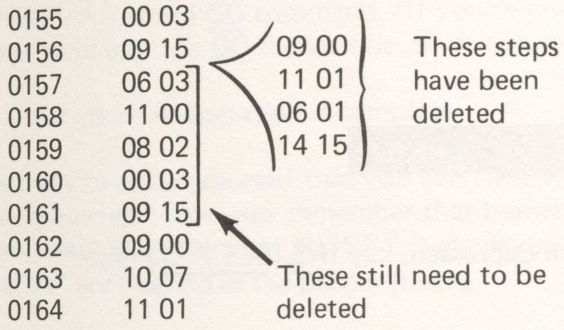
AFTER

0154	08 02
0155	00 03
0156	09 15
0157	11 01
0158	06 01
0159	14 15
0160	06 03
0161	11 00
0162	08 02
0163	00 03
0164	09 15
0165	09 00
0166	10 07

Not only had the program command been deleted, but all the following program steps were moved up one place in line as well – repacking the program into the smallest available number of spaces.

Figuring that he could continue to DELETE steps by the above process, he keyed DEL (with both the RUN and LEARN Mode Switches still depressed) three more times.

A printout of a section of the program revealed the following:



After keying DEL five more times, with both the RUN and LEARN Mode Switches still depressed, J. P. found he had completely eliminated the unwanted subroutine.

You will find it just as easy to eliminate unwanted steps in your programs.

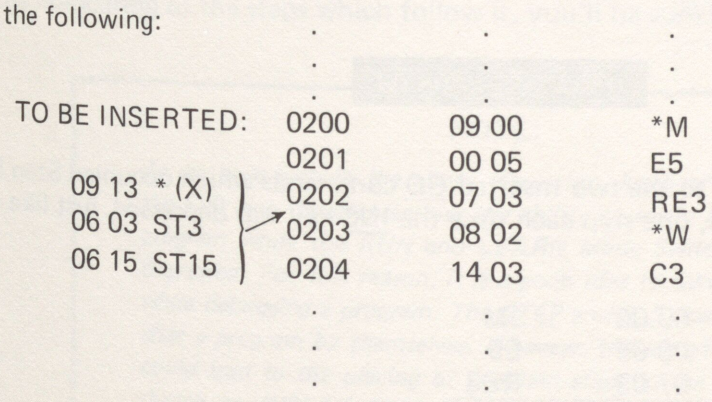
USING THE INS KEY TO INSERT STEPS IN YOUR 600 PROGRAM

INS.
SET P.C.

Now that we can jump to any MARK in a 600 program (using the SEARCH and STOP capability), move forward and backward within the program (using the STEP and B. S. features respectively), and DELETE any unwanted steps at will (using the DEL feature), there is only one function left to complete our debugging package — — the INSERT STEP feature. By giving the programmer the ability to add program steps anywhere in the program easily, this capability is one of the most powerful debugging features on the 600.

Example 5

Study the following:



Note that we wish to INSERT three steps between program step No. 0202 and step No. 0203. To do this, we must first make room for the new steps, and then key in the new steps.

A. INSERTING SPACE INTO THE PROGRAM

1. SET P. C. to Program Step Number 0203.

The window would display

0203	08 02

2. With the RUN and LEARN Mode Switches still depressed, key INS (SET P. C.) once.

The window would then display

0203	08 03

(i.e., the GO command)

A printout of the program section would look like this

0200	0900	*M
0201	00 05	E5
0202	07 03	RE3
0203	08 03	GO
0204	08 02	*W
0205	14 03	C3

In other words, what we did was INSERT a GO command after Step Number 0202, and then move all the remaining program steps one step back, in effect making room for the new step.

We now have successfully allowed space for the insertion of one of our three commands. We need two more spaces to INSERT the remaining steps into memory. Therefore, (with the RUN and LEARN Mode Switches still depressed), we key INS (SET P. C.) two more times.

Since the window still displays:

0203	08 03

what do you think happened to the two Inserted GO commands which occupied Step Number 0203? They were moved back as well, one step each time the INS key was depressed, just like all the rest of the steps:

0200	09 00	*M
0201	00 05	E5
0202	07 03	RE3
0203	08 03	GO
0204	08 03	GO
0205	08 03	GO
0206	08 02	*W
0207	14 03	C3

Now we are ready to key in our new commands.

B. INSERTING THE NEW STEPS IN THE PROVIDED SPACE

1. Backstep to the first Inserted GO command. (If you're not there already).
2. Key in the new steps over the GO commands, thus replacing them.

RESULT – – our insertion operation is complete.

The best way to assure yourself that you can INSERT extra steps when you need to, is to practice the technique right now. Just remember that before you INSERT a space into the program, the program step you want to follow your INSERT is displayed in the window. This will assure that the GO command "spacers" are inserted in the proper places.

The complete program update is shown below:

```
.          .          .
.          .          .
.          .          .
0200       09 00      *M
0201       00 05      E5
0202       07 03      RE3
0203       09 13      *X
0204       06 03      ST3
0205       06 15      ST15
0206       08 02      *W
0207       14 03      C3
.          .          .
.          .          .
.          .          .
.          .          .
```

SUGGESTION: If you must INSERT steps in a program in more than one location, you will find it easier to locate the proper places if you INSERT the last set of steps first. That is, if an INSERT is wanted between steps 7 and 8, and also between steps 35 and 36, make the latter INSERT first. Since an INSERT changes only the numbering of the steps which follow it, you'll be sure of finding the location of the other steps easily.

NOTE:

As we have seen in the above discussion, keyboard controls of the 600 give the programmer the ability to easily alter a loaded program while the RUN and LEARN Mode Switches are both depressed. For this reason, it is a good idea to take special care while debugging a program. The STEP and B. S. features will not alter a program by themselves. However, these keying operations could lead to the placing of program steps in the wrong places during an INSERT STEP procedure. In addition, the INSERT STEP (INS) and DELETE STEP (DEL) features could cause programming errors if the keys are depressed at the wrong time. For this reason, it is a good idea to make a list of all changes to be made in a program and determine the new VERIFY CODE before altering the program itself. This way, you can check to see that you have made only the program changes you wanted to make, and no others.

NOTE:

Be sure that the steps you INSERT and/or DELETE do not change the logic of the program in ways you didn't intend. For example, Inserting or Deleting a step directly after a J if 0 (or any other Jump command) will change the flow of the program and may cause the program to "hang-up." Similarly, the Deletion of part of a MARK command may cause the machine to misinterpret the associated steps. In other words, be certain of your program changes before you make them —— then you won't be surprised by the results.

When you attempt to develop an extensive and lengthy program it is suggested that you test the various parts and components of the program separately before you join the parts together. This will assure that the basic routines are working properly. Even though all parts may work when tested separately, there is no assurance that the final program will work when all parts are joined together.

If, at this point, the final program does not work, at least you can now focus your effort on, "how each part of the program is interrelated to each other part."

Another thought on debugging is worth mentioning. When you attempt to remove a bug from a program, you are apt to create another bug. Carefully study the ramifications of any change you propose.

Chapter 15 discusses the programmable ALPHA (α) Trace feature .... another very useful and effective debugging tool.

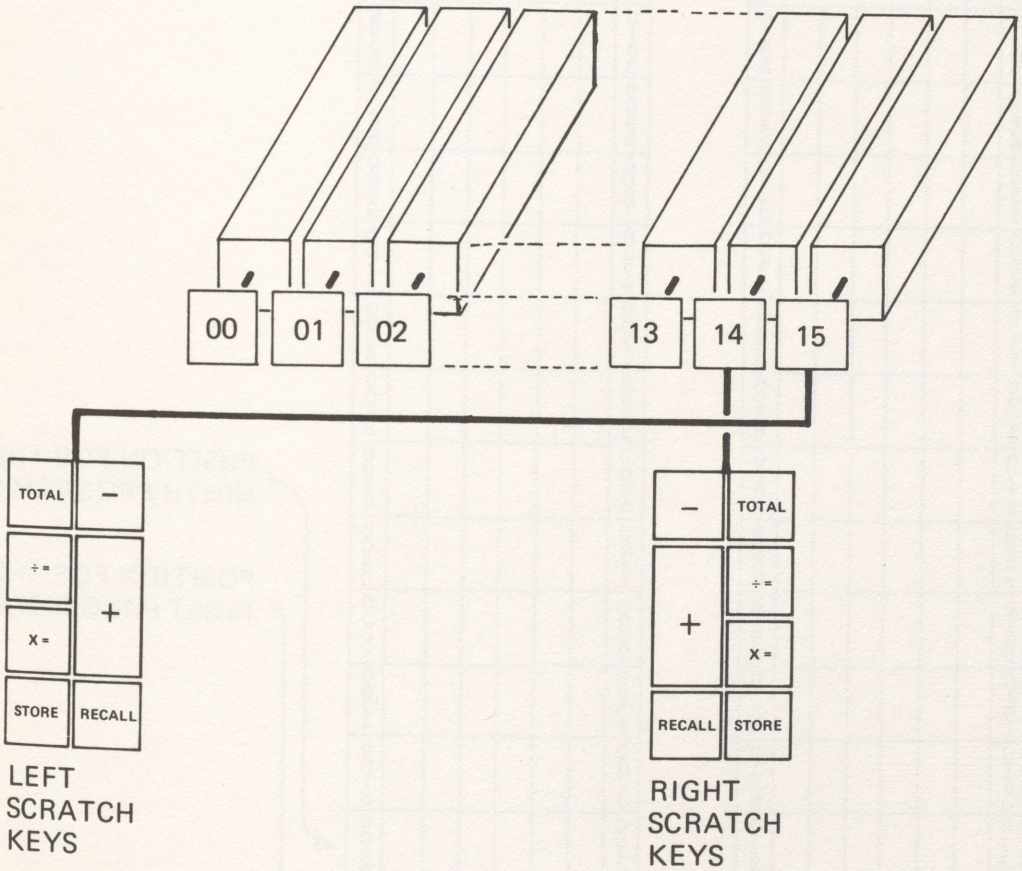
Chapter 10

DATA STORAGE

There are sixteen basic data registers available to the programmer. As you know, these data registers are addressed using the Data Register keys labeled [00] to [15] and the Function Selector switches.

Registers [14] and [15] are the right and left hand SCRATCH REGISTERS, respectively.

Study the diagram below.



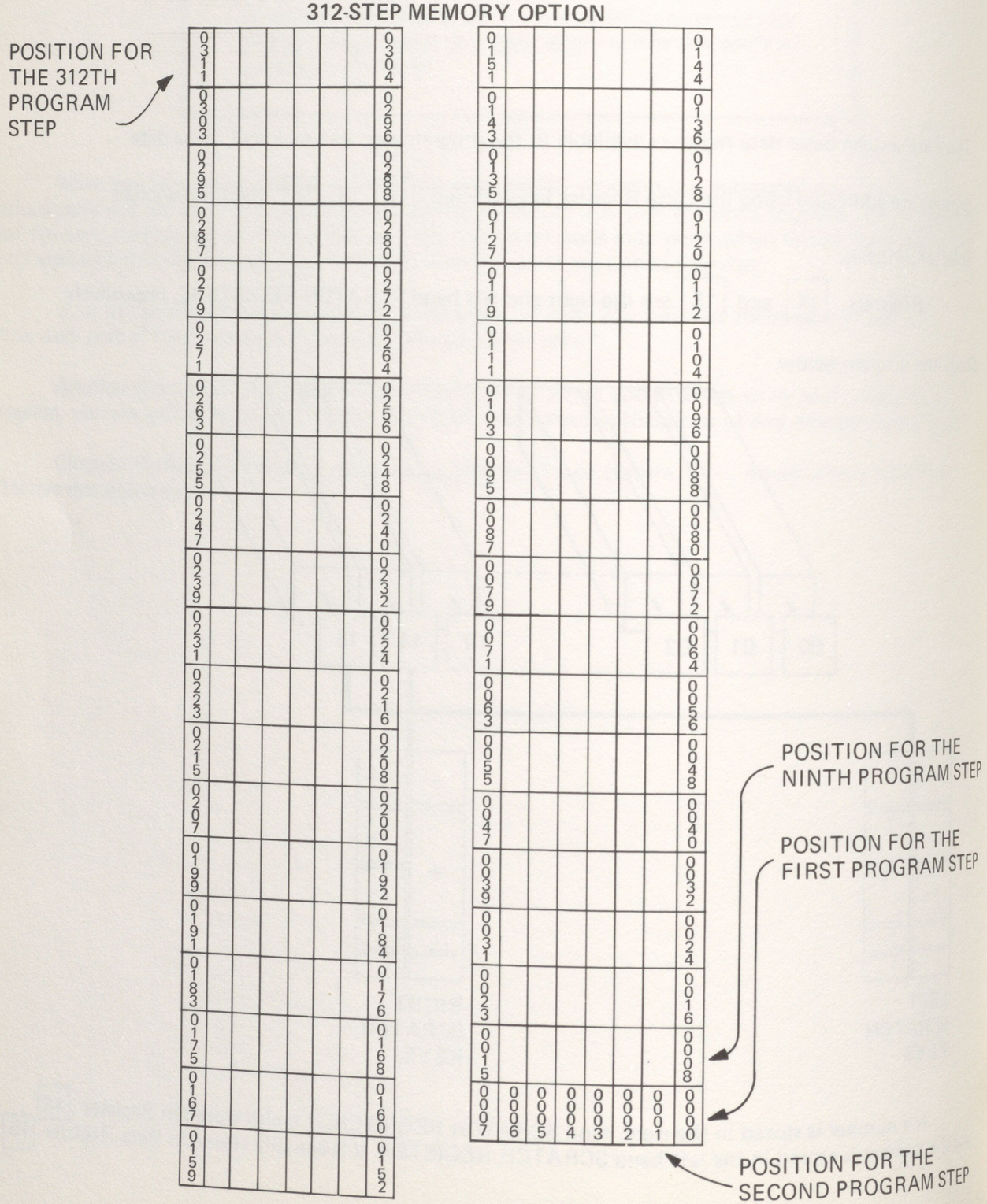
If a number is stored in the right hand SCRATCH REGISTER, it is also stored in Register [14], and if a number is stored in the left hand SCRATCH REGISTER, it is actually stored in Data Register [15].

When writing a program that requires the use of data registers, it is generally best to start by using the lowest numbered data registers first; i.e., [00] , [01] , [02] and so on up to Register [13] . Data Registers [14] and [15] are the right and left hand SCRATCH REGISTERS for storing immediate results.

A 600 Series calculator may contain 312 program steps, or 824 program steps, or 1848 program steps, depending upon the preference of the purchaser. We will begin our study with the 312 program step configuration.

Study the diagram of the 312-step system.

312-STEP MEMORY OPTION



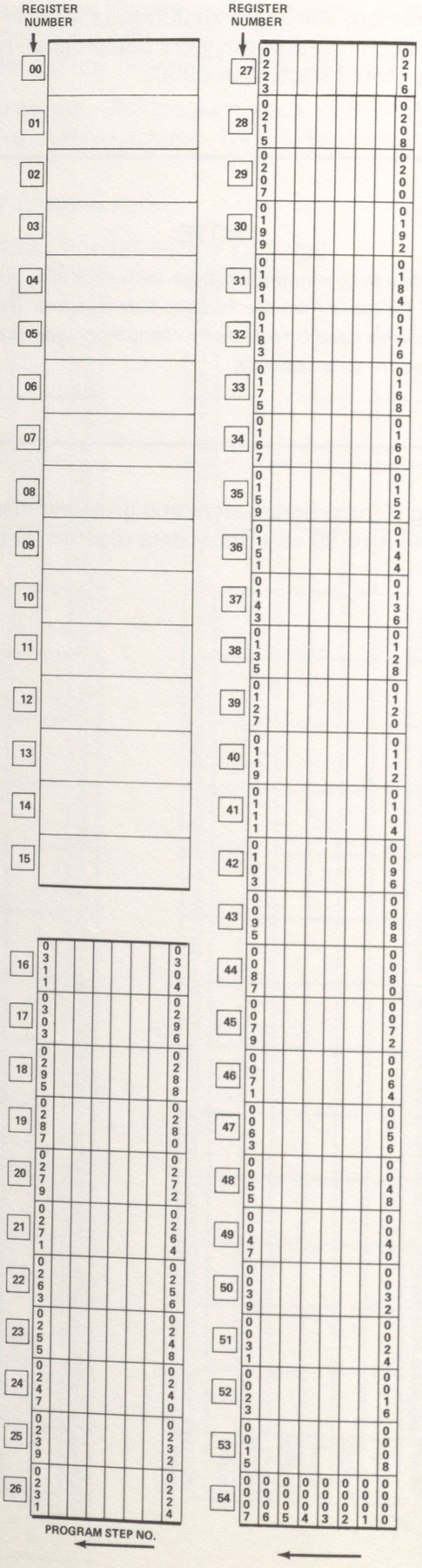
Notice that there are 312 locations starting with 0000 at the lower right-hand corner, and going to 0311 in the upper left-hand corner. Each location can store one program command such as $\underline{\sqrt{x}}$, $\underline{e^x}$, or SIN. The commands are generally stored starting at 0000.

NOTE:

While solving a programming problem with your 600, you may find a need for more than the 16 basic data registers. The 600 can adapt to this need by internally changing program step storage area into data registers.

The 600 does this by taking eight program steps and grouping them together as a data register. The 312-step model has the equivalent of 39 additional data registers, as the diagram on the following page will show.

600-2
312-PROGRAM STEPS/55 DATA REGISTERS



The diagram shows us several other things:

1.) Program step numbering begins at one end of the storage field while data register numbering begins at the other end.

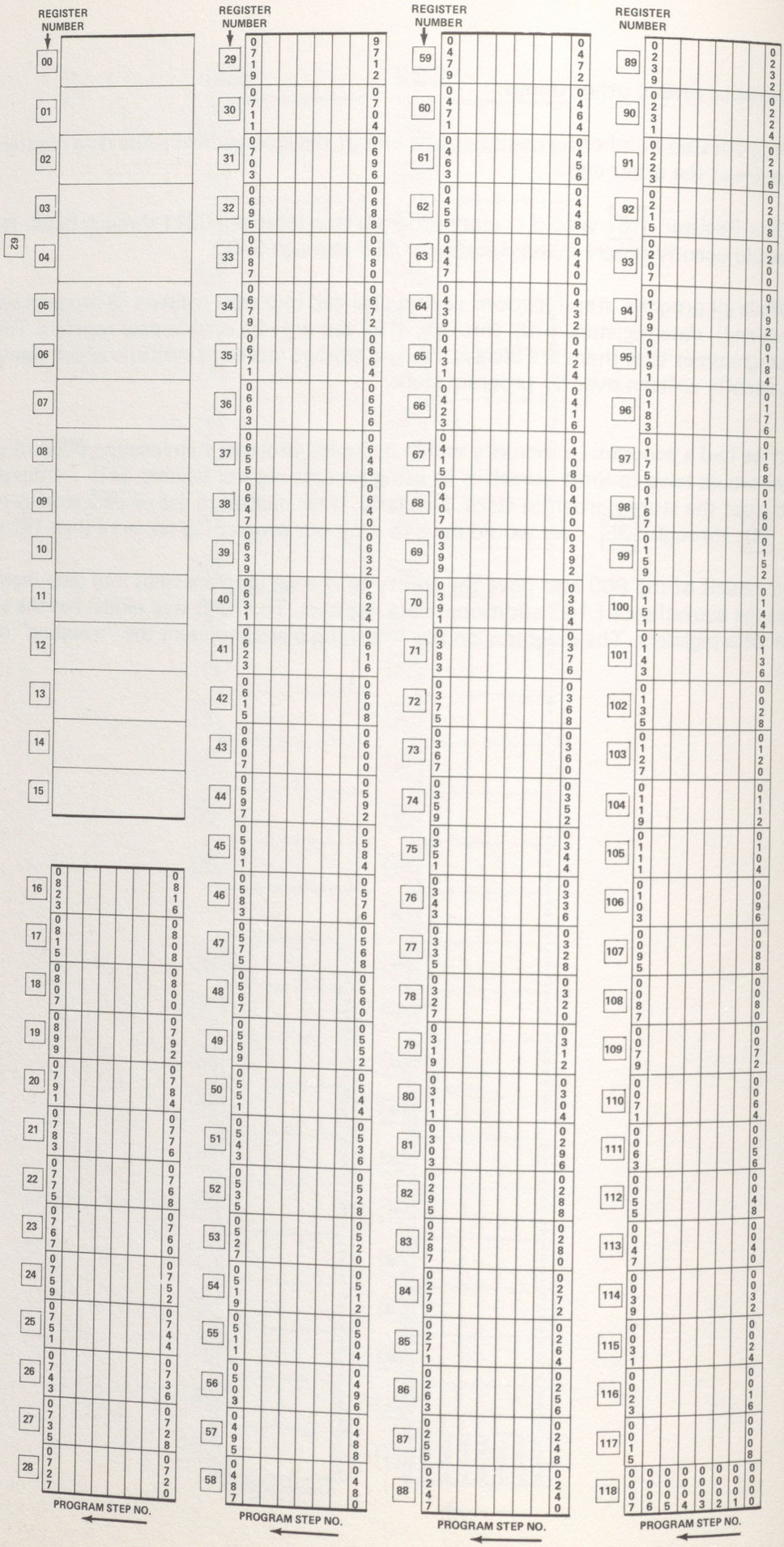
2.) Data Register [16] would occupy program step locations 0311 through 0304. Register [54] would occupy program step locations 0007 through 0000.

3.) While all program step locations can be changed into data registers (8 program steps = 1 data register), data registers [00] through [15] can only be used as data registers. This assures the programmer that there are always 16 data storage locations available which cannot be inadvertently written over by program steps.

4.) If we had a program in memory which occupied program step location 0000 through 0100, we would have to limit ourselves to using no data register beyond [41] . Otherwise, we would disrupt the stored program steps. Similarly, if we needed to use all data storage registers from [00] through [35] , we would have to limit our program to no more than 152 steps.

The other two models of the 600 also have the ability to change program steps into data registers. The 824 step model has the equivalent of 103 additional data registers. The 1848-step model has the equivalent of 231 additional data registers. The diagrams on the following pages represent the "mapping" of each model.

600-6
824-PROGRAM STEPS/119 DATA REGISTERS



600-14
1848-PROGRAM STEPS/247 DATA REGISTERS

REGISTER NUMBER		PROGRAM STEP NO.
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		
16	1847	1840
17	1839	1832
18	1831	1824
19	1823	1816
20	1815	1808
21	1807	1800
22	1799	1792
23	1791	1784
24	1783	1776
25	1775	1768
26	1767	1760
27	1759	1752
28	1751	1744
29	1743	1736
30	1735	1728
31	1727	1720
32	1719	1712
33	1711	1704
34	1703	1696
35	1695	1688
36	1687	1680
37	1679	1672
38	1671	1664
39	1663	1656
40	1655	1648
41	1647	1640
42	1639	1632
43	1631	1624
44	1623	1616
45	1615	1608
46	1607	1600
47	1590	1592
48	1591	1584
49	1583	1576
50	1575	1568
51	1567	1560
52	1559	1552
53	1551	1544
54	1543	1536
55	1535	1528
56	1527	1520
57	1519	1512
58	1511	1504
59	1503	1496
60	1495	1488
61	1487	1480
62	1479	1472
63	1471	1464
64	1463	1456
65	1455	1448
66	1447	1440
67	1439	1432
68	1431	1424
69	1423	1416
70	1415	1408
71	1407	1400
72	1399	1392
73	1391	1384
74	1383	1376
75	1375	1368
76	1367	1360
77	1359	1352
78	1351	1344
79	1343	1336
80	1335	1328
81	1327	1320
82	1319	1312
83	1311	1304
84	1303	1296
85	1295	1288
86	1287	1280
87	1279	1272
88	1271	1264
89	1263	1256
90	1255	1248
91	1247	1240
92	1239	1232
93	1231	1224
94	1223	1216
95	1215	1208
96	1207	1200
97	1199	1192
98	1191	1184
99	1183	1176
100	1175	1168
101	1167	1160
102	1159	1152
103	1191	1144
104	1143	1136
105	1135	1128
106	1127	1120
107	1119	1112
108	1111	1104
109	1103	1096
110	1095	1088
111	1087	1080
112	1079	1072
113	1071	1064
114	1063	1056
115	1055	1048
116	1047	1040
117	1039	1032
118	1031	1024
119	1023	1016
120	1015	1008
121	1007	1000
122	0999	0992
123	0991	0984
124	0983	0976
125	0975	0968
126	0967	0960
127	0959	0952
128	0951	0944
129	0943	0936
130	0935	0928
131	0927	0920
132	0919	0912
133	0911	0904
134	0903	0896
135	0895	0888
136	0887	0880
137	0879	0872
138	0871	0864
139	0863	0856
140	0855	0848
141	0847	0840
142	0839	0832
143	0831	0824
144	0823	0816
145	0815	0808
146	0807	0800
147	0799	0792
148	0791	0784
149	0783	0776
150	0775	0768
151	0767	0760
152	0759	0752
153	0751	0744
154	0743	0736
155	0735	0728
156	0727	0720
157	0719	0712
?58	0711	0704
159	0703	0696
160	0695	0688
161	0687	0680
162	0679	0672
163	0671	0664
164	0663	0656
165	0655	0648
166	0647	0640
167	0639	0632
168	0631	0624
169	0623	0616
170	0615	0608
171	0607	0600
172	0599	0592
173	0591	0584
174	0583	0576
175	0575	0568
176	0567	0560
177	0559	0552
178	0551	0544
179	0543	0536
180	0535	0528
181	0527	0520
182	0519	0512
183	0511	0504
184	0503	0496
185	0495	0488
186	0487	0480
187	0479	0472
188	0471	0464
189	0463	0456
190	0455	0448
191	0447	0440
192	0439	0432
193	0431	0424
194	0423	0416
195	0415	0408
196	0407	0400
197	0399	0392
198	0391	0384
199	0383	0376
200	0375	0368
201	0367	0360
202	0359	0352
203	0351	0344
204	0343	0336
205	0335	0328
206	0327	0320
207	0319	0312
208	0311	0304
209	0303	0296
210	0295	0288
211	0287	0280
212	0279	0272
213	0271	0264
214	0263	0256
215	0255	0248
216	0247	0240
217	0239	0232
218	0231	0224
219	0223	0216
220	0215	0208
221	0207	0200
222	0199	0192
223	0191	0184
224	0183	0176
225	0175	0168
226	0167	0160
227	0159	0152
228	0151	0144
229	0143	0136
230	0135	0128
231	0127	0120
232	0119	0112
233	0111	0104
234	0103	0096
235	0095	0088
236	0087	0080
237	0079	0072
238	0071	0064
239	0063	0056
240	0055	0048
241	0047	0040
242	0039	0032
243	0031	0024
244	0023	0016
245	0015	0008
246	0007	0000

0006 0005 0004 0003 0002 0001

NOTES

Chapter 11

INDIRECT ADDRESSING

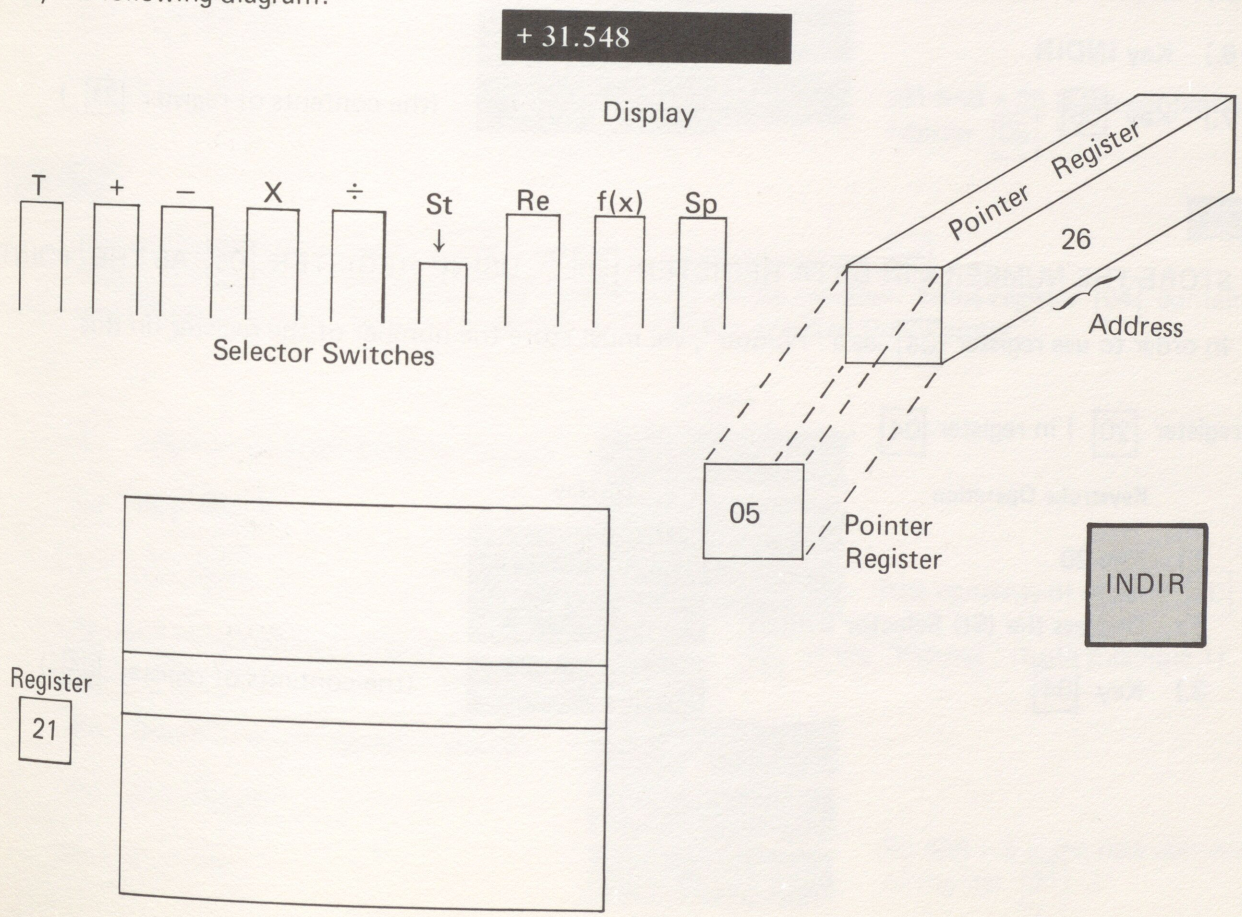
Chapter 10 showed how the 600 can interpret its internal storage area as either program step storage or data registers. Since the procedure for programming has already been explained (chapter 8, remember?), all that is left to explain is how to address the extended data registers. The 600 can be addressed both Directly and Indirectly. This chapter will explain the method focusing on INDIRECT ADDRESSING.

Why Indirect Addressing?

You will find that Indirect Addressing will help you with your programming problems in several ways. Since all data registers can be used with the Selector Switches at the top of the keyboard (the basic 16 registers plus the additional 39, 103, or 231), you can add, subtract, multiply, and divide data while it remains in the register. This is because the data is both addressed and processed by the same two-step command, with INDIRECT ADDRESSING. For an explanation of DIRECT ADDRESSING, see Appendix A.

Using INDIRECT Addressing is quite easy. It involves the use of the INDIR key, the appropriate Selector Switch, and a "Pointer" register, chosen from among one of the basic 16 keyboard registers. The contents of the "Pointer" register are interpreted as the address of a data register.

Study the following diagram:



The diagram represents the procedure for storing the number 31.548 in data register [21] . (Notice that 31.548 is in the window and that the "St" Selector Switch is depressed.) In the previous diagram, if the INDIR key is depressed, followed by keying [05] , here is what will happen:

The 600 will interpret register [05] as the "Pointer." The contents of the window (31.548) will therefore be stored in the register whose address is found in the Pointer. Since the contents of the "Pointer" register are "21", the new contents of register [21] will be 31.548.

Example 1

STORE 31.548 IN REGISTER [21] , USING [05] AS THE "POINTER."

		Display
1.)	Key **21**	+21
2.)	Depress the **(St) Selector Switch**	+21
3.)	Key [05]	+21.00000000 (the contents of register [05])

REGISTER [05] CAN NOW BE USED AS A POINTER FOR INDIRECT STORAGE IN REGISTER [21]

		Display
4.)	Key **31.548**	+31.548
5.)	Depress the **(St) Selector Switch**	+31.548
6.)	Key **INDIR**	+31.54800000
7.)	Key [05]	+31.54800000 (the contents of register [21])

Example 2

STORE THE NUMBER 8 IN DATA REGISTER [20] , USING REGISTER [04] AS THE "POINTER."

In order to use register [04] as a "Pointer", we must store the number of the register (in this case, register [20]) in register [04] .

	Keystroke Operation	Display
1.)	Key **20**	+20
2.)	Depress the **(St) Selector Switch**	+20
3.)	Key [04]	+20.00000000 (the contents of register [04])

NOW, TO INDIRECTLY STORE 8 IN REGISTER [20] , USING REGISTER [04] as THE "POINTER "

Keystroke Operation	Display	
4.) Key **8**	+8	
5.) Depress the **(St) Selector Switch**	+8	
6.) Key **INDIR**	+8.000000000	
7.) Key [04]	+8.000000000	(the contents of register [20]).

The examples given above represent the INDIRECT storage of data. It is just as easy to Indirectly Recall, Add, Subtract, Multiply, and Divide data while it remains in the register.

Example 3

Add 25 to the contents of register [21] .

We can use register [05] from the last example as "Pointer" (remembering that "21" is in register [05]).

Keystroke Operation	Display	
1.) Key 25	+25	
2.) Depress the **(St) Selector Switch**	+25	
3.) Key **INDIR**	+25.00000000	
4.) Key [05]	+56.54800000	(31.548 + 25 is the contents of register [05]).

Example 4

Divide the contents of register [21] by the contents of register [20] .

First, we must recall the contents of register [20] to the window. Since register [04] contains the number 20 (used as the "Pointer" in Example 2), we can use it as the "Pointer."

	Display	
5.) Depress the **(Re) Selector Switch**	+.0000000000	
6.) Key **INDIR**	+.0000000000	
7.) Key [04]	+8.000000000	(the contents of register [20] .)

We can now divide into register [21] , using register [05] as the "Pointer" (from Example 1).

8.) Depress the **(÷) Selector Switch**	+8.000000000	
9.) Key **INDIR**	+8.000000000	
10.) Key [05]	+7.068500000	(56.458 ÷ 8 is the new contents of register [21] .)

As you can see in the previous example, INDIRECT Addressing becomes a two-step command, once a "Pointer" register is established.

Using INDIRECT Addressing Under Program Control

Suppose we had a large set of two-digit numbers (ex. – from a matrix), which we needed to store. We could use a separate register for each number or we could store them two to a register – that is, we could "Pack" the data into the registers.

– How could we write a program to store 20 two-digit numbers in only ten Data Registers?

Suppose two of the numbers are 99 and 25. The possible approach to follow is to place 99 in an adder, multiply 25 by 10^{-7} and add it to 99.

The following results are obtained.

$$\begin{array}{rl} & 99.00000000 \\ + & \underline{\ .00000250} \\ + & 99.00000250 \end{array}$$

We can now move this "Packed" number to a Data Storage Register.

The following example does just this.

Example 5

Write a program which will pack "n" integer data pairs, two per data register, and indirectly store them sequentially, starting in register [16] . Include a one second pause feature which will display each packed data pair in the window prior to storing it.

OPERATING INSTRUCTIONS:

– In RUN Mode –
Key PRIME, key "n" representing the number of data pairs, key GO, key first number of Pair #1, key GO, key second number of Pair #1, key GO, (Packed data pair appears in window for verification.) Key first number of Pair #2, key GO, etc.

Keystroke Operations	Comments
LEARN MODE	
1. Key **MARK**	
2. Key **1**	
3. **(St) Selector Switch Down** 4. Key [05]	Stores "n" in [05] . "n" will be used as a counter.

5. Key 16
6. (St) Selector Switch Down
7. Key [06]

[06] will be used as the "Pointer" register.

8. Key **MARK**
9. Key **2**
10. Key **CLEAR DISPLAY**
11. Key **STOP**
12. Key **STORE RIGHT**

Stores 1st entry of data pair in Right Keyboard Register.

13. Key **CLEAR DISPLAY**
14. Key **STOP**
15. Key **ALPHA** (α)
16. Key **SHIFT**
17. f(x) Switch Down
18. Key [07]
19. Key **PLUS RIGHT**

Multiplies 2nd entry of data pair by 10^{-7} and adds it to the 1st entry of the data pair — data pair is now packed.

20. Key **ALPHA** (α)
21. Key **STOP**
22. Key **ALPHA** (α)
23. Key **STOP**

One second pause displays packed data pair before INDIRECT Storage.

*24. Key **INDIR**
25. (St) Selector Switch Down
*26. Key [06]

INDIRECT Storage of packed data pair in register denoted by the "Pointer" [06] .

27. Key **1**
28. (+) Selector Switch Down
29. Key [06]

Adds 1 to "Pointer."

*INDIRECT Commands

30. Key **1**
31. **(–) Selector Switch Down** — Subtracts 1 from counter.
32. Key [05]
33. Key **J if 0**
34. Key **SEARCH** — Test counter for zero value, which denotes all data pairs have been entered and stored.
35. Key **2**
36. Key **STOP**

***INDIRECT COMMANDS**

Now, how do we write a subroutine to "Unpack" or recall data from the storage registers?

This program would simply undo what the packing program did. The contents of a data register would be moved to the display. Here, the decimal part of the number would be removed by executing the INT X function. The resulting number would be one of the numbers that was originally stored. This number would then be displayed. We would now subtract this number from the original packed number, and multiply this result by 10^{+7} to obtain the second half of the Packed register.

$$.00000250 \times 10^{+7} = 25.00000000$$

Example 6

Write a subroutine which will recall any packed data pair "n" corresponding to the data pairs indirectly stored in Example 5 above.

OPERATING INSTRUCTIONS — IN RUN MODE —

Key PRIME, key "n" (the number of the data pair wanted — i.e., 1 = 1st, 2 = 2nd, etc.), f(x) Switch down, key [01]. (Packed data pair appears in window), key GO, (1st number of data pair "n" appears in window), key GO (2nd number of data pair "n" appears in window).

Keystroke Operations

LEARN MODE

1. Key **MARK**
2. **f(x) SWITCH DOWN**
3. Key [01]

4. **(St) Selector Switch Down**
5. Key [05]
6. Key **15**
7. **(+) Selector Switch Down**
8. Key [05]

Stores the data pair No. "n" in register [05] and adds 15 to it, arriving at the location of the stored data pair. [05] becomes the INDIRECT "Pointer."

* 9. Key **INDIR**
10. **(Re) Selector Switch Down**
*11. Key [05]
12. Key **STORE LEFT**
13. Key **STOP**

Recalls data pair "n" and displays it in the window.

14. Key **Integer**
15. Key **STOP**

Displays the 1st number of data pair "n".

16. Key **MINUS LEFT**
17. Key **ALPHA** (α)
18. **f(x) SWITCH DOWN**
19. Key [07]

Unpacks the 2nd number of data pair "n" and displays it in the window.

20. Key **RETURN**

***INDIRECT Commands**

Chapter 12

AN EXERCISE IN PROGRAMMING

The use of statistical applications in all aspects of industrial research for assisting in formulating scientific judgments needs little elaboration. Wang Laboratories makes available specialized hardware and programs (600 General Library) which encompass the entire spectrum of calculations for statistical applications from averages, least square fits, to three-way analysis of variance, etc.

With this in mind, several statistical examples have been analyzed and rewritten to assist the user in gaining familiarity with the programming features of the Wang Model 600. These examples which follow are written from the point of view of a user attempting to think through the solution by breaking down the total problem into smaller problems. In other words, the intent is to use the old but still useful and logical axiom of Euclid:

The whole is equal to the sum of all its parts.

It seems that Stanley Stanilaus, a marketing analyst for S.A.P. Industries, was assigned the task of programming the newly purchased Wang Model 600 Programmable Calculator.

Possessing a strong math and limited computer background (too far behind him to be helpful), Stan expressed a lack of confidence in mastering the technique for the particular formulas presented him. Of course, Stan was hesitant to reveal his distaste for this new challenge. Each time he looked at the formulas he shuddered.

After a couple of days of keyboard playing, he discovered how really simple it was to use the Wang 600. There was very little Stan couldn't do manually (RUN MODE), with his calculator.

Fully prepared for tackling his programming challenge, Stan looked at the formulas.

$$ {}_nP_r = \frac{n!}{(n-r)!} $$

$$ P(x) = \frac{1}{2} + Z(x) \sum_{n=0}^{\infty} \frac{X^{2n+1}}{1 \cdot 3 \cdot 5 \cdot \cdots \cdot (2n+1)} \quad \text{for } X \geq 0 $$

$$ \text{where } Z(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} $$

Stan decided to rewrite the two problems as four problems. Knowing there may be a need in his work to evaluate only n!, he made this Problem 1. ${}_nP_r$ was Problem 2; Z(x) was Problem 3; and evaluating P(x) was Problem 4.

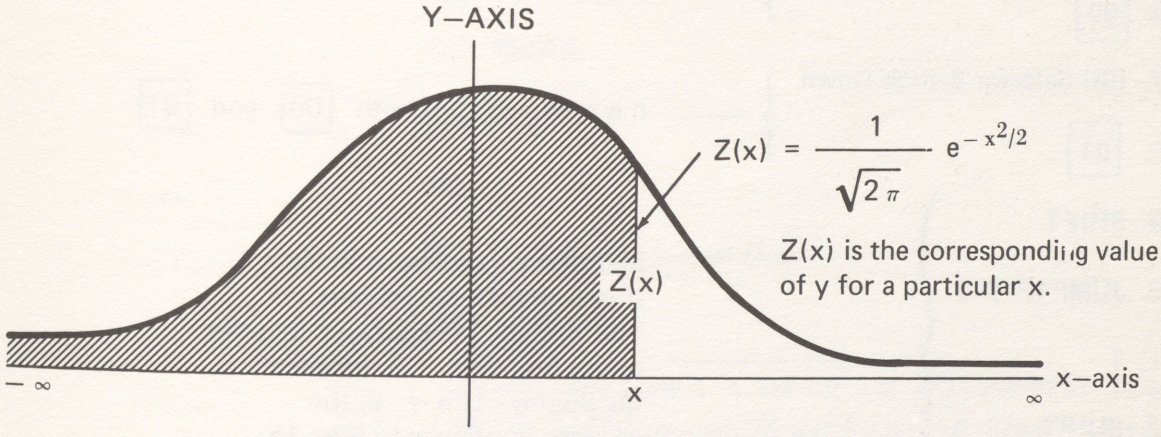
PROBLEM 1 n! find nx(n−1) x (n−2)x. x1

PROBLEM 2 ${}_nP_r = \frac{n!}{(n-r)!}$ Permutation of n things taken r at a time.

PROBLEM 3 $Z(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ Normal Density Function

PROBLEM 4 $P(x) = \frac{1}{2} + Z(x) \sum_{n=0}^{\infty} \frac{X^{2n+1}}{1 \cdot 3 \cdot 5 \cdot \cdots \cdot (2n+1)}$ for $X \geq 0$

Problems 3 and 4 involve evaluating the area under the (bell-shaped) Normal Distribution Curve.



Area under curve from − ∞ to x is expressed by the P(x) formula.

Problem 1 Program n! = n x (n−1)(n-2)(n-3)x x1

> Remember 0! is defined as equal to 1
> This provision has to be included in
> the program.

Stan decided to activate this program by pressing the subroutine key [00] . Working backwards, he wrote the operating instructions as follows:

> OPERATING INSTRUCTIONS for n!
>
> PRIME; f(x) switch down, Key a value for n, Key [00]

Stan knew that for [00] to be used as a subroutine key, the f(x) Switch must be in the DOWN position.

Stan also decided to number his steps starting with 1 and using one step for each keystroke depression, he wrote:

1. **f(x) Switch Down**
2. **SHIFT**
3. **MARK**
4. **00**

} Stan knew (or thought) naming the program by [00] meant n! would result by pressing [00] .

5. **(St) Selector Switch Down**
6. **00**

} n is stored in [00]

7. **(St) Selector Switch Down**
8. **01**

} n is now stored in both [00] and [01]

9. **SHIFT**
10. **JUMP IF ≠ 0**
11. **1**
12. **SHIFT**
13. **RETURN**

} Stan made a provision for 0! = 1
If n=0 step 11 is executed.
The number 1 is displayed and the return key (Step 13) returns to the display. If n ≠ 0, the program continues to Step 15.

14. SHIFT
15. MARK
16. 0
17. 1

This routine allows continuous subtraction of 1 from n for eventual repeated multiplication.

18. (–) Selector Switch Down
19. [00]

1 is subtracted from n. (n–1) is now in [00] and in display and n is in [01] .

20. SHIFT
21. JUMP IF ≠ 0
22. (Re) Selector Switch Down
23. [01]

When n–1 is equal to zero, program recalls n! which is in Register [01] . See Steps 28 and 29.

24. SHIFT
25. RETURN
26. (X) Selector Switch Down
27. [01]
28. SEARCH
29. 0

This routine is necessary to continue the loop. It takes the program back to Step 15.

15. MARK
16. 0
17. 1
18. (–) Selector Switch Down
19. [00]

1 is subtracted from n–1. (m–2) is in [00] and nx(n–1) is in [01] . Steps 20 through 25 test for n–2=0. Since n–2 is not equal to 0, nx(n–1)x(n–2) is stored in [01] . The program stops and displays n! when repeated subtractions yield a 0 in Register [00] (Step 19). Whereupon Step 21 reads a zero, recalls the contents of [01] (n!) and stops (Step 25).

Putting the Model 600 LEARN mode switch down and pressing PRIME, Stan had set the program counter to step 0000. Then he pressed keys according to the keystrokes on the preceding pages. When Stan finished, he scratched his head in a puzzled fashion. He pressed a total of 29 keys and switches, yet the program counter showed only 0017 or 0018 steps. Stanley put the program in again. He observed carefully that the f(x) Switch, SHIFT, (St) Switch, (–) Switch, (Re) Switch, and (X) Switch did not use up program steps.

Pleasantly surprised, Stan rewrote his program with new notation as follows:

STEP	COMMAND	COMMENT
0000	MARK	
0001	00	
0002	(St) 00	(St) Selector Switch Down, Key 00
0003	(St) 01	(St) Selector Switch Down, Key 01
0004	JUMP IF ≠ 0	Test for n = 0 and provision for 0! = 1
0005	1	
0006	RETURN	
0007	MARK	
0008	0	
0009	1	
0010	(–) 00	(–) Selector Switch Down, Key 00
0011	JUMP IF ≠ 0	Test for determining how many times to multiple n x (n–1) x (n–2) x . . .1
0012	(Re) 01	
0013	RETURN	
0014	(Re) 00	(Re) Selector Switch Down, Key 00
0015	(X) 01	(X) Selector Switch Down, Key 01
0016	SEARCH	
0017	0	

To run the program Stan keyed 5 as a value for n, pressed 00 and read the wrong answer, 50. Then, he remembered he had forgotten to put the f(x) switch down. This time he wrote the operating instructions, PRIME, f(x) switch down, key a value for n, key 00 .

Problem 2 Program the formula for a permutation of n things taken r at a time.

$$ {}_nP_r = \frac{n!}{(n-r)!} $$

Stanley decided there was no need to reprogram n! because it was programmed as Problem 1 using subroutine key [00] . He decided to write the program to:

1. Use subroutine key [01]
2. Store n
3. Store r
4. Find n! using [00] from Problem 1
5. Subtract r from n
6. Find (n−r)! using [00] from Problem 1
7. Divide $\frac{n!}{(n-r)!}$

Knowing the n! program used 18 steps Stanley did the following:

PRIME, SET P.C. 0 0 1 8, LEARN MODE

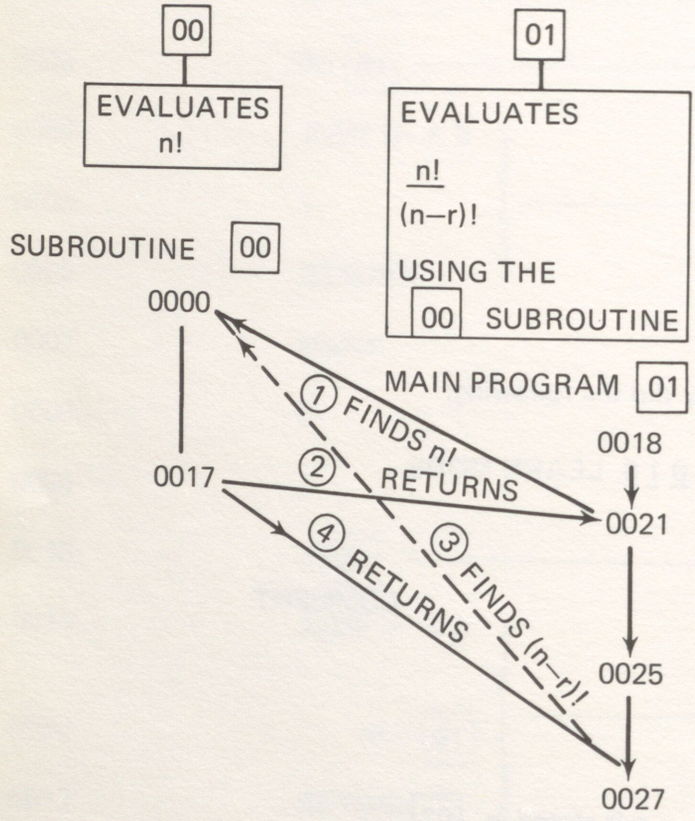
Then he wrote this program:

STEP	COMMAND		COMMENT
0018	MARK		
0019	[01]		
0020	(St) [02]	⟶	n is stored in [02]
0021	f(x) [00]	⟶	n! is evaluated by [00] routine of Problem 1
0022	(St) [03]	⟶	n! is stored in [03]
0023	STOP	⟶	Key r, press GO to continue
0024	(−) [02]	⟶	n−r, is stored in [02]
0025	f(x) [00]	⟶	(n−r)! is evaluated by [00] routine of Problem 1

STEP	COMMAND		COMMENT
0026	÷ 03	⟶	n! already stored in 03 is divided by (n–r)!
0027	Return	⟶	Answer is displayed in window

Stanley wrote the operating instructions as RUN MODE, PRIME, f(x) down, key a value for n, key 01 key a value for r, and key GO. The result ${}_nP_r = \frac{n!}{(n-r)!}$ is read in the display.

Reviewing what he had done, Stanley drew the following diagram:



01 is the main line program. At Step 0021 the subroutine 00 evaluates n! and returns to the main program.

At Step 0025, it is necessary to use subroutine 00 again to evaluate (n–r)!

Problem 3 Evaluate $Z(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$

Looking ahead to Problem 4, Stanley realized the n! routine will have to be used along with the subroutine for Z(x), so he decided to make this program subroutine 02 .

STEP	COMMAND	COMMENT
0028	MARK	
0029	02	
0030	x^2	squares value for x
0031	STORE RIGHT	and stores it in right register
0032	2	
0033	÷ RIGHT	$x^2/2$ is in the right register
0034	CHANGE SIGN	$-x^2/2$ is in the display
0035	e^x	
0036	STORE RIGHT	$e^{-x^2/2}$ is stored in right register
0037	2	
0038	STORE LEFT	
0039	α	These two keys in this sequence always generate π to nine decimal places
0040	00	
0041	X LEFT	2π in left register and in display
0042	$\sqrt{x}$	$\sqrt{2\pi}$ in display
0043	÷ RIGHT	Contents of right register ($e^{-x^2/2}$) is divided by display ($\sqrt{2\pi}$)
0044	RETURN	

Operating Instructions – RUN MODE

PRIME, f(x) switch, Key x,
02. Read Z(x).

Problem 4

Feeling rather good about his accomplishments with the solutions to Problems 1, 2, and 3, Stanley looked quizzically at the formula for Problem 4.

$$P(x) = 1/2 + Z(x) \sum_{n=0}^{\infty} \frac{x^{2n+1}}{1 \cdot 3 \cdot 5. \ldots .(2n+1)} \quad x \geq 0$$

Z(x) is no problem. It is already done as subroutine f(x) [02] . It appeared that evaluating x^{2n+1} was simple enough to do. Just use logs for that. The summation from n = 0 to ∞ is tough. Thinking out loud, Stanley recalled that:

$$\sum_{n=0}^{\infty} \frac{x^{2n+1}}{1 \cdot 3 \cdot 5. \ldots (2n+1)} = \frac{x^{2 \cdot 0+1}}{1} + \frac{x^{2 \cdot 1+1}}{1 \cdot 3} + \frac{x^{2 \cdot 2+1}}{1 \cdot 3 \cdot 5} + \text{etc.}$$

$$\sum_{n=0}^{\infty} \frac{x^{2n+1}}{1 \cdot 3 \cdot 5. \ldots (2n+1)} = \frac{x^1}{1} + \frac{x^3}{1 \cdot 3} + \frac{x^5}{1 \cdot 3 \cdot 5} + \frac{x^7}{1 \cdot 3 \cdot 5 \cdot 7} + \ldots \text{etc.}$$

To stop the summation at ∞, to find n! for just the odd numbers and piece it together appears to be the major difficulty.

Here are some of Stanley's pitfalls and how he overcame them.

PITFALL 1

Stanley thought he could use the n!. subroutine for the denominator of each term in the summation by merely dividing out the even terms. However, he discovered that n! only works to 69. That is, 69! $\cong$ to 10^{+99} . For n values greater than 69 n! is useless. And the summation from n = 0 to ∞ may exceed 69 for certain values of x.

Solution: Recognizing each term in the denominator is multiplied by the constant 2 more than the last multiplicand, Stanley wrote the series,

$$1,\ 1 \times (1+2),\ 1 \times (1+2) \times (1+2+2),\ 1 \times (1+2) \times (1+2+2) \times (1+2+2+2), \ldots$$

$$\updownarrow \qquad \updownarrow \qquad \updownarrow \qquad \updownarrow$$

or $D_o, \quad D_1, \quad D_2, \quad D_3 \ldots$

$$D_o = D_o \times 1$$

$$D_1 = D_o \times 3$$

$$D_2 = D_1 \times 5$$

$$D_3 = D_2 \times 7$$

He knew that a loop which adds 2 to the last multiplicand of each denominator D had to be built into his program.

PITFALL 2

Stanley tried to write a loop for the numerator N of each term and couldn't decide the best way. At first he thought he would use the same loop for the denominator but this tended to complicate his solution for denominators.

Solution: Stanley observed that each numerator after the first could be generated by multiplying by x^2.

$$x^1, \quad x^3, \quad x^5, \quad x^7$$

$$\updownarrow \qquad \updownarrow \qquad \updownarrow \qquad \updownarrow$$

$$N_o \qquad N_1 \qquad N_2 \qquad N_3$$

$$N_o = x$$

$$N_1 = N_o \cdot x^2 = x^3$$

$$N_2 = N_1 \cdot x^2 = x^5$$

$$N_3 = N_2 \cdot x^2 = x^7$$

He decided to simply write a routine for multiplying each numerator by x^2 to get the next numerator.

And putting it all together, Stanley

$$\sum_{n=0}^{\infty} \frac{x^{2n+1}}{1 \times 3 \cdot 5 \ldots (2n+1)} = \frac{x}{1} + \frac{x^3}{1 \cdot 3} + \frac{x^5}{1 \cdot 3 \cdot 5} + \frac{x^7}{1 \cdot 3 \cdot 5 \cdot 7} + \ldots + \ldots$$

$$= \frac{N_o}{D_o} + \frac{N_1}{D_1} + \frac{N_2}{D_2} + \frac{N_3}{D_3} + \ldots + \ldots$$

$$= T_o + T_1 + T_2 + T_3 + \ldots$$

knew that his loop for adding 2 and his routine for multiplying by x^2 would suffice to generate succeeding terms.

PITFALL 3

Stanley still didn't know how or when to stop generating terms. A loop without a decision to stop meant the loop would go on endlessly. Experimenting by calculating the problem to five terms and nine terms to find when the nth term would be negligible resulted in disaster. For x can have any positive value greater than zero.

Solution 3. Stanley finally realized that the Wang Model 600 display only carries ten digits. "That's it", he said, "I'll write the program for ten place accuracy".

There exists a term n such that $\sum_{i=0}^{n} T_i = \sum_{i=0}^{n} T_i + T_{n+1}$ with 10 place accuracy.

In other words T_{n+1} approaches zero. That is, if $\sum_{i=0}^{n} T_i = .1234567890$ and adding T_{n+1} shows the same answer of .1234567890, then for terms beyond T_{n+1} to ∞ will always give us the same answer.

I'll stop the summation and my loop when $\sum_{i=0}^{n} T_i + T_{n+1} = \sum_{i=0}^{n} T_i$.

Stanley wrote this program with the operating instructions PRIME, f(x) Switch down, key a value for x, key [03].

STEP	COMMAND	COMMENT
0045	MARK	
0046	03	
0047	(St) 00	$x = T_o$
0048	(St) 03	$x = T_o$ and x is in the display
0049	f(x) 02	Searches for Z(x) subroutine evaluates Z(x) and stores it in 05
0050	(St) 05	
0051	(Re) 00	recalls x to the display
0052	x^2	
0053	(St) 01	x^2 is the constant multiplier for generating new numerators
0054	1	First denominator for first term is here
0055	(St) 02	
0056	MARK	Routine for generating new terms and deciding when to stop
0057	5	
0058	(Re) 01	Recall constant x^2 from Register 01
0059	(x) 00	Multiplies contents of 00 by x^2 to generate $x^3, x^5, x^7, \ldots$
0060	2	Adds 2 to first denominator 1 and adds 2 to second denominator 3
0061	(+) 02	
0062	(÷) 00	Divides $\frac{x^3}{3}$, $\frac{x^5}{3}$ by 5, $\frac{x^7}{3 \times 5}$ by 7, etc. to find T_1, T_2, T_3, etc., . . .
0063	(St) 04	Each term T_1, T_2, T_3, etc. is stored successively
0064	(+) 03	$\Sigma\ T_n + T_{n+1}$,

STEP	COMMAND	COMMENT
0065	(Re) [04]	
0066	JUMP IF 0	T_{n+1}, if upon recalling T_{n+1}, the ten digit display shows all zeros, we've gone far enough and the program jumps two steps.
0067	SEARCH	
0068	5	
0069	(Re) [03]	Recall Σ
0070	(X) [05]	Multiply Σ by Z(x) which was stored in [05].
0071	•	
0072	5	Add 1/2 to Z(x) Σ
0073	(+) [05]	
0074	Return	Read P(x).

Stanley ran the P(x) program and got the following results:

For x = .1136	P(x) = .5452225563
For x = .5689	P(x) = .7152879973
For x = 2.578	P(x) = .9950313001
For x = 5	P(x) = .9999997132

As a final review of the four problems in this chapter, Stanley attempted to systematize his programming chores by dealing with each separately as a subroutine. Stanley learned:

1. The special top row of switches do not use up a program step nor does the SHIFT key.
2. In writing a program as one of the subroutine keys [00], [01] etc., the f(x) Switch must be down prior to the subroutine key depression.
3. A loop within a program is totally useless without a decision to stop or get out of the loop. A loop should always contain a decision for terminating the loop or directing the program to a new path.

4. Duplication of program steps can be avoided by searching subroutines. This is evidenced by the n! (00) routine which was called up for the ${}_nP_r = \frac{n!}{(n-r)!}$ (01) routine when it was needed.

5. Breaking down a total problem into its components and dealing with each component separately forces the user to think through a problem methodically as a series of simpler, less-complicated problems.

   As an exercise you may want to consider programming the following:

$$\sum_{i=1}^{n} \frac{2i+3}{i\,(i+1)} \cdot \frac{1}{3i}$$

Chapter 13

CASSETTE OPERATING SYSTEM

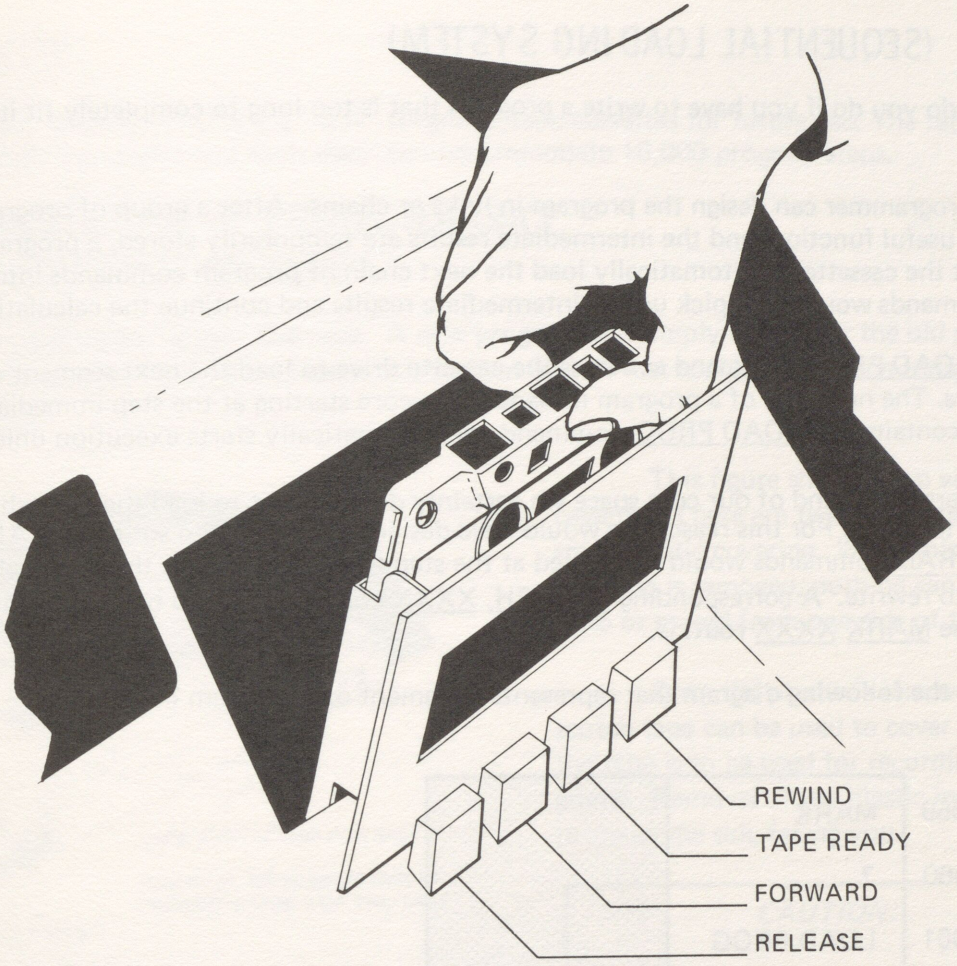
Tape cassette technology has revolutionized home entertainment products. Large bulky turntables and capstan drive tape recorders have been replaced by compact tape cassette players and recorders. Even the television in our living room will soon be affected by this emerging new technology when video tape cassettes will become available.

In addition to revolutionizing home entertainment products, cassettes have had a stimulating effect on the computer industry. Computers now read in data and programs from high speed multiple tape drive cassettes. These cassette systems offer a low-cost alternative to disk and card systems for many users.

This new technology has recently been applied to programmable calculators. Over 10,000 cassette operating calculators have already been installed. It is anticipated that, within a year or two, all programmable calculators will take advantage of this new technology.

The main advantage of a cassette operating calculator is that a calculator is no longer limited in problem solving capacity by the number of program steps that can be stored in the machine at any one time. Programs can be "chained" (on tape) and automatically called in sequence (by the program) into memory when required.

This chapter will not only discuss the technique of "chaining" programs, but it will also discuss how to store and load a set of data registers with a tape cassette, and generally, how to use the tape cassette calculating system.



INSERTING TAPE CASSETTE

There are four switches associated with the tape drive mechanism. These four switches control the manual operating procedures when using the magnetic tape cassette.

1. RELEASE – When engaged, the door swings open allowing the operator to remove or insert a tape cassette.

2. FORWARD – This switch when depressed, will move the tape rapidly in a forward direction.

3. TAPE-READY – This button is depressed when a user is ready to load or record a program or a set of data registers. The operator places the head of the tape reader in contact with the tape.

4. REWIND – To either rewind a tape that has been used or rewind it back to its original position.

Note:

A MACHINE ERROR INDICATOR is located just to the right of the PROGRAM INDICATOR. If data is not transferred properly from or to the tape, the light turns on and the sign of the display flashes. This indicates that the information has not been transferred properly and the operation should be repeated. This flashing light should NOT be confused with the PROGRAM ERROR INDICATOR located just to its left.

CHAINING (SEQUENTIAL LOADING SYSTEM)

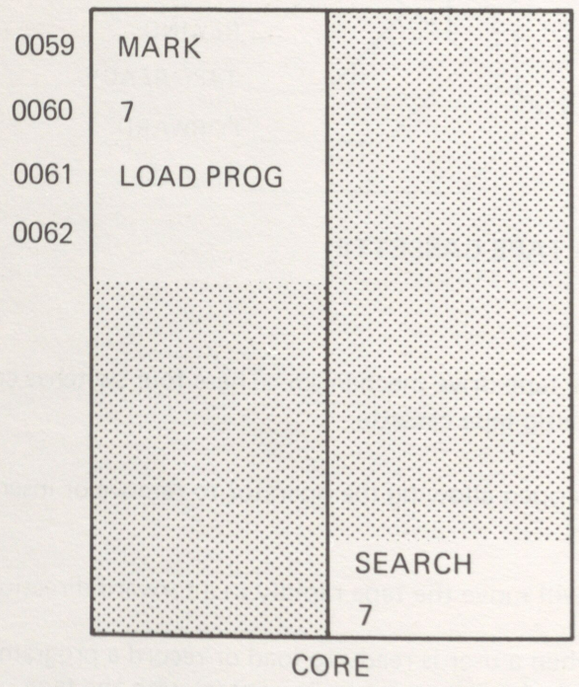
What do you do if you have to write a program that is too long to completely fit it in core at one time?

The programmer can design the program in links or chains. After a group of program commands has served its useful function, and the intermediate results are temporarily stored, a programmed command could instruct the cassette to automatically load the next chain of program commands into core. This group of commands would now pick up the intermediate results and continue the calculations.

The LOAD PROG command activates the cassette drive to load the next segment of a program into core. The next link of a program is loaded into core starting at the step immediately following the step that contains the LOAD PROG command and automatically starts execution unless there is a stop.

If we are at the end of our core space we certainly do not want to load the next chain of program commands at this spot. For this reason we would have designed the program so that a MARK XXXX and LOAD PROGRAM commands would be located at the step that is just before the group of program steps we are going to rewrite. A corresponding SEARCH, XXXX command would be located at the end of the core to flag the MARK XXXX routine.

Study the following diagram that represents a segment of a program in core.



THE SHADED AREA REPRESENTS THE SPACE IN CORE WHERE A PROGRAM SEGMENT IS TO BE WRITTEN OVER BY THE NEXT GROUP OF PROGRAM COMMANDS

When the SEARCH 7 command is finally executed, the system branches to MARK 7. The next block of program steps would be loaded starting at step 0062.

Although the 600 Series Calculator is in itself limited by the number of program steps that can be stored in the machine at one time, the ability to chain programs under program control enables the 600 Calculator to operator like a calculator that has a capacity of 16,000 steps. Many small computers envy this program step capacity.

TAPE CASSETTE

Programs are saved on 4" x 2½" x ½" magnetic tape cassettes for future use. The tape cassette consists of two tracks and each track can accommodate 16,000 program steps.

HOW CAN A PROGRAM BE PROTECTED ONCE IT IS PUT ON TAPE?

There is no need to "erase" the tape. A new program will simply write over the old program. To ensure that a good program stored on tape is not written over or lost accidentally, each track of tape can be protected.



This figure shows a top view of the tape cassette. There are two small openings with a small plastic covering. When this small plastic covering is removed, nothing can be recorded onto or erased from one side of the tape.

Once the plastic is removed, a piece of scotch tape can be used to cover the opening if the tape is to be used for recording other programs. Removal of the plastic insert on the right, protects the side facing you.

CAUTION:

Do not use any cassette with this calculator except the special Wang cassette or an approved facsimile.

The tape used for the calculator must be "certified tape"; that is, a special type of tape which meets certain rigid standards concerning the magnetic coating. Ordinary music tape is not certified tape. Tapes are inserted with the exposed tape up and the side facing the operator is the side of the tape being used. After inserting the cassette properly, close the door by pushing it gently.

HOW TO TRANSFER A PROGRAM FROM MEMORY TO TAPE USING THE RECORD PROGRAM KEY

As you now know, a program can be stored on a magnetic tape for later use. To do this, use the following procedure.

1. Place the 600 in RUN MODE.
2. Insert tape cassette.
3. Rewind tape.
4. Engage TAPE READY switch.

5. If you want to start loading at step 0000, simply depress PRIME. If you want to start loading at a step other than 0000, SET P. C. to the desired step (see SET P.C.); i.e., Step 54 is SET P. C. 0054.

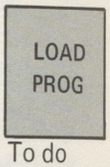
6. Key RECORD PROG. All the steps from the initial setting of the P. C. up to and including the END PROG command will be loaded onto tape. That is, the entire program will be transferred to the tape.

LOAD PROGRAM – Programmable.



The LOAD PROG key transfers a program block from magnetic tape to memory.

The procedure is to SET P. C. and index the LOAD PROG. The first program command located on tape is then stored in the step designated by the P. C. and the program continues loading until it encounters an END PROG instruction. The END PROG is the last step loaded into memory.



HOW TO LOAD A PROGRAM FROM TAPE INTO MEMORY

To use a program stored on tape, you must first transfer the program block into memory. To do this, the following procedure is required:

1. The RUN MODE Switch is depressed.

2. Depress the Rewind key after inserting the tape cassette (This key should be held down until the tape is rewound.)

3. Depress the TAPE READY switch.

4. Depress the LOAD PROG key. When the tape stops turning, the first program block will be in memory. By depressing the LOAD PROG key a second time, the next block would be loaded into memory, and so forth. However, there is an easier way to search for a specific program block on a tape. It involves the use of the ALPHA (α) key together with the LOAD PROG key.

TAPE BLOCK SEARCH AND LOAD – USING ALPHA LOAD PROG

Using the two-step ALPHA LOAD PROG. command, the 600 user can decide which program block on a tape he wants to use, and the 600 will automatically skip over all the intervening blocks, and load the next program block (the one needed) into program memory.

Example 1

Suppose we had a tape containing 7 program blocks and wanted to load Block #5 into program memory. We would do the following:

1.) (Insert tape, REWIND and depress the "TAPE READY" SWITCH.)

2.) Key PRIME. (This sets the Program Step Counter to 0000.)

3.) Key **4** (The number of blocks we need to skip over.)

4.) Key SHIFT, ALPHA (α), and LOAD PROG.

RESULT – The first four program blocks on the tape would be skipped over, and the fifth block loaded into memory.

Example 2

Now, suppose we had a program in memory that we had just written, and wanted to save it on a tape which already had 7 blocks recorded on it. As long as the program does not extend into the last 8 program step memory locations, we can easily record it without affecting the already taped blocks, by doing the following:

1.) Insert the tape cassette, depress REWIND, and then depress the "TAPE READY" SWITCH.

2.) Key SET P. C. to the last location in your program memory (i.e., 0311, 0823, or 1847, depending on the model).

3.) Key **6**.

4.) Key SHIFT, ALPHA (α), and LOAD PROG. The calculator will skip the first 6 tape blocks and attempt to load Tape Block No. 7 into the last 8 program step locations of memory. This will cause the PROGRAM ERROR Indicator to light up, if Tape Block No. 7 is more than 8 steps in size.

5.) Wait until the tape drive stops, and depress PRIME.

6.) Key RECORD PROG.

RESULT – The program in memory will now be Program Tape Block No. 8 on tape.

CAUTION

While the tape drive on the Wang 600 has proven its reliability, it is still subject to some speed variation due to surges in the available power supply (such as caused by an air conditioner starting).

For this reason, assume any previously recorded program blocks following a newly recorded program have been altered, until each is tested and found to be operable. This is especially important when a program is recorded in the middle of a tape, replacing another previously recorded program.

STORING AND LOADING DATA REGISTERS WITH THE TAPE

STORE
RECALL

As we have seen, programs can be easily recorded onto a tape cassette, and loaded back into memory. A logical question at this point is "can we do the same thing with data in data registers?" The answer is a resounding "yes."

Suppose you were solving a problem on your 600 calculator and found that you needed to save the contents of a set of data registers. By using a tape cassette together with the ALPHA STORE and ALPHA RECALL commands, data registers could be stored and re-entered on tape either manually or under program control.

Example 3

Suppose we had data in registers [00] through [26], and needed to use it for several programs. We could key the data into the registers manually each time we needed it, or we could record it on tape:

1.) Insert a tape cassette, REWIND, and depress the TAPE READY Switch.

2.) Key PRIME.

3.) Key 26, the number of the last data register you wish to record. The 600 will record registers [00] through the number keyed (here we have keyed 26) in reverse order; that is, registers [26], [25], [24] .... [00] will be recorded in that order.

4.) Key ALPHA (α), STORE.

RESULT – When the tape stops, we will have stored the contents of registers [00] through [26] on tape. If we wished to RECALL all of these registers, we would REWIND the tape, and use the ALPHA RECALL keys in place of ALPHA STORE. Try it.

– In RUN Mode –

1.) REWIND the tape and depress the TAPE READY Switch.

2.) Key PRIME.

3.) Key 26

4.) Key ALPHA (α), RECALL.

RESULT – The 27 data registers which were stored on the tape will now occupy data registers [00] through [26]. If we want to, we can RECALL the data registers into locations [05] through [31] instead. All we need to do is key in 31 instead of 26, in Step No. 3 above. The 600 automatically loads the recorded registers, beginning in the register whose number appears in the display, and working downward in number. Since, in this example, we have 26 registers recorded as a block, the last register to be filled will be register [05].

NOTE:

The ALPHA STORE can be used to store as many data registers as we wish. However, once stored on tape, the entire block of data registers must be recalled; otherwise, the data will not be entered properly and the Program Error Indicator will light up.

Example 4

Write a program which will RECALL a set of 18 data registers from a tape, take their square root, and record them back on the tape. Use data register [01] as a "Pointer."

NOTE:

Since we are using more than the 16 basic keyboard registers, we will use Indirect Addressing, and a "Pointer" register.

OPERATING INSTRUCTIONS:
(REWIND the tape, depress TAPE READY), key PRIME, SEARCH 2.

Keystroke Operations	Comments
1. Key **MARK**	
2. Key **2**	
3. Key **19**	This will load the 18 recorded registers into Data Registers [02] through [19].
4. Key **ALPHA** (α)	
5. Key **RECALL**	
6. **(St) Selector Switch Down**	
7. Key **18**	
8. Key [00]	
9. **(St) Selector Switch Down**	
10. Key **1**	
11. Key [01]	
12. Key **MARK**	
13. Key **3**	
14. **(Re) Selector Switch Down**	This checks whether we have completed all registers.
15. Key [01]	
16. Key **ALPHA** (α)	
17. Key **J IF +**	
18. Key **SEARCH**	
19. Key **4**	
20. **(+) Selector Switch Down**	Adds 1 to "Pointer" in [01].
21. Key **1**	
22. Key [01]	
23. Key **INDIR**	Uses [01] as "Pointer" for INDIRECT RECALL.
24. **(Re) Selector Switch Down**	
25. Key [01]	

26. Key **SQUARE ROOT** ($\sqrt{x}$)
27. Key **INDIR**
28. **(St) Selector Switch Down**
29. Key [01]
30. Key **SEARCH**
31. Key **3**
32. Key **MARK**
33. Key **4**
34. Key **19**
35. Key **ALPHA** (α)
36. Key **STORE**
37. Key **END**

This will load our data onto the tape, as a block, following the first block.

NOTE:

Data occupies only registers [02] through [19]. [01] will contain the number of the last register used. (i.e., [19] here.)

Chapter 14

PRINTING CAPABILITY (OPTIONAL)

The Wang 600 has an optional output device called a 21-column drum printer which can operate at the rate of 160 lines per minute. The Printer can be instructed directly from the keyboard under manual and automatic control, or it can be programmed to print individual numeric and alpha characters. In addition, a program step trace can be initiated either manually, or under program control.

Two switches and one key are associated with the printing mechanism. They are PAPER FEED, PRINT ON switches and the PRINT key. Before explaining these keys let us divert to some basic instructions on paper and ribbon replacements.

PAPER REPLACEMENT

Replacement paper rolls can be ordered through your local Wang sales representative or directly from Wang headquarters in Tewksbury, Massachusetts. If other types of paper are used, decreased clarity of printout may result. Install the paper roll so that it will unwind from underneath. Depress PAPER FEED pushbutton switch to feed paper around rotating drum.

RIBBON INSTALLATION



PRINT ON



This is a manually operated switch. It must be in the ON position when printed paper tape output, or when paper feed is desired.

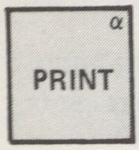
PAPER FEED



The PAPER FEED switch has many functions. It is used for feeding new paper into the machine and can be manually engaged when additional spacing is desired between printed data or it can be written into a program if the need for additional spacing arises,

PRINT
00 15 } CODE FOR LINE FEED

The PAPER FEED switch, whether engaged manually or under program control, will feed at the rate of one line per depression or at the rate of one line per program code.



The PRINT key located on the lower righthand area of the keyboard is the key used to instruct the machine to print. A depression of the PRINT key under manual control will print the contents of the display register. Under List Program and Learn and Print, the PRINT command is automatic and a function symbol will be printed beside each line of data, i.e.,

* LG = $Log_{10}X$

* LN = Natural Log

* SN = SIN

* LP = Load Program

> Note:
>
> *For a list of Key command identifications see Appendix C, Printing Symbol Identification.*

The PRINT key has an alpha sign on the top righthand corner. It is called a double-function key or an Upper and Lowercase Function key. The PRINT function is a lowercase, the ALPHA function is an uppercase.

SAMPLE PRINTOUT

Floating	1 . 0 0 0 0 0 0 0 0 0 0
Scientific	1 . 0 0 0 0 0 0 0 0 0 0 + 00
2 – Decimal Places	1 . 0 0
3 – Decimal Places	1 . 0 0 0
4 – Decimal Places	1 . 0 0 0 0
5 – Decimal Places	1 . 0 0 0 0 0
6 – Decimal Places	1 . 0 0 0 0 0 0
7 – Decimal Places	1 . 0 0 0 0 0 0 0
8 – Decimal Places	1 . 0 0 0 0 0 0 0 0
9 – Decimal Places	1 . 0 0 0 0 0 0 0 0 0

LEARN-PRINT MODE (For printing model only)

The LEARN-PRINT and the LIST PROGRAM modes are used when hard-copy output of the program is desired. In the LEARN-PRINT mode, the function of each indexed key is "learned" into memory and is also listed on paper tape. As each key is depressed, the program step number, the program instruction code and operation symbol are listed. This enables the user to obtain a hard-copy record of his program as it is keyed. This can be extremely useful for program debugging.

LIST-PROGRAM MODE (For printing model only)

When the 600 is in the LIST-PROGRAM mode and the GO key is depressed, it automatically lists the program steps, the instruction codes, and operation symbols in increments of 100 steps, or stops listing when it encounters an END PROGRAM instruction. (If a program is more than 100 steps, depress the GO key after each 100-step increment.) If it is decided to stop the listing at a given point, the STEP key must be depressed. If you wish to resume, key GO. This is useful for program debugging.

AUTOMATIC PRINTING

A 600 Printer Model is equipped with an Automatic Printing feature. What this means is that a user can instruct the machine to print automatically all functional operations being processed. To turn this feature on or off, the following keystrokes are required:

AUTO PRINT "ON"	AUTO PRINT "OFF"
1. SHIFT	1. SHIFT
2. α	2. PRINT (α)
3. PRINT	3. SHIFT
	4. PRINT (α)

MANUAL PRINT OPERATION (Automatic Print OFF)

Keystroke Operation	Display	Actual Printout	
1. Key **5**	+5		
2. Key **STORE RIGHT**	+5.000000000		
3. Key **PRINT**	+5.000000000	5.000000000	ST 14

To print the results, the PRINT key must be depressed.

AUTOMATIC PRINT OPERATION (Automatic Print ON)

Keystroke Operation	Display	Actual Printout	
1. Key **10**	+10		
2. Key **STORE RIGHT**	+10.00000000	10.00000000	ST 14

The decimal point setting can be selected with the SET P.C. key. See Page 151.

SINGLE and DOUBLE PRINT CYCLES

Two different printing operations can occur, single print or double print cycles. To provide a more meaningful printed record, a printer of single cycle operation occurs before addition and subtraction to show the number being added or subtracted rather than the running total.

Function	No. of Printings	Symbol	Print Entry	Print Result	Function	No. of Printings	Symbol	Print Entry	Print Result
PLUS	1	+	YES	NO	INT	2	I	YES	YES
MINUS	1	–	YES	NO	I/X	2	1/X	YES	YES
STORE	1	ST	NO	YES	10^x	2	10^x	YES	YES
RECALL	1	RE	NO	YES	SIN	2	SN	YES	YES
TOTAL	1	T	NO	YES	COS	2	CS	YES	YES
MULTIPLY	2	X	YES	YES	TAN	2	TN	YES	YES
DIVIDE	2	÷	YES	YES	R → D	2	RD	YES	YES
e^x	2	e^x	YES	YES	SIN^{-1}	2	S^{-1}	YES	YES
Log_e	2	LN	YES	YES	COS^{-1}	2	C^{-1}	YES	YES
X^2	2	X^2	YES	YES	TAN^{-1}	2	T^{-1}	YES	YES
$\sqrt{X}$	2	$\sqrt{X}$	YES	YES	D → R	2	DR	YES	YES
|X|	2	|X|	YES	YES	Log_{10}	2	LG	YES	YES

AUTOMATIC PRINTING CYCLES

Example 1

	PRIME	PRINTS		
1.	Key **5**	+5.00000000	* X^2	Print out occurs after $\underline{X^2}$ is keyed
2.	Key X^2	+25.00000000		

Example 2

1.	Key **25**	+25.00000000	* $\sqrt{X}$	Prints out occurs after $\underline{\sqrt{X}}$ is keyed
2.	Key $\sqrt{x}$	+5.000000000		

Example 3

1. Key **25**	+25.00000000	*LN	Print out occurs after $Log_e X$ key depression
2. Key **$Log_e X$**	+3.218875824		

Example 4

1. Key **25.784**	+25.78400000	*I	Print out occurs after INT X key depression
2. SHIFT			
3. Key **Integer (INT X)**	+25.00000000		

Example 5

1. Key **25**	+25.00000000	$*10^x$	Printout occurs after 10^x key depression
2. SHIFT			
3. Key **10^x**	+1.000000000	·25	

Note:

The TOTAL key prints the results and generates a double paper feed.

MANUAL PRINTING (Decimal Point Setting)

If a specific decimal point setting is desired for manual print operation, the unit can be instructed to print in Floating Decimal Point mode, Scientific Notation mode, or up to nine designated places. THE DECIMAL POINT SETTING FOR MANUAL PRINTING ALSO APPLIES FOR AUTOMATIC PRINTING DESCRIBED ABOVE.

To condition the unit to read the desired decimal setting, the SET P. C. key must be depressed, followed by the index of a 4-digit number. In this application, the machine will read the fourth digit that was indexed. The best procedure to follow is to key in three zeros and then the digit that will generate the desired decimal point position.

Fourth digit of the program counter	The decimal setting that will print
0	Floating
1	Scientific Notation
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Example 6 Floating Decimal

Keystroke Operation	Display	Actual Printout
1. Key **123456789**	+123456789	
2. Key **PC 0000**		
3. Key **PRINT**		123456789.0

Example 7 Scientific Notation

Keystroke Operation	Display	Actual Printout
1. Key **123456789**	+123456789	
2. Key **PC 0001**		
3. Key **PRINT**		1.234567890 +08

Example 8 5 Places

Keystroke Operation	Display	Actual Printout
1. Key **123**	+123	
2. Key **PC 0005**		
3. Key **PRINT**		123.00000

Example 9

This example will demonstrate how we can multiply two numbers (5.216 and .145) and under manual control of the keyboard, print the results.

(a) 2 decimal places
(b) 5 decimal places
(c) Floating-Point mode
(d) Scientific Notation

RUN MODE

Keystroke Operation

1. Key **5.216**
2. Key **STORE LEFT**
3. Key **.145**
4. Key **MULT LEFT**

DISPLAY READS +.756320000

	Actual Printout
(a) SET P.C. 0002 PRINT	.75
(b) SET P.C. 0005 PRINT	.75632
(c) SET P.C. 0000 PRINT	.7563200000
(d) SET P.C. 0001 PRINT	7.563200000 –01

PROGRAMMING THE PRINTER

This section deals with the instructions necessary to program the printer to print numeric and alpha characters as part of a program.

Because the Wang 600 optional 21-column drum printer is capable of printing not only the numeric values in the display, but also specific alpha characters, it is called an alpha-numeric printer.

A two-step command is needed to print anything under program control. The first command is PRINT. The second command is a combination command that specifies not only the number of digits after the decimal point that is to be printed out, but also the alpha character that is to be printed out with the result. The two-step operation takes the general form of

STEP 1. PRINT

STEP 2. XX XX,

Specifies an alpha character (The high-order part of the code.)

Specifies the number of digits to be printed after the decimal point (The low-order part of the code).

Let us first explain the technique used to specify the number of digits that are to be printed after the decimal point.

The printed output can have from 0 to 9 digits after the decimal point. What more convenient keys to use to generate the codes for 0 to 9 digits than the first nine Data Register keys, [00] to [09].

00	01	02	03	04	05	06	07	08	09	CODES (LOW ORDER)
[00]	[01]	[02]	[03]	[04]	[05]	[06]	[07]	[08]	[09]	KEYS
0	1	2	3	4	5	6	7	8	9	DIGITS AFTER THE DECIMAL POINT

Note:

Registers [10], [11], *and* [15] *are special codes for*

[10] *Floating Decimal Point*

[11] *Scientific Notation*

[15] *Activating the Paper Feed to skip one line.*

To specify four digits after the decimal point the [04] key would be depressed. The general form of the printout command scheme would now be:

PRINT
XX 04

We still must have a scheme to generate the XX alpha-character codes. How do you think the alpha-characters are generated?

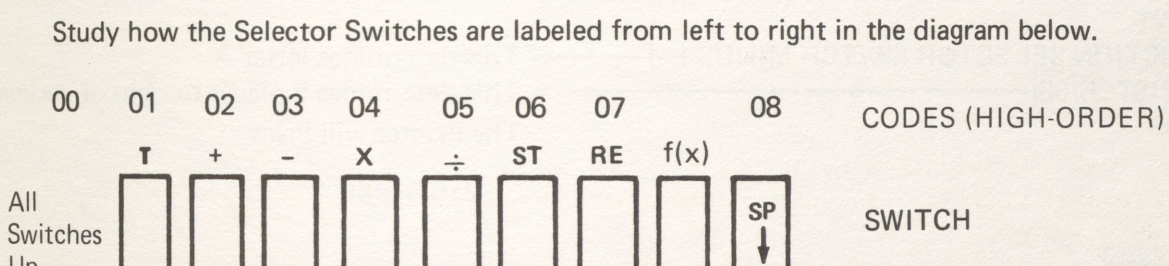
The Selector Switches are used to generate the alpha-character codes.

The program designer has the option to use any of the sixteen available characters and they are given here with their codes.

X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15

As mentioned earlier, these characters are generated by depressing the appropriate Selector Switch, generating the high-order code associated with that switch.

Study how the Selector Switches are labeled from left to right in the diagram below.



By depressing the [+] Switch, the "Z" character would be specified. By depressing the [ST] Switch, the "D" character would be specified. How would you specify the "X" character?

The "X" character would be specified when all switches are in the UP position.

The code for "Z" is 02, the code for "D" is 06 and the code for "X" would be 00.

The generated printout scheme would be:

for Z
PRINT
02 XX

for D
PRINT
06 XX

for X
PRINT
00 XX

NOTE:

If the Printer is OFF, the program will halt if a PRINT command is encountered during execution.

Example 10

This program will calculate the area of a Circle ($A = \pi r^2$) for any value of r. The result will be printed with five place accuracy and will be identified by the letter "A".

LEARN MODE

Keystroke Operation

1. **Mark**
2. **1**
3. **STOP**
4. **X^2**
5. **STORE LEFT**
6. **3.14**
7. **MULT. LEFT**
8. **PRINT**
9. **FUNCTION SELECTOR SWITCH MINUS (–)** → This determines letter A
10. **REGISTER 05** → This determines 5 places to right of decimal.
11. **STOP**

Operating Instructions

RUN MODE

1. Key 8.3
2. SEARCH 1
3. GO

The Printer will Print

216.31460 A

Example 11

This program will calculate the circumference of a circle ($C = 2 \pi r$) for any value of r. The result will be printed with two-place accuracy and will be identified by the letter "C".

LEARN MODE

Keystroke Operation

1. **MARK**
2. **1**
3. **STOP**
4. **STORE LEFT**
5. **3.14**
6. **MULT. LEFT**
7. **2**
8. **MULT. LEFT**
9. **PRINT**
10. **FUNCTION SELECTOR SWITCH DIVIDE (÷)** → This determines letter C
11. **REGISTER 02** → Determines two places to the right of decimal.
12. **STOP**

Operating Instructions

RUN MODE

1. Key 8.3
2. SEARCH 1
3. GO

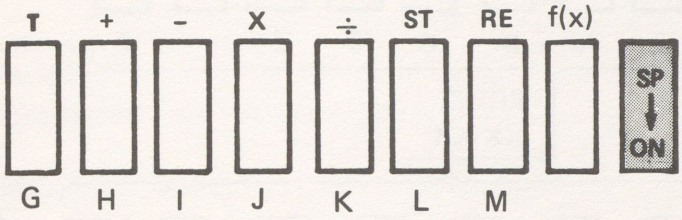
The Printer will Print

52.12 C

So far all we have indicated is how to generate the first nine alpha characters.

Do you have any idea how to generate the additional seven alpha characters using the same Selector Switches?

We would relabel the first seven switches as follows:

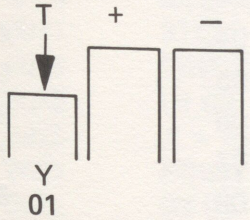


Notice that the switches have been relabeled from left to right with the previously unassigned characters. Also, notice the [SP ON] Switch has been shaded in the above diagram. This is to help us remember that the alpha characters "G, H, I, J, K, L, and M" can be generated by depressing the appropriate character switch and the [SP ON] Switch.

To generate the letter "G" the [T] Switch and the [SP ON] Switch are both depressed. To generate an "H" the [+] Switch and the [SP ON] Switch are both depressed.

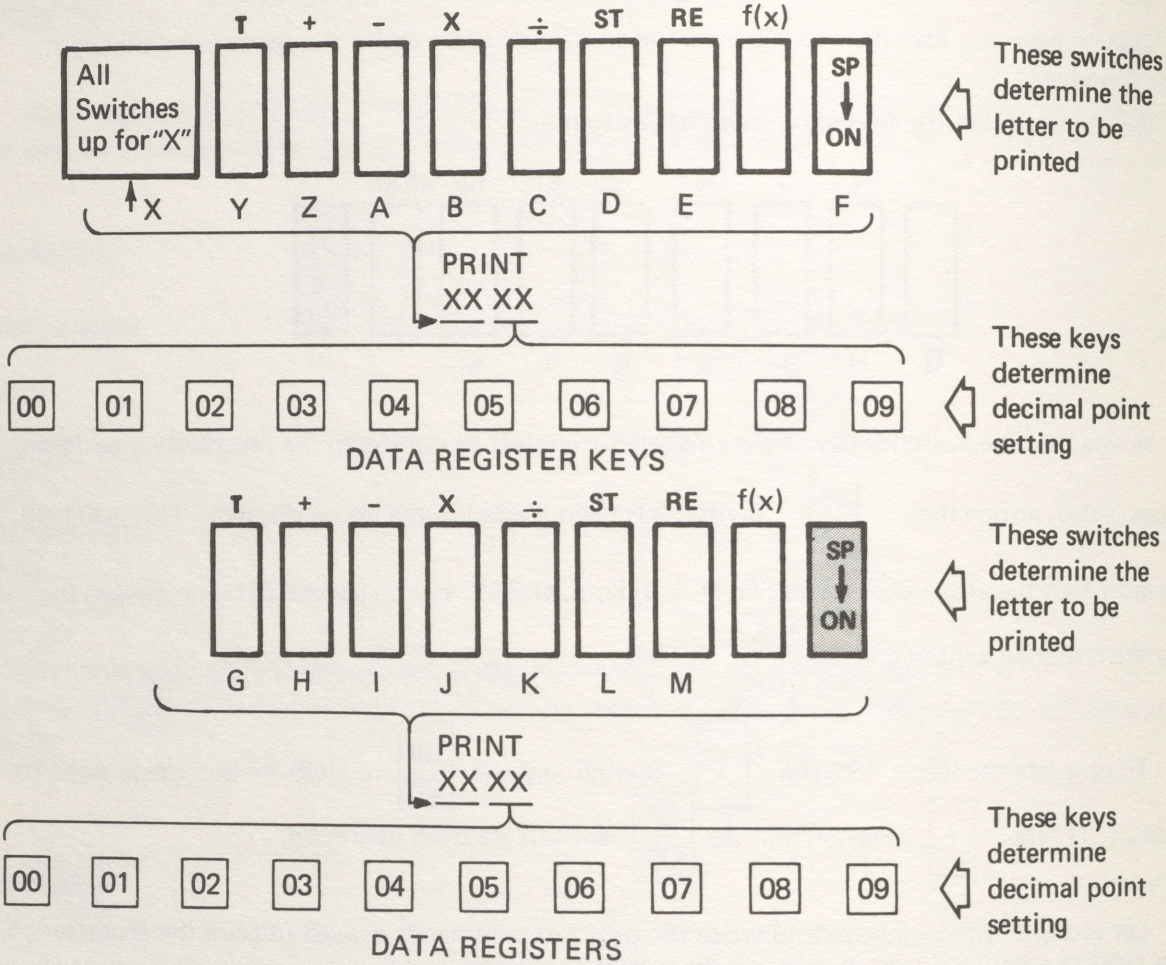
Let us try all the functions and write the printout commands needed to print the character "Y" and the results contained in the display to three decimal places.

To generate the "Y" character, depress the [T] Switch. To specify three decimal places, depress Register Key [03]. It is that simple. The complete operation would be

SELECTOR	STEPS	CODE
	[PRINT]	09 02
	[03]	01 03

The Selector Switches are used to specify the alpha character, the Register keys [00] to [09] are used to specify the number of digits after the decimal point.

A diagram of these relationships is given here.



ALPHA CHARACTER CODES (HIGH ORDER)		
FUNCTION SWITCH	**CODE**	**ALPHA CHAR.**
All switches Up	00	X
Total (down)	01	Y
Plus (down)	02	Z
Minus (down)	03	A
Mult (down)	04	B
Divide (down)	05	C
Store (down)	06	D
Recall (down)	07	E
Special (down)	08	F
Special and Total (down)	09	G
Special and Plus (down)	10	H
Special and Minus (down)	11	I
Special and Mult (down)	12	J
Special and Divide (down)	13	K
Special and Store (down)	14	L
Special and Recall (down)	15	M

DECIMAL SETTING CODES (LOW ORDER)		
REGISTER	**CODE**	**DECIMAL SETTING**
00	00	0
01	01	1
02	02	2
03	03	3
04	04	4
05	05	5
06	06	6
07	07	7
08	08	8
09	09	9
10	10	Floating Point
11	11	Scientific Notation
15	15	Paper Feed

Example 12

Write a program to calculate compound interest. The formula is:

$$A = B(1 + i)^k$$

where K is the number of payments
i is the interest
B is the principle or amount borrowed

Be sure the program prints out A, B, i and K to three places.

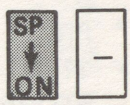
The operating instructions would be:

RUN MODE

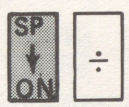
PRIME key i, SEARCH 0, key K, GO, key B, GO.

The program could be:

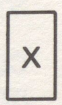
0000	MARK	
0001	0	
0002	PRINT	PRINT i TO 3 PLACES
0003	11 03	
0004	STORE RIGHT	
0005	1	
0006	+ RIGHT	(1 + i)
0007	Log e^X	Log_e (1 + i)
0008	STORE RIGHT	
0009	CLEAR DISPLAY	
0010	STOP	INDEX K
0011	PRINT	PRINT K, 3 PLACES
0012	13 03	
0013	X = RIGHT	K Log_e (1 + i)
0014	e^X	e^X (K Log_e (1 + i))
0015	STORE RIGHT	
0016	CLEAR DISPLAY	
0017	STOP	INDEX B
0018	PRINT	PRINT B, 3 PLACES
0019	04 03	
0020	X = RIGHT	B $(1 + i)^K$
0021	PRINT	PRINT A, 3 PLACES
0022	03 03	
0023	STOP	
0024	END	



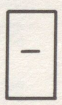
both down
is code 11 for J



both down
is code 13 for K



down
is code 04 for B



down
is code 03 for A

Run the program for i = .06, K = 5 and B = $10,000.

The output would be:

.060	I
5.000	K
10000.000	B
13382.255	A

Study the output. There really isn't any need to print K, the number of payments, to three places. To change the output, change Step 0012 from 13 03 to 13 00. Code 13 00 would be generated by depressing both SP ON & ÷ , and Data Register 00 .

Example 13

Rewrite the program to vary the amount that would be borrowed. Vary B from $10,000 to $20,000 in increments of $1000. Print out I, K, B initial, B final and D, the delta value.

The program would be:

```
MARK
0
PRINT
11 03
STORE RIGHT
1
+ RIGHT
Log eX
STORE RIGHT
CLEAR DISPLAY
STOP                    INDEX K
PRINT
13 03
X = RIGHT
eX
STORE RIGHT
STORE 00
CLEAR DISPLAY
STOP                    INDEX B initial
STORE 01
PRINT
04 03
CLEAR DISPLAY
STOP                    INDEX B final
```

```
06 03
PRINT
00 15          } PAPER
PRINT          } FEED
00 15
MARK
1
RECALL 00
STORE LEFT
RECALL 01
PRINT
04 03
X = LEFT
PRINT
03 03
PRINT

00 15
RECALL 03
+ 01
STORE LEFT
RECALL 02
-LEFT
CHANGE SIGN
```

```
STORE [02]                                  J IF +
PRINT                                       GO
04 03                                       STOP
CLEAR DISPLAY                               SEARCH
STOP                    INDEX INCREMENT     1

STORE [03]                                  STOP
PRINT                                       END
```

RUN MODE

Operating Instructions: Prime, Key i, Search 0, Key k, GO, key B(initial), GO, Key B (final), GO, Key Increment, GO

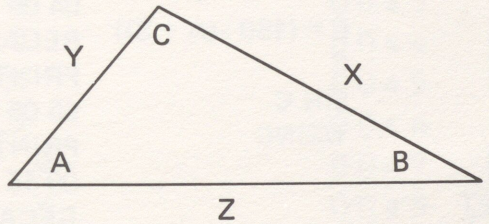
The output of this program follows.

```
     .060   I      12000.000   B      17000.000   B
    5.000   K      16058.706   A      22749.834   A
10000.000   B
20000.000   B      13000.000   B      18000.000   B
 1000.000   D      17396.932   A      24088.060   A

                   14000.000   B      19000.000   B
10000.000   B      18735.158   A      25426.285   A
13382.255   A
                   15000.000   B      20000.000   B
11000.000   B      20073.383   A      26764.511   A
14720.481   A
                   16000.000   B
                   21411.609   A
```

Example 14

Given angles A and B and side Z, calculate angle C and sides X and Y of the following:



From the law of sines we have the following relationships to work with.

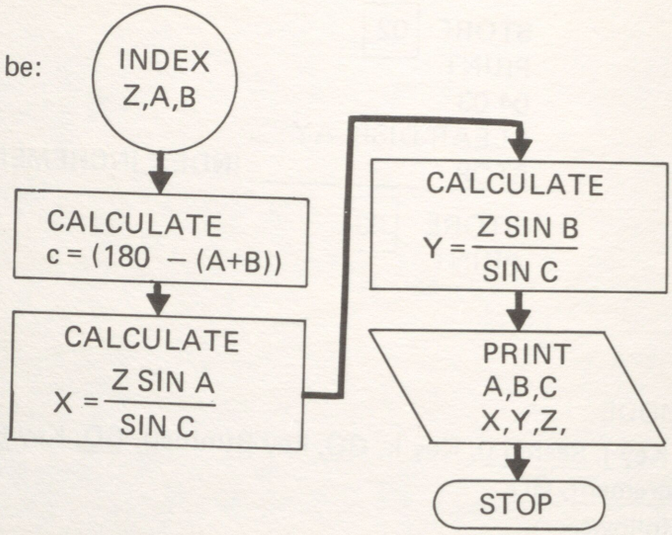
$$C = (180-(A+B))$$

$$X = \frac{Z \sin A}{\sin C}$$

and

$$Y = \frac{Z \sin B}{\sin C}$$

The flow diagram would be:



The operating instructions would be: RUN MODE

Key Z, SEARCH 0, key A, GO, key B, GO.

MARK			
0			
STORE 00	Z		
CLEAR DISPLAY			
STOP			
STORE 01	A		
CLEAR DISPLAY			
STOP			
STORE 02	B		
CHANGE SIGN			
STORE LEFT			
RECALL 01			
−LEFT	−A −B		
1			
8			
0			
+LEFT			
STORE 03	C = (180−(A + B))		
SIN X	C		
1/X	SIN C		
STORE 10	1/SINC		
STORE LEFT			
RECALL 00			
X LEFT	Z		
RECALL 01	SIN C		
SIN X	A		
X LEFT	SIN A		
	X = Z SIN A / SIN C		
		STORE 04	X
		RECALL 10	1 / SINC
		STORE LEFT	
		RECALL 00	
		X LEFT	
		RECALL 02	
		SIN (X)	
		X = LEFT	Y = Z SIN B / SIN C
		STORE 05	Y
		RECALL 01	
		PRINT	}
		03 05	} PRINT A, 5 PLACES
		RECALL 02	
		PRINT	}
		04 05	} PRINT B, 5 PLACES
		RECALL 03	
		PRINT	}
		05 05	} PRINT C, 5 PLACES
		PRINT	}
		00 15	} SPACE
		RECALL 04	
		PRINT	}
		00 05	} PRINT X, 5 PLACES
		RECALL 05	
		PRINT	}
		01 05	} PRINT Y, 5 PLACES
		RECALL 00	
		PRINT	}
		02 05	} PRINT Z, 5 PLACES
		STOP	

Run the program for Z = 5 and A and B = 45°.
The output would be:

```
45.00000    A
45.00000    B
90.00000    C

 3.53553    X
 3.53553    Y
 5.00000    Z
```

Run the program for Z = 5, A and B = 60°.

```
60.00000    A
60.00000    B
60.00000    C

 4.99999    X
 4.99999    Y
 5.00000    Z
```

Notice that this program does not immediately print out the input values. A more efficient approach would be to print out the input values as they are inputted. This program below does that.

STEP	CODE	COMMAND
0000	08 00	* M
0001	00 00	E 0
0002	06 00	ST 0
0003	00 15	E 15
0004	09 03	* SP
0005	06 01	ST 1
0006	00 15	E 15
0007	09 03	* SP
0008	06 02	ST 2
0009	00 12	E 12
0010	06 15	ST 15
0011	07 01	RE 1
0012	03 15	− 15
0013	00 01	E 1
0014	00 08	E 8
0015	00 00	E 0
0016	02 15	+ 15
0017	06 03	ST 3
0018	08 06	* SN
0019	08 15	* 1/x
0020	06 10	ST 10
0021	06 15	ST 15
0022	07 00	RE 0
0023	04 15	× 15
0024	07 01	RE 1
0025	08 06	* SN
0026	04 15	× 15
0027	06 04	ST 4
0028	07 10	RE 10
0029	06 15	ST 15
0030	07 00	RE 0
0031	04 15	× 15
0032	07 02	RE 2
0033	08 06	* SN
0034	04 15	× 15
0035	06 05	ST 5
0036	07 01	RE 1
0037	08 02	* W
0038	03 05	− 5
0039	07 02	RE 2
0040	08 02	* W
0041	04 05	× 5
0042	07 03	RE 3
0043	08 02	* W
0044	05 05	÷ 5
0045	08 02	* W
0046	00 15	E 15
0047	07 04	RE 4
0048	08 02	* W
0049	00 05	E 5
0050	07 05	RE 5
0051	08 02	* W
0052	01 05	T 5
0053	07 00	RE 0
0054	08 02	* W
0055	02 05	+ 5
0056	09 03	* SP
0057	09 14	* EP

Chapter 15

THE PROGRAM STEP TRACE VIA PRINTER CONTROL (AUTO PRINT MODELS ONLY)

If a programmer could open up his calculator and watch his program being executed step by step, he would be able to correct his programming problems in half the time. Since he could see an error when it first occured, most of the problem of fixing a programming error would be eliminated. While this isn't practical with today's high-speed electronic machines, the Wang 600 offers the next best thing – a step-by-step printout of each program operation showing all intermediate results. The program logic becomes clear as well since all MARK SEARCH and Subroutine branchings are shown as they occur in the normal flow of the program. This allows the programmer to make on-the-spot changes and modifications in a program.

The method of initiating a Program Step Trace is by means of a two-step ALPHA command, which programs control of the Column Printer. The two-step ALPHA (α) $\text{Log}_e X$ (or $\text{Log}_{10} X$) signals the TRACE-ON, while ALPHA (α) e^x (or 10^x) turns the Program TRACE-OFF. Output of the Printer includes a printout of the window contents, and the program symbol representing the program step being executed at the time of the printout.

MANUAL CONTROL OF PROGRAM TRACE

The Step Trace feature can be initiated from the keyboard as well as by a program. Suppose we had a program in memory which we wanted to Step Trace through its entire operation. All we would have to do is turn the Trace Mode ON and then key the normal operating instructions: Depress the "PRINTER ON" Switch,

1. Key PRIME.
2. Key ALPHA (α), $\text{Log}_e X$ (or $\text{Log}_{10} X$) – (turns TRACE ON)
3. Key in the data needed by the program (if the normal program instructions require it), and key GO.

R E S U L T S – The Printer will trace the entire program. If a "TRACE OFF" command (ALPHA (α) e^x or ALPHA (α) 10^x) is found within the program however, the Trace will end at that point.

PROGRAMMED CONTROL OF THE PROGRAM TRACE FEATURE

Example 1

To insert a program step TRACE in the INDIRECT Addressing Example 5 (Page 120), we would do the following: (Trace Program Steps 16 through 21).

1. Load the program, and key SET P.C. 0022. (See suggestion on Page 107.)
2. Depress the RUN and LEARN Mode Switches together, and Insert two steps (GO Commands).
3. Replace the inserted GO commands with ALPHA (α) and e^x (or 10^x). This will terminate the Trace after what was originally Program Step No. 21.
4. With the LEARN Switch depressed, key SET P. C. 0016.

5. Depress the RUN and LEARN Mode Switches together, and Insert two steps (GO Commands).

6. Replace the inserted GO commands with ALPHA (α), and $Log_e x$ (or $Log_{10} x$). This two-step command signals the Trace to begin.

Result: If the printer is ON when the program is executed, a printout of the window and the associated program step symbol for each intermediate step will result (i.e., for each data pair entered in the Indirect Addressing example there will be a printout of 8 steps - - 6 program steps plus 2 steps associated with the "TRACE-OFF" command).

The section of the program with the inserted "ALPHA TRACE ON AND OFF" is listed below:

.	.	.	
.	.	.	
.	.	.	
0015	02 14	Plus Right	
0016	09 02	*Alpha (α)	"TRACE ON"
0017	08 10	*$Log_e X$	
0018	15 11	Indirect	
0019	06 06	Store [06]	(Pointer Register)
0020	00 01	1	
0021	02 06	Plus [06]	
0022	00 01	1	
0023	03 05	Minus [05]	
0024	09 02	Alpha (α)	"TRACE OFF"
0025	08 11	e^x	
0026	08 04	Jump if 0	
.	.	.	
.	.	.	
.	.	.	

Note:

If the "TRACE-ON" instruction had been inserted, but the "TRACE-OFF" had not, what do you think would result? – Everything after the "TRACE-ON" command would be traced, right up to the END PROG. Command.

The resulting printout for the three data pairs would be:

(N = 3)

Pair No. 1 = 57, 23

No. 2 = 16, 25

No. 3 = 14, 51

```
57.00000230     D11
57.00000230     ST6
57.00000230     E1
1               +6
17.00000000     E1
1               -5
2.000000000     *   α
2.000000000     *  e^x

16.00000250     D11
16.00000250     ST6
16.00000250     E1
1               +6
18.00000000     E1
1               -5
1.000000000     *   α
1.000000000     *  e^x

14.00000510     D11
14.00000510     ST6
14.00000510     E1
1               +6
19.00000000     E1
1               -5
0.000000000     *   α
0.000000000     *  e^x
```

The printout shows us several things. The first three printout rows represent the packed data pair as it is stored indirectly. Rows four and five represent the incrementing of the pointer register, so that it is ready to indirectly store the next data pair if there is one. Row 6 determines how many data pairs are left, with the result appearing in Rows seven and eight.

As can be seen from Example 1, the Program Trace feature gives the programmer both an insight into the processing of a program, and the ability to check his logic, at the touch of a finger. The benefits are obvious -- problems are easily identified, and their solutions quickly implemented.

NOTE:

If the Printer is not turned on, the Trace will automatically be disabled.

APPENDICES

APPENDICES

Page

APPENDIX A: Direct Addressing 169

APPENDIX B: The 15 07 Subroutine Command 174

A procedure is given for using the new 15 07 code to increase the number of subroutines from 32 to 255.

APPENDIX C: The RUN/LEARN MODE features 177

This appendix contains a concise review of all the non-programmable debugging features.

APPENDIX D: Example of ALPHA RETURN 179

An example of the use of the ALPHA RETURN is given for a subroutine which determines whether a keyed-in number is a prime.

APPENDIX E: The Wang User's ROM 181

APPENDIX F: Printing Symbol Identification 182

Identification of symbols used (i.e., JE means Jump if Error) in printout.

APPENDIX G: Glossary of Keyboard Operations 183

A brief description of the functions performed by keys, and switches.

APPENDIX H: 600 Library of Available Programs 191

Table of Contents of present printed Volumes of Programs. These volumes are constantly being updated and expanded by our professional 600 programming staff.

APPENDIX I: Program Codes . . . . . 194 & 201

A list of machine codes for keystrokes and combinations of keys and switches for various functions.

INDEX 195

600 KEYBOARD FOLDOUT PAGE . . . . . Last Page

APPENDIX A

DIRECT ADDRESSING

Chapter 11 described the procedure for using INDIRECT Addressing with the data registers of the 600 calculator. These same data registers can also be DIRECT Addressed.

As you have found, many of the keys and switches on the 600 Calculator have more than one use. The ALPHA (α) key alone has about 10 different uses, depending upon the key it is paired with in a two-step command. Similarly, the basic 16 data register keys and the Function Selector Switches at the top of the keyboard have so far been used for data manipulation (Chapter 4) subroutines (Chapter 8), and alpha-numeric programming of the column printer (Chapter 14).

By using the Function Selector Switches together with the 16 basic data register keys and the 

key, all the data registers on the Model 600 can be Direct Addressed. Let's look at the High and Low Order Codes from page 151 again:

HIGH ORDER CODES	
FUNCTION SWITCH	CODE
All switches Up	00
Total (down)	01
Plus (down)	02
Minus (down)	03
Mult (down)	04
Divide (down)	05
Store (down)	06
Recall (down)	07
Special (down)	08
Special and Total (down)	09
Special and Plus (down)	10
Special and Minus (down)	11
Special and Mult (down)	12
Special and Divide (down)	13
Special and Store (down)	14
Special and Recall (down)	15

LOW ORDER CODES	
REGISTER	CODE
00	00
01	01
02	02
03	03
04	04
05	05
06	06
07	07
08	08
09	09
10	10
11	11
12	12
13	13
14	14
15	15

These codes are used to program the printer as the second part of a two part command. (Remember that PRINT is the first command.)

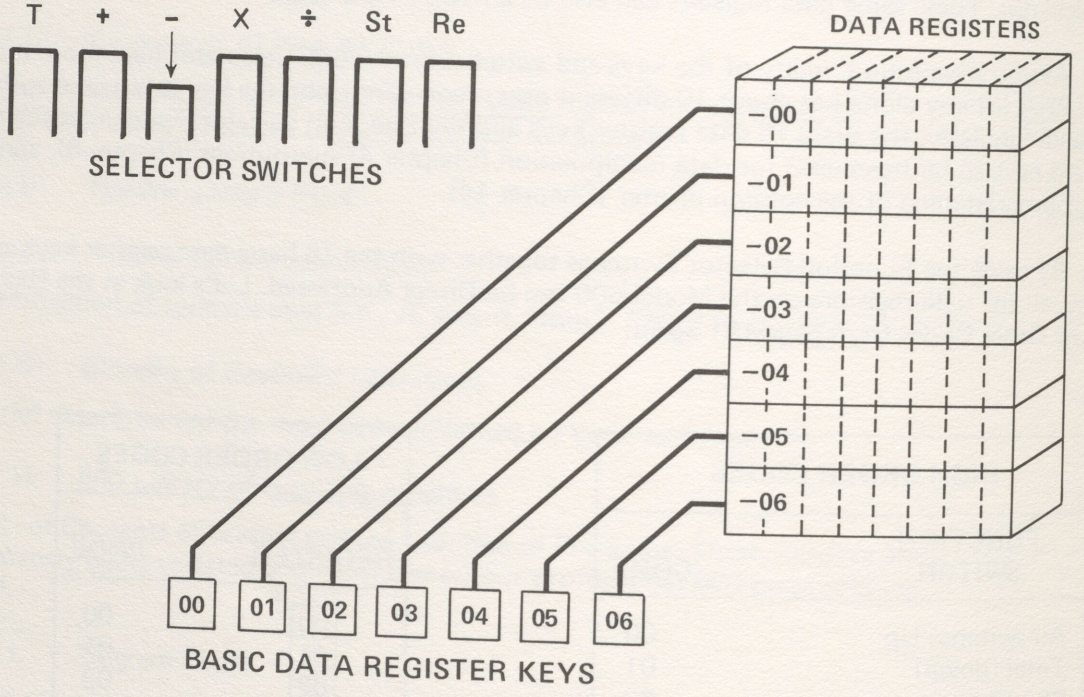
In DIRECT Addressing, these same codes are used as the second part of a two part command. The first command in this case however is either STORE or RECALL.

Study the diagram on the following page, which represents the DIRECT Addressing scheme of the 312 step model.

The diagram shows us several things:

1. The data registers are divided into four groups with each group determined by the Selector Switch which is depressed (i.e., the high order code).

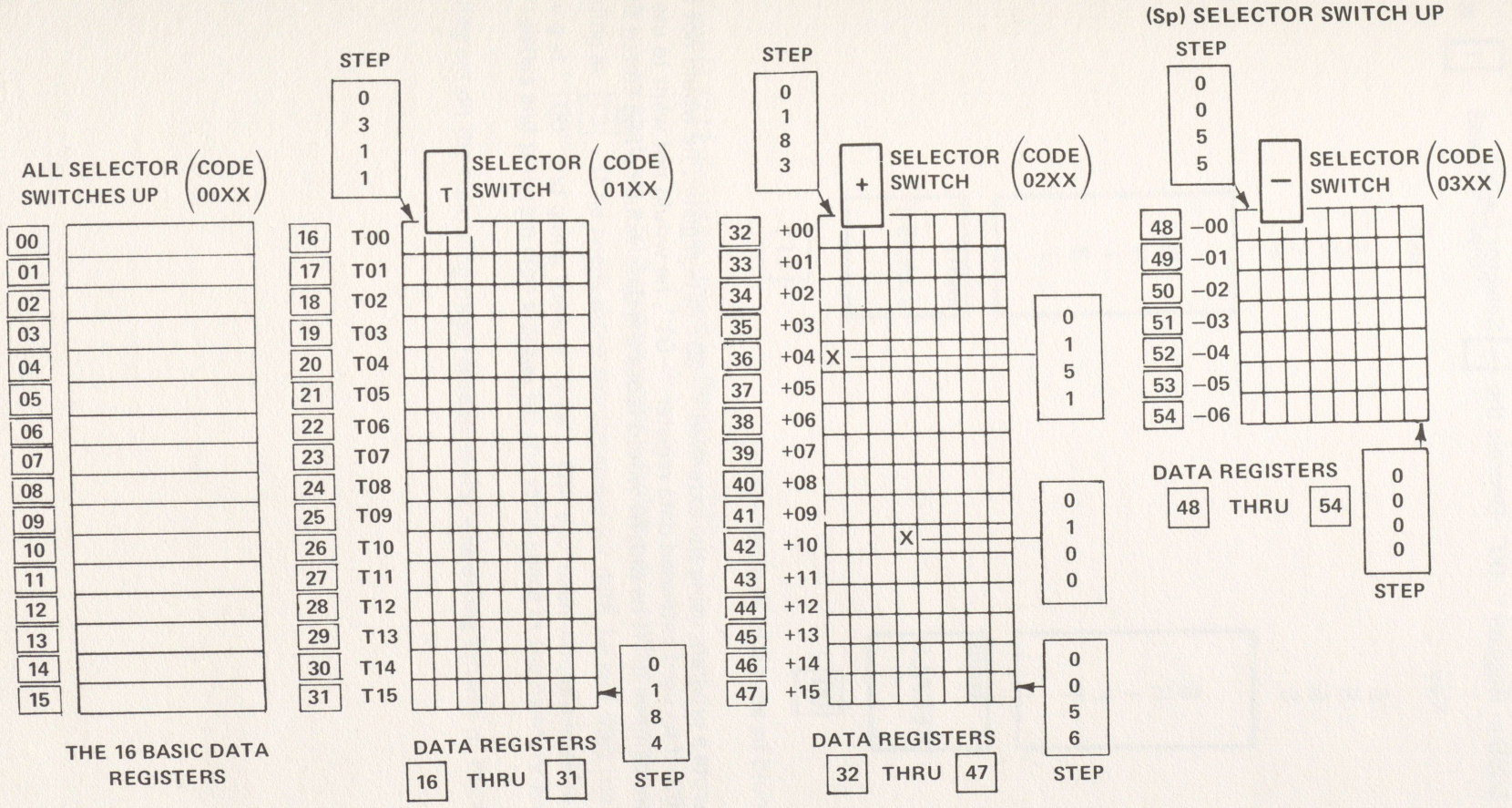
2. Within each set of registers, each individual register is associated with one of the 16 data register keys, as the following diagram also shows:



3. The 312 step model has the equivalent of 39 extra data registers (312 ÷8 = 39), which are divided into three groups — two groups of 16 registers, plus one group of seven registers (2 X 16 + 7 = 39).

4. DIRECT Address register "−06" is the same as INDIRECT register [54], and occupies the space of program steps 0000 thru 0007. Similarly, DIRECT Address register "T00" is the same as INDIRECT register [16], and occupies the space of program steps 0304 thru 0311.

5. As with INDIRECT Addressing, the first 16 data registers are reserved for data only.



311 Program Steps or 39 Additional Data Registers

(54 – 15 = 39)

Example

Store the number 5555 in Register "–00". Depress the [–] Selector Switch

Key

5
5
5
5

[SHIFT]

[STORE RECALL]

[00]

The value 5555 would be stored in Register –00.

If a programmer wrote a program using the extended "–00" data register, he would not use program steps 0056 to 0063. If he used extended data register "–01", he would not want to use 0048 to 0055 for storing program commands. If he did, he would loose the data value he had stored there.

Example

Store 7777 in Register "T12".

Depress [T] Selector Switch

Key

7
7
7
7

[SHIFT]

[STORE RECALL]

[12]

The number 7777 would be stored in Register "T12".

A programmer that has a 312-step capacity calculator would first use the basic data registers starting at [00] and ending at [13] . If he needs more storage space for data, he will then use the converted data registers starting at "T00" and going toward "T15". This approach would minimize any conflict between data storage registers and program step storage for the same space in the memory area.

The diagram on the following page represents the addressing scheme of the three 600 models.

DIRECT ADDRESSING SCHEME FOR THE THREE 600 MODELS

Notice that the data registers are numbered sequentially throughout the three storage options. If your 600 is version 2 as explained above, data storage would extend through the "T", "+", and "–" register blocks. The diagram shows us that data register [54] would be address (03 06). If your 600 is version 6, (824 steps or 103 additional data registers), data storage would extend into the "Re" register block. Data register [118] would be address (07 06). If your 600 is version 14 (1848 steps or 231 additional data registers), data storage would extend into the "Sp Re" register (both the SPECIAL and RECALL Switches down). Data register [246] would be address (15 06). Notice that the (Sp) Switch adds "8" to the value of any other switch depressed.

Note:

Data register [158] should not be Directly Addressed under program control if you intend to Record the program on a tape cassette (do you see why?). The address of data register [158] is (09 14) – the END PROG code. If the (09 14) code is found in a program that is in the process of being recorded, the tape will record only program steps up to this END PROG code.

APPENDIX B

THE 15 07 SUBROUTINE COMMAND

While using the 600's increased programming ability and program step storage area, you may find a need for more than the 32 subroutine codes made available by using the "f(x)" switch, the SHIFT key, and the 16 data register keys. The two step "15 07" (Go Sub) command expands the number of subroutine codes available (from the basic 32) to include every programmable key on the keybaord.

The new "15 07" command is the first part of the two step command. The second step specifies the name of the subroutine you wish to use.

Example:

09 00	MARK
08 15	1/X
00 01	1
06 14	Store Right
00 01	1
02 14	+ Right
09 15	Return

To call the above subroutine, the command sequence would be:

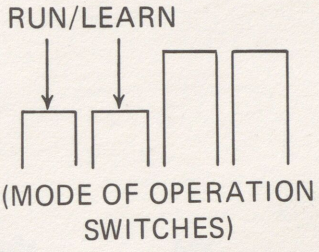
1. Key [07] with the (Sp) and (Re) Switches down (code 15 07)
2. Key 1/X.

RESULT —— The 600 will branch to MARK 1/X, execute the add sequence, and return to the keyboard or program depending on whether the subroutine was requested from the keyboard or by a program.

> **Note:**
>
> *Any keyboard code can be used in place of the 1/X code, including any one of the basic 32 subroutine codes.*

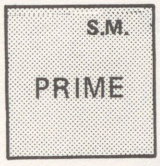
Appendix C

THE RUN/LEARN MODE OPTIONS



A. MARK SEARCH with Program STOP

1. Depress the RUN and LEARN Mode Switches simultaneously.
2. Key S. M. (PRIME).
3. Key in the MARK you wish to Search.



Result: The window will display the first program step following the MARK identified in the SEARCH (i.e., two steps after the MARK Command of 09 00).

B. Backstepping Procedure

1. Depress the RUN and LEARN Mode Switches simultaneously.
2. Depress the B. S. (VERIFY PROG.) key once, for each step you wish to BACKSTEP.



Result: The window will display the steps in descending order.

C. Program Step Deletion

1. Depress the LEARN Mode Switch and locate the step you wish to DELETE (in the window).
2. Depress the RUN and LEARN Mode Switches simultaneously.
3. Depress the DEL (RECORD PROG) key. (To DELETE three steps depress DEL three times.)



Result: The program step in the window will be deleted and the next step will take its place.

D. Program Step Insertion

1. Depress the LEARN Mode Switch and locate the step in front of which you wish to INSERT your steps.

2. Depress the RUN and LEARN Mode Switches simultaneously.

3. Depress the INS (SET P. C.) key once for each step you wish to INSERT. This will INSERT a GO Command (08 03) "spacer" each time INS is keyed.

4. BACKSTEP to the first GO Command (if you are not there already).

5. Key in the new steps you wish to add over the Inserted GO Commands.



Appendix D

EXAMPLE OF ALPHA RETURN (AUTO PRINT ONLY)
Prime No. Determination

Operating Instructions:

Key PRIME, Key "n", f(x) Switch down,
(TURN AUTO PRINT ON)

Key [11] .

Step	Code	Key
0000	09 00	* Mark
0001	10 11	SR1011
0002	06 01	Store in [01]
0003	08 13	* $\sqrt{x}$
0004	09 12	* Integer
0005	06 02	Store in [02]
0006	00 01	1
0007	06 15	Store Left
0008	09 00	* Mark
0009	00 01	1
0010	07 02	Recall [02]
0011	06 00	Store [00]
0012	07 15	Recall Left
0013	09 02	* Alpha (α)
0014	08 05	* Jump If Plus
0015	08 00	* Search
0016	00 03	3
0017	07 01	Recall [01]
0018	06 14	Store Right
0019	00 01	1
0020	02 15	Plus Left
0021	05 14	Divide Right
0022	06 00	Store [00]
0023	09 12	* Integer
0024	09 02	* Alpha (α)
0025	08 04	* Jump If 0
0026	08 00	* Search
0027	00 01	1
0028	00 01	1
0029	09 02	* Alpha (α)

Step	Code	Key
0030	08 04	* Jump If 0
0031	08 00	* Search
0032	00 02	2
0033	09 00	* Mark
0034	00 03	3
0035	07 01	Recall [01]
0036	09 02	* Alpha (α)
0037	09 15	* Return
0038	09 00	* Mark
0039	00 02	2
0040	00 10	Decimal Point
0041	00 05	5
0042	02 01	Plus [01]
0043	09 02	* Alpha (α)
0044	09 15	* Return

Sample output for keyed in data:

n=100	100.5000000	
n=101	101.0000000	= Prime
n=102	102.5000000	
n=103	103.0000000	= Prime
n=104	104.5000000	
n=105	105.5000000	
n=106	106.5000000	
n=107	107.0000000	= Prime
n=108	108.5000000	
n=109	109.0000000	= Prime

APPENDIX E

USER'S ROM*

As you know, the ability to record and load programs in the Model 600 Calculator with the optional tape cassette drive enables you to use any number of previously written programs, at the touch of a key. By inserting a cassette and touching the LOAD PROG Key, the program is automatically loaded into the memory area, ready to use.

But suppose, as is so often the case, there is a particular routine or set of routines which you use time and again, without any variation. Depending on its length and complexity, this set of routines may take up a considerable proportion of the calculator's memory, leaving relatively little room for other programs, which you may change from day to day, but which constantly make use of your standard set of routines.

In a situation such as this, the ideal solution would be to have your very own "routine package" permanently wired into the Calculator, available for your use, just as the $\sqrt{X}$, e^x, or SIN functions are.

The Wang USER'S ROM* is exactly this.

Composed of solid state integrated circuits, each User's ROM can accommodate up to 2048 program steps, and is custom-made to your program specifications at Wang Laboratories. The ROM is set up in cartridge form and plugs into a receptacle on top of the calculator. Both erasable (reusable) and non-erasable circuits are available, depending upon the application.

How the User's ROM works

You can think of a User's ROM as an extension of the Calculator's Memory, since it can be composed of a single program, or a number of subroutines, any of which can be called directly from the keyboard, or from a program in the calculator's regular memory. In addition to the above, however, the ROM can be set up to be completely interactive with the regular memory – that is, the ROM can use calculator storage registers and call a subroutine keyed into the regular memory as well as vice versa. The combinations are practically unlimited.

Accessing the User's ROM

As you would expect, a set of two special commands are required to access the User's ROM. These commands are comparable to the Model 600 Calculator's SEARCH (08 00) and GO SUBROUTINE (15 07) command (see Appendix B):

	User's ROM	600 Calculator
SEARCH	(15 03)	(08 00)
GO SUBROUTINE	(15 12)	(15 07)
SUBROUTINE MARKS	(12 00) thru (12 15) and (13 00) thru (13 15)	(10 00) thru (10 15) and (11 00) thru (11 15)

For further information contact:

SALES SUPPORT
Wang Laboratories, Inc.
836 North Street
Tewksbury, Massachusetts 01876

*ROM stands for "Read Only Memory"

APPENDIX F

PRINTING SYMBOL IDENTIFICATION

– 14	Minus Reg 14 (Right Reg)
T 14	Total Reg 14 (Right Reg)
+ 14	Plus Reg 14 (Right Reg)
÷ 14	Divide Reg 14 (Right Reg)
× 14	Multiply Reg 14 (Right Reg)
ST 14	Store Reg 14 (Right Reg)
RE 14	Recall Reg 14 (Right Reg)
* LP	Load Program
* Jo	Jump if 0
* S	Search
* J+	Jump if Plus
* GO	GO
* RE	Recall
* W	Print
* ST	Store
* α	Alpha
* SP	Stop
* M	Mark
* JE	Jump if Error
* EP	End Program
* Jø	Jump if not 0
E0	Data Entry 0
E1	Data Entry 1
E2	Data Entry 2
E3	Data Entry 3
E4	Data Entry 4
E5	Data Entry 5
E6	Data Entry 6
E7	Data Entry 7
E8	Data Entry 8
E9	Data Entry 9
T0	Total Reg 0
+1	Plus Reg 1
–2	Minus Reg 2
ST 3	Store Reg 3
RE 3	Recall Reg 3
× 3	Multiply Reg 3
÷ 3	Divide Reg 3
f 4	Function of X Reg 4
F4	Shifted Function of X Reg 4
* SN	Sin
* CS	Cos
* RD	Rad to Deg
* TN	Tan
* 1/X	Reciprocal
* LN	$\text{Log}_e X$
* e^x	Anti Log
* X^2	Square
* $\sqrt{X}$	Square Root
* I	Integer
* $\lvert X \rvert$	Absolute Value
* LG	$\text{Log}_{10} X$
* 10^x	10^x
* DR	Deg to Rad
* T^{-1}	Arc Tan
* RT	Return
* S^{-1}	Arcsine
* C^{-1}	Arc Cosine
T 15	Total Reg 15 (Left Reg)
– 15	Minus Reg 15 (Left Reg)
+ 15	Plus Reg 15 (Left Reg)
÷ 15	Divide Reg 15 (Left Reg)
× 15	Multiply Reg 15 (Left Reg)
ST 15	Store Reg 15 (Left Reg)
RE 15	Recall Reg 15 (Left Reg)

APPENDIX G

GLOSSARY OF KEYBOARD OPERATIONS

This appendix contains a brief description of the keys and switches on your Wang Model 600. It provides a general and quick summation of the use of the keys. For in depth explanations refer to the index and Table of Contents.

MODEL 600 KEYBOARD OPERATIONS

> **Note:**
>
> *All operations, unless otherwise stated, may be used in both manual and program control. All trigonometric operations involve an absolute maximum of 572° or less than 10 radians.*

1. S.M. PRIME	2. SIN	3. COS	
4. RAD→ DEG	5. TAN	6. 1/X	
	7. $LOG_e X$	8. e^x	
	9. X^2	10. $\sqrt{X}$	

1. PRIME
   a. In the Learn Mode – sets P.C. at step 0000.
   b. In the Run Mode – replaces data in the display with zeros and sets the program counter at step 0000.
   c. In the Run/Learn Mode – becomes S.M. – Searches for Program MARK specified in window and stops (used only in manual control).
2. SIN
   a. Generates the trigonometric sine of an angle.
   b. When used following ALPHA (α) under program control, skips next two steps if contents of register [00] are numerically less than window contents.
3. COS — Generates the trigonometric cosine of an angle.
4. RAD → Deg. — Converts any radian measure to degree measure.
5. TAN — Generates the trigonometric tangent of an angle.
6. 1/X — Generates the reciprocal of the number in the display.

7. $\underline{\text{Log}_e X}$
   a. Generates the natural logarithm of the number in the display.
   b. Used following ALPHA (α), turns "Program Trace" mode ON.

8. $\underline{e^x}$
   a. Raises e to the power in the display.
   b. Used following ALPHA (α), turns "Program Trace" mode OFF.

9. $\underline{x^2}$ Squares the number in the display.

10. $\underline{\sqrt{x}}$ Generates the square root of the number in the display.

11. TOTAL (left) Transfers data from the left register to the display and simultaneously clears the left register.

12. **— (left)** Subtracts the number in the display from the left register and displays the result.

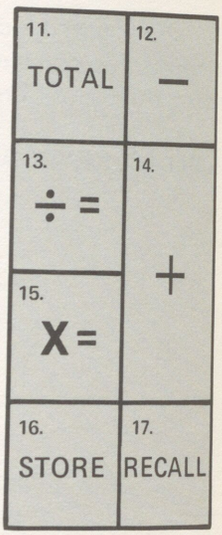
13. **÷ (left)** Divides the number in the display into the contents of the left register and displays the quotient.

14. **+ (left)** Adds the number in the display to the contents of the left register and displays the sum.

15. **X (left)** Multiplies the number in the left register by the display and displays the product.

16. STORE (left) Stores the number in the display in the left register.

17. RECALL (left) Recalls to the display the contents of the left register.

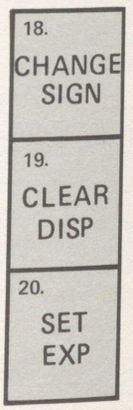


> **Note:**
>
> *The seven previous statements also apply to the operations of the right register.*

18. CHANGE SIGN Changes the sign of the number in the display. When used with SET EXP, this keystroke changes the sign of the exponent.

19. CLEAR DISPLAY Clears the contents of the display.

20. SET EXP After SET EXP is depressed, the next two numeric keystrokes determine the value of the exponent. If the exponent is to be negative, CH SIGN must be depressed additionally.



21. LOAD PROG	22. J if 0	23. I/O	24. INS SET P.C.
25. SEARCH	26. J if +	27. GROUP 1	28. B.S. VERIFY PROG
29. GO	30. RECALL	31. GROUP 2	32. DEL RECORD PROG
	33. PRINT	34. INDIR	35. STEP

21. LOAD PROG
   a. Transfers a program block from tape cassette to 600 memory.
   b. When used following ALPHA (α) skips the number of blocks indicated by the window and loads the following block.

22. J IF 0
   a. Skips the next two program steps if the display contains zero (used only under program control).
   b. When used following ALPHA (α) under program control, skips next two steps if contents of Register [00] are numerically equal to the window contents.

23. I/O
   Used with add-on 600 peripherals, such as the 601 Output Typewriter to format a printout of the contents of the 600 DISPLAY.

24. SET P.C.
   a. Usually used in either LEARN or RUN mode to set the program counter at a desired step in a program. Simply decide the mode to be used, key SET P.C., and then key the desired step number using the numeric keys.
   b. Also used to determine the decimal format for manual and automatic printing. (Cannot be used under Program Control).
   c. In the Run/Learn mode – becomes INS – Inserts a GO Command in front of the program step shown in the window (i.e., allows for program step insertion). –Non-Programmable–.

25. SEARCH
   This keystroke, followed by a code will search and execute the MARK labeled by that code.

26. J IF +
   a. Skips the next two program steps if the sign of the number in the display is positive (used only under program control).
   b. When used following ALPHA (α) under program control, skips next two steps if contents of Register [00] are numerically greater than or equal to the window.

27. GROUP 1
   Used to address add-on GROUP 1 Peripherals.

28. VERIFY PROG.
   a. Decimally adds together the high order and low order digits of the program codes from step 0000 until an END PROG code is encountered. The END PROG code is included in the verify number (used only under manual control).
   b. In the Run/Learn Mode – becomes B.S. Allows for Backstepping sequentially through a program as displayed in window – Non-Programmable –.

29. GO
   Manually indexed to begin the execution of a program, or to resume the execution of a program after a STOP command has been encountered.

30. RECALL
   a. Used to recall any register in memory.
   b. When used following ALPHA (α) will recall a data register block recorded on tape back into data registers, beginning with the register number in the window and proceeding backward to Register [00].

31. GROUP 2 — Used to address add-on GROUP 2 Peripherals.

32. RECORD PROG.
a. Will transfer a program from 600 memory onto a tape cassette starting from where the P.C. is set and ending when an END PROG is reached. This step is executed in either RUN or LEARN mode (used only under manual control).
b. In the Run/Learn Mode – becomes DEL – causes the deletion of the program step displayed in the window, and repacks program into smallest available space.

33. PRINT — Under manual control this keystroke will print the contents of the display. The format will depend upon the program counter. (For further information, see Chapter 14).

34. INDIR — Used in conjunction with the Function Selector Switches, and one of the sixteen basic Data Registers (used as "Pointer") to Indirectly address any Data Register. To Total, Add, Subtract, Multiply, Divide, Recall or Store, a number displayed in the window, in a particular Data Register (either one of the basic 16 or an extended one).
a. Enter the number of the Data Register in one of the basic 16 keyboard registers (thus making that basic register the "Pointer") by using the (St) Switch.
b. Lock in the proper Function Selector Switch.
c. Key INDIR, and
d. Depress the "Pointer" Register key selected in step (a) above.

35. Step
a. In the RUN mode this keystroke will stop an executing program. Also in RUN mode, when a program is in memory but not executing, the STEP keystroke will execute the next program step.
b. In the LEARN mode, the STEP keystroke will advance the program counter one step. (Only used under manual control.)

36. SHIFT — Depression of this key followed by the depression of one of the double function keys will perform the operation of the upper function. (Keying SHIFT once turns SHIFT ON; keying SHIFT again turns it OFF.)

37. SIN^{-1} — Generates the trigonometric arcsine of a number.

38. COS^{-1} — Generates the trigonometric arccosine of a number.

39. DEG → RAD — Converts any degree measure to radian measure.

40. TAN^{-1} — Generates the trigonometric arctangent of a number.

	37. SIN^{-1}	38. COS^{-1}
39. Deg→Rad	40. TAN^{-1}	41. RETURN
36. S H I F T	42. $Log_{10}X$	43. 10^x
	44. Int X	45. |X|

41. RETURN

a. Indicates the end of a subroutine and returns the program to the point from which it branched. (Used only under program control.

b. When used following ALPHA (α) under program control, will execute a subroutine RETURN. A printing of the window will result if RETURN is to the keyboard, and the Auto Print feature has been turned ON.

42. $Log_{10}X$ — Generates the base 10 logarithm of the number in the display.

43. 10^x — Raises 10 to the power in the display.

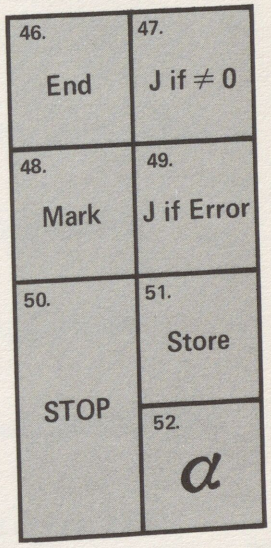
44. INT X — Truncates the decimal part of the number in the display.

45. |X| — Finds the absolute value of the number in the display.

46. END — Used to signal the end of a RECORD PROGRAM or LOAD PROGRAM operation. It is the last instruction to be transferred during either of these operations (used only under program control).

47. J IF ≠ 0 — Skips the next two program steps if the display does not contain zero (used only under program control).

48. MARK — Tells the machine where to start a program, or where to start executing a subroutine or a branch (used only under program control).



49. J IF ERROR — Under program control, this keystroke skips the next two program steps if a program error has been made. Under manual control this keystroke will turn off the PROGRAM ERROR indicator while not affecting the contents of the display.

50. STOP — Used only under program control. Halts the program so that the user may input data.

51. STORE

a. Used to store data in any register in memory.

b. When used following ALPHA (α), will store a set of data registers on tape, beginning at Register 00 and ending at the register number entered in the window.

52. ALPHA (α)

a. Used in turning the Automatic Print ON and OFF.
Procedure: ON = SHIFT, α, PRINT.
OFF = SHIFT, α, SHIFT, α.

b. Used for various operations connected with the f(x) Selector Switch. See f(x).

c. Used for various two-step commands; followed by LOAD PROG, J IF 0, J IF +, STORE, RECALL, $Log_e X$ (or $Log_{10}X$), e^x (or 10^x), RETURN, or SIN.

ALPHANUMERIC COMPARE COMMANDS USING A TWO-STEP I/O (1502) CODE

(Ex.: +3.700000000 + 00 ‡ + 3.700000000-00)

53. I/O 14 00 (I/O JUMP IF EQUAL)

Two-step command used under Program Control only — skips next two program steps if Register [00] is alphanumerically equal to the contents of the window. The (14 00) code is generated by keying [00] with both the (Sp) and (St) Selector Switches down.

54. I/O 15 00 (I/O JUMP IF NOT EQUAL)

Two-step command used under Program Control only — skips next two program steps if Register [00] is alphanumerically not equal to the contents of the window. The (15 00) code is generated by keying [00] with both the (Sp) and (Re) Selector Switches down.

MODES OF OPERATION

55. RUN — Most widely used mode. All manual keyboard calculations are performed in this mode. Programs can only be executed in the RUN mode.

56. LEARN — Used when manually entering a program into memory.

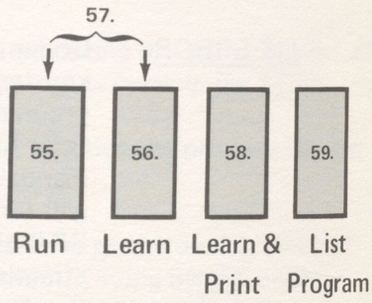
57. RUN/LEARN (Simultaneous)

When used together with PRIME, becomes S.M., MARK SEARCH and STOP.

When used together with SET P.C., becomes INS — Step INSERT Command.

When used together with VERIFY PROG, becomes B.S. — BACKSTEP command.

When used together with RECORD PROG, becomes DEL — Step DELETE Command.



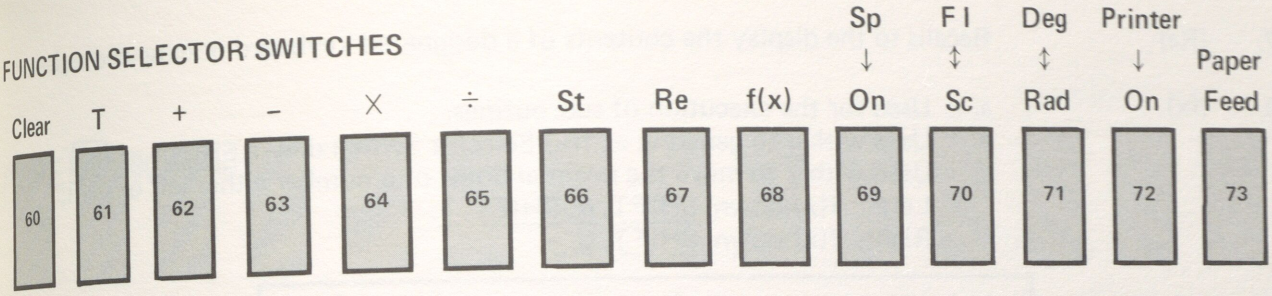
58. LEARN AND PRINT

In this mode, the code of each indexed key is "learned" or entered into memory and also listed with the key's operation symbol as output on paper tape.

59. LIST PROGRAM

When the 600 is in LIST PROGRAM mode and GO is depressed, it automatically lists the program step number, instruction code, and operation symbol, stopping every 100 steps or upon reaching an END PROG command. To continue listing the program after each 100 steps, depress GO. To discontinue a listing, key PRIME. To interrupt a listing, key STEP; to resume, key GO.

FUNCTION SELECTOR SWITCHES



Note:

To apply a function to one of the basic 16 keyboard registers (appearing directly below the function selector switches), simply lock in the function selector switch first and then depress the register being addressed. To apply the same function consecutively to several keybaord registers, simply lock in the selector switch and then depress each register being addressed. To apply a function (displayed in the window) to one of the extended data registers (EDR'S) using the INDIR Key.

1. *Enter the EDR's number in one of the basic 16 keyboard registers (thus making that register the "Pointer") by using the (St) Selector Switch as described below.*
2. *Lock in the appropriate Function Selector Switch.*
3. *Key INDIR, and*
4. *Depress the "Pointer" Register key. The result will appear in the Display.*

60. **(Clear)** Will clear the contents of registers [00] - [15] simultaneously, but will not affect the contents of the display.

61. **(T)** Recalls to the display and clears the contents of a designated keyboard or extended register.

62. **(+)** Adds the number in the display to the contents of a designated keyboard or extended register and displays the sum.

63. **(–)** Subtracts the number in the display from the contents of a designated keyboard or extended register and displays the results.

64. **(X)** Multiplies the contents of a designated keyboard or extended register by the number in the display and displays the product.

65. **(÷)** Divides the number in the display into the contents of a designated keyboard or extended register and displays the quotient.

66. **(St)**

a. Stores the number in the display in a designated keyboard or indirect register.
b. Depression of the (St) switch along with [Sp → On] allows the user to exchange data between the display and a designated keyboard or extended register.

67. (Re) — Recalls to the display the contents of a designated keyboard or extended register.

68. (fx)

a. Used for the execution of subroutines.
b. Used with $\underline{\alpha}$ to generate π: f(x) Selector Switch down, SHIFT, $\underline{\alpha}$, [00] .
c. Used with $\underline{\alpha}$ to move the decimal point of a number either left or right;
Left: f(x) down, SHIFT, $\underline{\alpha}$, SHIFT, $\underline{*}$.
Right: f(x) down, SHIFT, $\underline{\alpha}$, $\underline{*}$.

> **Note:**
>
> *Asterisk represents the keyboard register* [00] - [15] *corresponding to the number of places the decimal point is to be shifted.*

69. (Sp) — Depressing both the (Sp) and the (St) switches allows the user to exchange data between the display and a designated register.

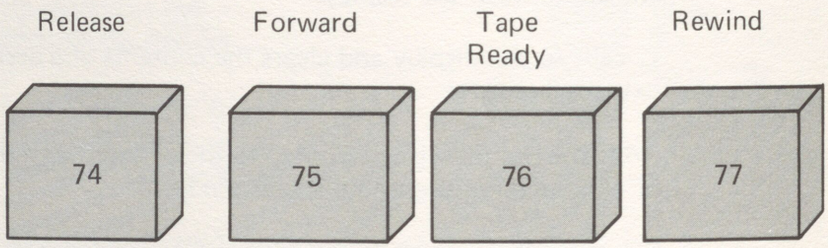
70. [F I ↓ Sc] — On the 600, operations are usually performed in the floating decimal point "mode". To initiate the unit for scientific notation, simply depress this switch.

71. [Deg ↕ Rad] — Determines whether degrees or radians will be used as a measure of the trigonometric functions.

72. [PRINTER ON] — Turns the printer on or off. (Used only under manual control.)

73. [PAPER FEED] — Manually this switch can be used for feeding new paper into the machine or for spacing between data. Can also be programmed for spacing under program control.

TAPE DRIVE OPERATIONS



74. RELEASE — When depressed, the door swings open, allowing for the insertion or removal of a tape cassette.

75. FORWARD — This switch, when depressed, will move the tape rapidly in a forward direction.

76. TAPE READY — Must be depressed in order to execute either the LOAD PROG or RECORD PROG command.

77. REWIND — Used to rewind a tape.

APPENDIX H

GENERAL PROGRAM LIBRARY TABLE OF CONTENTS

SIDE 1

BLOCK NO.	PROGRAM TITLE	PROGRAM NO.
1	LINEAR REGRESSION ANALYSIS	1000-2-ST3
2	NON-LINEAR REGRESSION: LEAST SQUARES PARABOLA	1001-2-ST3
3	NON-LINEAR REGRESSION: LEAST SQUARES POWER CURVE	1002-2-ST3
4	LEAST SQUARES FIT – EXPONENTIAL	1003-2-ST3
5	LEAST SQUARES FIT – EXPONENTIAL (General)	1004-2-ST3
6	LEAST SQUARES FIT – HYPERBOLIC	1005-2-ST3
7	t-STATISTICS, TESTING POPULATION MEAN $\mu = \mu_o$	1006-2-ST2
8	t-STATISTICS, TESTING CORRELATION COEFFICIENTS $\rho = 0$	1007-2-ST2
9	t-STATISTICS, TESTING LINEAR REGRESSION	1008-2-ST2
10	t-STATISTICS, TESTING FOR SIGNIFICANCE DIFFERENCE IN MEAN (Paired)	1009-2-ST2
11	t-STATISTICS TESTING FOR SIGNIFICANCE DIFFERENCE IN MEAN (Unpaired)	1010-2-ST2
12	X^2-STATISTICS, TESTING POPULATION VARIANCE $\sigma^2 = \sigma^2$	1011-2-ST2
13	X^2 - STATISTICS FOR OBSERVED FREQUENCIES AND EXPECTED FREQUENCIES	1012-2-ST2
14	POISSON DISTRIBUTION	1013-2-ST2
15	MEAN, VARIANCE, STANDARD DEVIATION (Ungrouped)	1014-2-ST1
16	MEAN, VARIANCE, STANDARD DEVIATION (Grouped)	1015-2-ST1

TABLE OF CONTENTS (Continued)

SIDE 1

BLOCK NO.	PROGRAM TITLE	PROGRAM NO.
17	RANDOM NUMBER GENERATOR	1016-2-ST6
18	MATRIX ADDITION	1017-2-MA2
19	MATRIX SUBTRACTION	1018-2-MA2
20	MATRIX MULTIPLICATION	1019-2-MA2
21	MATRIX INVERSION	1020-2-MA3
22	SOLUTION OF n SIMULTANEOUS EQUATIONS	1021-2-MA3
23	FOURIER ANALYSIS	1022-2-MA2
24	FORMULAE EVALUATION	1023-6-MA6
25	BASE A $\rightleftarrows$ BASE B	1024-2-MA6
26	SIMPSON'S RULE – f(x) KNOWN	1025-2-MA4
27	SIMPSON'S RULE – f(x) UNKNOWN	1026-2-MA4
28	HYPERBOLIC FUNCTIONS	1027-2-MA1
29	VECTOR ANALYSIS AND OPERATIONS	1028-2-MA3
30	ROOTS OF FUNCTIONS (Half Interval Search)	1029-2-MA2
31	ROOTS OF EQUATIONS	1030-2-MA2
32	GAMMA FUNCTION	1031-2-MA1
33	COMPLEX NUMBER CALCULATOR	1032-2-MA6
34	POLAR TO RECTANGULAR, RECTANGULAR TO POLAR	1033-2-MA1
35	INVOLUTE	1034-2-ME1
36	SPRING DESIGN-COMPRESSION AND EXTENSIONS SPRINGS	1035-2-ME1
37	STRESS AND STRAIN FROM A 45° ROSETTE	1036-2-ME1
38	PLANETARY-EPICYCLIC GEAR RATIOS	1037-2-ME1
39	FOUR GEAR TRAIN	1038-2-ME1

SIDE 2

1	FLOW CALCULATIONS – VENTURI METER	1039-2-CE2
2	CHEZY-MANNING EQUATION	1040-2-CE2
3	RECTANGULAR WEIR	1041-2-CE2
4	FLOW IN PIPE	1042-2-CE2
5	PANHANDLE FORMULA	1043-2-CE2
6	HARDY-CROSS BALANCING OF A PIPE NETWORK	1044-2-CE2

TABLE OF CONTENTS (Continued)

BLOCK NO.	PROGRAM TITLE	PROGRAM NO.
7	CANTILEVER BEAM – INTERMEDIATE LOAD AT A POINT	1045-2-CE3
8	CANTILEVER BEAM – UNIFORM LOAD	1046-2-CE3
9	CANTILEVER BEAM – TRIANGULAR LOAD	1047-2-CE3
10	END SUPPORTED BEAM – CENTER LOAD	1048-2-CE3
11	END SUPPORTED BEAM – UNIFORM LOAD	1049-2-CE3
12	END SUPPORTED BEAM – TRIANGULAR LOAD	1050-2-CE3
13	BOND DOLLAR PRICE AND YIELD	1051-2-BF2
14	PERIODIC PAYMENT	1052-2-BF2
15	ADD-ON INTEREST PAYMENTS	1053-2-BF2
16	COMPOUND INTEREST AND AMOUNT	1054-2-BF2
17	PRESENT VALUE OF A SERIES OF YEARLY CASH FLOWS	1055-2-BF1
18	PERIODIC INVESTMENT	1056-2-BF2
19	SUM FROM INVESTMENT	1057-2-BF2
20	DECLINING BALANCE DEPRECIATION RATE	1058-2-BF1
21	DEPRECIATION CHARGE (Declining Balance)	1059-2-BF1
22	SALVAGE VALUE (Book)	1060-2-BF1
23	DEPRECIATION (Sum of the Digits)	1061-2-BF1
24	AVERAGE LIFE (Annual, Semi-annual)	1062-2-BF2
25	BASIC E.E. DEMO	1063-2-EE2
26	DESIGN E.E. DEMO A	1064-2-EE2
27	DESIGN E.E. DEMO B – BAND PASS FILTER DESIGN	1065-2-EE2
28	DESIGN E.E. DEMO C	1066-2-EE2
29	TCHEBYSHEFF LOW PASS FILTER DESIGN	1067-2-EE2
30	BUTTERWORTH FILTER DESIGN	1068-2-EE2
31	Y-Δ AND Δ-Y TRANSFORMATION	1069-2-EE1

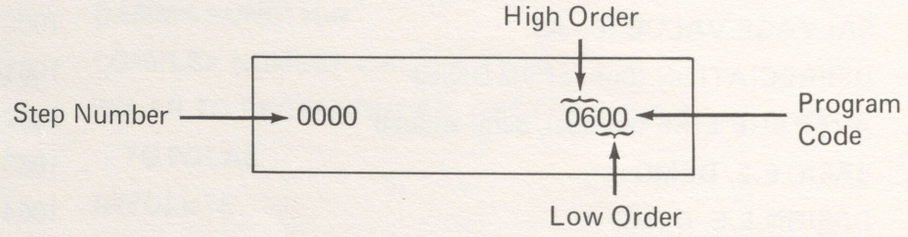
APPENDIX I

PROGRAM CODES

It is important to realize that, unlike other programmable calculators, the Wang 600 has many built-in features that enable the programmer to conserve program steps. For example, the Wang 600 can perform any available operation on any one of registers [00] – [15] using only one program step. This is accomplished by using the special selector switches in conjunction with the keyboard register keys. An example will best illustrate this point.

Example: (1) Depress PRIME and key 11 into the display. (2) Now lock in the (St) Switch and depress Keyboard Register [00] .

We have just manually stored "11" in register [00] using two keystrokes (locking in the (St) switch and depressing [00] . However, if you put the 600 in LEARN mode, follow steps (1) and (2) above and depress PRIME; you will see



0600 is a single program code representing two manual operations.

The first half, or high-order digits, of the code represents the function STORE ("06" represents (St)). The second half, or low order digits, represents the register being addressed. In our example, we see that register [00] is addressed.

Just as 0600 says (St) in Reg. [00],
0610 says (St) in Reg. [10] and
0314 says subtract from Reg. [14]

A list of 600 program codes follows.

Absolute Value |X| , 16

Addition, 7

Addition (+) Special Selector Switch, 23

Alpha Character Codes, 158

Alpha Compare Commands, 78

Alpha Key (π Generation), 25

Alphanumeric Printing Control

154 - 155, 157 - 158

Alpha Return and Print (Subroutine), 93, 94

Example, (See Appendix D)

Angles, by Trigonometric Ratios, 19

Antilog e^x , 13

Appendices, 167 - 194, 201

Arccosine COS^{-1} , 18

Arcsine SIN^{-1} , 18

Arctangent TAN^{-1} , 18

Automatic Printing , 149

Automatic Printing Cycles, 150, 151

Backstep B.S. , 99

Base 10 Log $LOG_{10}X$, 17

Birthday Problem, 96

Block Diagram, 48 - 53 (Chapter 6)

Branching, 63, 88

Cassette Operating System (Chapter 13), 138 - 146

Chaining Programs, 138 - 140

CHANGE SIGN , 3, 4

Clear Switch, 1, 21

CLEAR DISPLAY , 3, 4

Codes for Addressing Subroutines, 85

(1507) GO SUB Command

(See Appendix B)

Conversion of Program Step Storage to

Data Storage Registers, 109 - 111

Core Overload Prevention, 140

Cosine COS , 14, 15

Counter, 73

Data Entry Key, Right and Left Registers

(Chapter 2), 3 - 11

Data Register Manipulation with Tape

(Alpha Store, Alpha Recall)

(See Appendix G), 143, 144

Data Storage, (Chapter 10) 109 - 115

Debugging (Chapter 9), 97 - 108

Backstep, 99

Deleting Steps in Program, 103

Inserting Steps in Program, 105

Machine Error Indicator, 101

Mark Search and Stop, 102

Program Error Indicator, 100

Set P. C., 98

Step Key, 98

Verify Program, 101

Decimal Point Setting for Printing, 151

Decimal Setting Codes, 158

Decision (Concept), 67

Decision-Making Commands, 68

Decrementing Routine, 67, 74 - 75

Degrees-to-Radians [DEG-RAD], 17

Degrees-to-Radians Switch, 14

Delete [DEL] (Run/Learn Mode Only) 103

Direct Addressing, (See Appendix A)

Division [÷] Left and Right, 8

Division, (÷), Special Selector Switch, 24

Drum Printer (21 Column), 147, 154

[END] Program, 59

ERROR Condition Lights, 71

f(00) to f(15), 85, 87 - 88

f(x) Special Selector Switch, 25

Use with [SHIFT] for Subroutine Code, 85

Floating Point Mode, 8, 20, 151 - 154

Floating Point Model, Scientific Notation, 8 - 10

Flow Diagrams (Chapter 6), 48 - 53

relation to computer programming

(Chapter 7), 54 - 65

Forward Switch (Tape Control), 139

Function Printing, 150

Function Selector Keys (Chapter 4), 20 - 27

Special Selector Switches (See Appendix G)

Glossary, (See Appendix G)

[GROUP 1], [GROUP 2], (See Appendix G), 2

Indirect Addressing, [INDIR], (Chapter 11), 117 - 123

Input X, Keying into Display, 58

[I/O], (See Appendix G), 2

Alphanumeric Compares, (See Appendix G)

Insert Step [INS] (Run/Learn Mode Only) 105 - 108

Inserting Tape Cassette, 139

Integer [INT X], 12, 16

Jump Compare Commands, 79

Alpha - J if 0

Alpha - J if +

Alpha - SIN

Alphanumeric Compares, (See Appendix G)

Jump if Equal

Jump if Not Equal

Jump if Plus [J IF +], 68, 76

[JUMP IF ERROR], 68, 71

Jump if Not Zero [J IF ≠ 0], 68, 71

Jump if Zero [J IF = 0], 68, 70

Keyboard Operations, Glossary, (See Appendix G)

Learn Mode/Run Mode Switches, 47, 60, 97 - 98

Learn-Print Mode Switch, 149

Library of Available Programs for 600,

(See Appendix H)

List Program Mode, 149

LOAD PROG , 142, 143

Loading Program from Tape into Memory, 143

Loops, 66

Example: 66, 67, 70, 73 - 78

Machine Error Indicator, 1, 101, 139

Manual Operation, Preparing Machine,

(Chapter 1), 1, 2

Manual Print Operation, 149, 151

Manual Printing, 151 - 153

MARK , 56 - 63, 85, 140

Mark Search and Stop S.M.

(Run/Learn Mode Only), 102 - 103

Minus (–), Special Selector Switch,

used during Storage Conversion, 109

Mode of Operation, Run, Learn, Run/Learn,

Learn - Print, List Program, 2

Models of 600, 99, 110

Multiplication X = Left and Right, 8

Multiplication (X), Special Selector Switch, 23

Natural Log $LOG_e X$, 13

Negative Numbers, Reiterative Adding Program, 69

Non-Programmable Debugging Features

(See Appendix C), (Chapter 9), 97 - 108

- Mark Search and Stop (S.M.)
- Backstep (B. S.)
- Delete Step (DEL)
- Insert Step (INS)
- Prime
- Set P. C.
- Verify Program
- Step

Packing and Unpacking Data Registers, 120 - 123

Paper Feed Switch, 147, 148

Paper Replacement, 147, 148

Pause, Half Second 67, 69 - 70

Pi (π) Generation - Special, 25

Plus + Right and Left, 7

Plus (+) Special Selector Switch, 24

Pointer Register (Chapter 11), 117 - 123

PRIME , 1, 2, 60, 100, 101

Prime Number Determination

(Example), 93

PRINT , 6, 147 - 148

Printer/On Switch, 147, 148

Printing Capability (Optional),

(Chapter 14), 147 - 163

Printing Symbol Identifications,

(See Appendix F)

Processing Operation, 56 - 59

Program Blocks, 99

Program Codes, (See Appendix I)

Program Control of Column Printer, 154 - 155

Program Count, 98

Counter Setting (Set P.C.), 60

Program Error Indicator, 1, 100

Program with Subroutine, 90

Program Step and Data Storage
(Chapter 10), 109 - 115
Also See Appendix A
Program Step Trace, (Chapter 15), 164 - 166
Programming
Approach for Infinite Sum - 10-Place Accuracy
134 - 136
Concepts, Loops, Decisions, Subroutines,
(Chapter 8), 66 - 96
Exercise in Statistics
(Chapter 12), 125 - 137
Into Memory via Keyboard, 60
Introduction To, 46 - 47
The Printer, 154 - 163
Protection of Taped Programs, 141
Pythagorian Theorem (Debugging Example), 98
Example, 28
RAD-DEG Conversion Switch, 14
Radians-to-Degrees [RAD-DEG], 14
Range of Exponents, 3
[RECALL] - (See Appendix A) –
Direct Addressing
Recalling Data Registers With Tape – 143-144
[RECALL], Left and Right, 5, 6
Recall (RE) - Special Selector Switch, 11, 22
Reciprocal [1/X], 14
Register Clearing, 1, 21
Register Keys, (Chapter 4), 20 - 27
(Chapter 10) 109 - 116, (Chapter 11) 117 - 123
Registers, Right and Left, (Chapter 2), 3 - 11
Release Switch (Tape Control), 139
Repetitive Multiplication (Example), 35, 36
[RETURN], 87
Rewind Switch (Tape Control), 139
ROM, (Read-Only-Memory) User Definable
(See Appendix E)
Run Mode, 2 47
Run/Learn Mode, 2, 97
Options (Summary) (See Appendix C)
Scientific and Engineering Keys
(Chapter 3), 12 - 19
Scientific Mode Switch, 8 - 11
Scratch Registers, 59, 109 - 110
[SEARCH], 56 - 63, 69
Set Exponent [SET EXP], 3
[SET P. C.], 60, 142, 143
[SHIFT], 16, 26, 27
Shifting Decimal Point, 26, 27
Sine [SIN], 14, 15
Single and Double Print Cycles, 150-151
Special Selector Keys and Registers,
(Chapter 4), 20 - 27

Special Switch, 1, 21

Exchanging Data (Sp), Special Selector Switch, 25, 157 - 163

Square X^2, 12

Square Root $\sqrt{x}$, 12, 13

Statistics, Programming Exercise, (Chapter 12), 125 - 137

STEP, 75, 98

STORE (See Appendix A) –

Direct Addressing

Storing Data Registers With Tape – 143-144

Store (St) Special Selector Switch, 10, 21

STORE, Left and Right, 4, 5

Subroutines, 25, 83 - 92

Keys, 85

Subroutine Command (1507), (See Appendix B)

Subtraction –, Right and Left, 7

Subtraction (–), Special Selector Switch, 24

Subroutine Levels, 88, 89

Subroutine/Sub Program, 83 - 92

Symbols, Flow Diagrams, 48

Tangent Function TAN, 14 - 15

Tape Cassette, 138, 139, 141

Tape Block Search and Load

Alpha Load Program, 142, 143

Tape Drive Operations, 139

Tape Ready Switch (Tape Control), 139

Ten-To-A-Power 10^x, 17

Terminating Loops, 67

TOTAL, Right and Left, 6

Total (T), Special Selector Switch, 22

Trace On/Trace Off, 164, 165

Example, 165

Transfer of Programs in Memory to Tape 141, 142

Trigonometric Examples, 29, 32 - 35, 37 - 44

Two-Step Commands Using Alpha

Program Step Trace, 164 - 166

Alpha Return and Print, 93, 94

(See Appendix D)

Jump Compare Commands, 79

Tape Block Search and Load

(Alpha Load Prog.), 142, 143

Storage and Load Data Registers

w/Tape, 143, 144

Upper- and Lowercase Function Keys, (Chapter 3), 12 - 19

VERIFY PROG, 101

NOTES:

PROGRAM CODES

CODE	KEY	CODE	KEY	CODE	KEY	CODE	KEY
0000	0	0015	CLD	0801	RECALL	0901	STORE
0001	1	0114	TOTAL R	0802	PRINT	0902	α
0002	2	0115	TOTAL L	0803	GO	0903	STOP
0003	3	0214	+ R	0804	J if 0	0904	J if $\neq 0$
0004	4	0215	+ L	0805	J if +	0905	J if ERROR
0005	5	0314	− R	0806	SIN	0906	SIN^{-1}
0006	6	0315	− L	0807	COS	0907	COS^{-1}
0007	7	0414	X R	0808	TAN	0908	TAN^{-1}
0008	8	0415	X L	0809	RAD − DEG	0909	DEG − RAD
0009	9	0514	÷ R	0810	$Log_e X$	0910	$Log_{10} X$
0010	DECIMAL POINT	0515	÷ L	0811	e^x	0911	10^x
0011	SET EXP	0614	ST R	0812	X^2	0912	INT X
0012	CHS	0615	ST L	0813	$\sqrt{X}$	0913	$\|X\|$
0014	CLEAR	0714	RE R	0814	LOAD PROG	0914	END
		0715	RE L	0815	1/X	0915	RETURN
		0800	SEARCH	0900	MARK		

Decimal Shifting Left *

CODE	KEY
1101	10^{-1}
1102	10^{-2}
1103	10^{-3}
1104	10^{-4}
1105	10^{-5}
1106	10^{-6}
1107	10^{-7}
1108	10^{-8}
1109	10^{-9}
1110	10^{-10}
1111	10^{-11}
1112	10^{-12}
1113	10^{-13}
1114	10^{-14}
1115	10^{-15}

Decimal Shifting Right *

CODE	KEY
1001	10^1
1002	10^2
1003	10^3
1004	10^4
1005	10^5
1006	10^6
1007	10^7
1008	10^8
1009	10^9
1010	10^{10}
1011	10^{11}
1012	10^{12}
1013	10^{13}
1014	10^{14}
1015	10^{15}

Register Keys †

CODE	KEY
XX00	00
XX01	01
XX02	02
XX03	03
XX04	04
XX05	05
XX06	06
XX07	07
XX08	08
XX09	09
XX10	10
XX11	11
XX12	12
XX13	13
XX14	14
XX15	15

Selector Switches †

CODE	SWITCH
01YY	TOTAL
02YY	+
03YY	−
04YY	X
05YY	÷
06YY	STORE
07YY	RECALL
08YY	SP
10YY	f(x)
11YY	F(x)
14YY	SP & ST (exchange data)

Alpha Characters **

CODE		SWITCH
00YY	X	All up
01YY	Y	SP up T down
02YY	Z	SP up + down
03YY	A	SP up − down
04YY	B	SP up X down
05YY	C	SP up ÷ down
06YY	D	SP up ST down
07YY	E	SP up RE down
08YY	F	SP down
09YY	G	SP down T down
10YY	H	SP down + down
11YY	I	SP down − down
12YY	J	SP down X down
13YY	K	SP down ÷ down
14YY	L	SP down ST down
15YY	M	SP down RE down

Decimal Setting **

CODE	DECIMAL PLACES	KEY
XX00	0	00
XX01	1	01
XX02	2	02
XX03	3	03
XX04	4	04
XX05	5	05
XX06	6	06
XX07	7	07
XX08	8	08
XX09	9	09
XX10	FLOATING	10
XX11	SCIENTIFIC	11
0015	PAPER FEED	15

Miscellaneous

CODE	KEY
0902	α
1000	00 } π
0902	α
0903	STOP } PAUSE
1502	I/O
1511	INDIRECT
1513	GROUP 1
1514	GROUP 2

TWO STEP ALPHA NUMERIC COMPARE BETWEEN 00 AND WINDOW

1502 } JUMP
1400 } IF EQUAL

1502 } JUMP
1500 } IF NOT EQUAL

*Must be preceded by an α command.

†Selector Switches (high order) Register Keys (low order).

**Must be preceded by a Print command.

**Alpha character (high order) Decimal Setting (low order).

ALPHA COMMANDS

KEY	DESCRIPTION	CODE
*LOAD PROG.	PROGRAM BLOCK SEARCH AND LOAD	0814
*$Log_e X$	TURNS TRACE "ON"	0810
*e^x	TURNS TRACE "OFF"	0811
*J IF 0	SKIP IF 00 = WINDOW	0804
*J IF +	SKIP IF 00 ≥ WINDOW	0805
*SIN	SKIP IF 00 < WINDOW	0806
*STORE	STORE REGISTER ON TAPE	0901
*RECALL	RECALL REGISTER FROM TAPE	0801
*RETURN	SUBROUTINE RETURN AND PRINT	0915

PERIPHERALS

ANY 600 SERIES CALCULATOR CAN ACCOMMODATE THESE PERIPHERALS



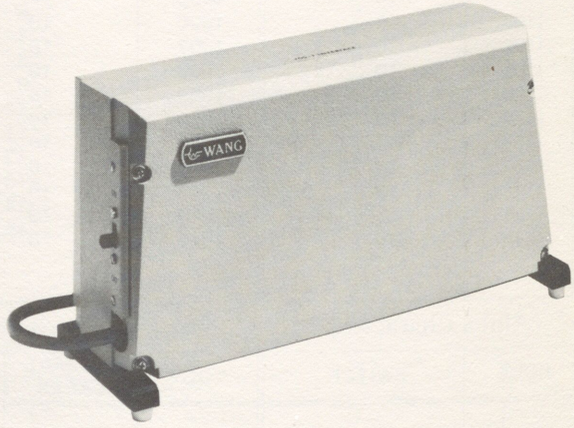
MODEL 601 OUTPUT WRITER

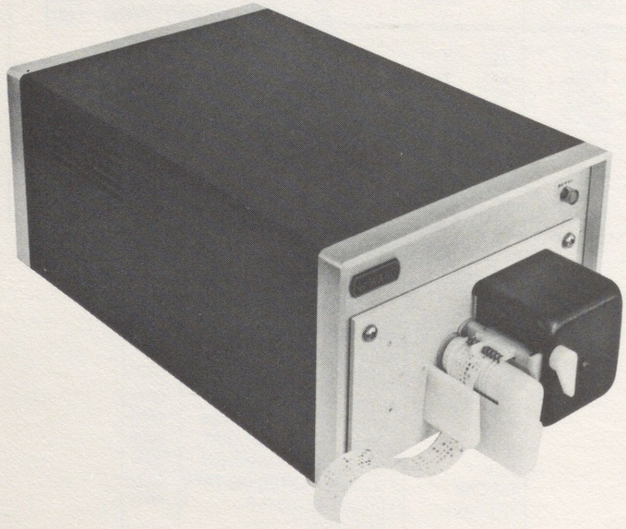
A modified IBM Selectric® typewriter is used as a printing mechanism to type numeric output, data, alphabetic headings and messages. All typewriter functions can be actuated by the calculator's special function keys to provide complete control over the alphanumeric output format. (It may also be used as a conventional typewriter.)

® Registered trademark IBM

MODEL 605-1A MICROFACE

This unit allows a variety of external devices to be interfaced to any Wang 600 calculator.

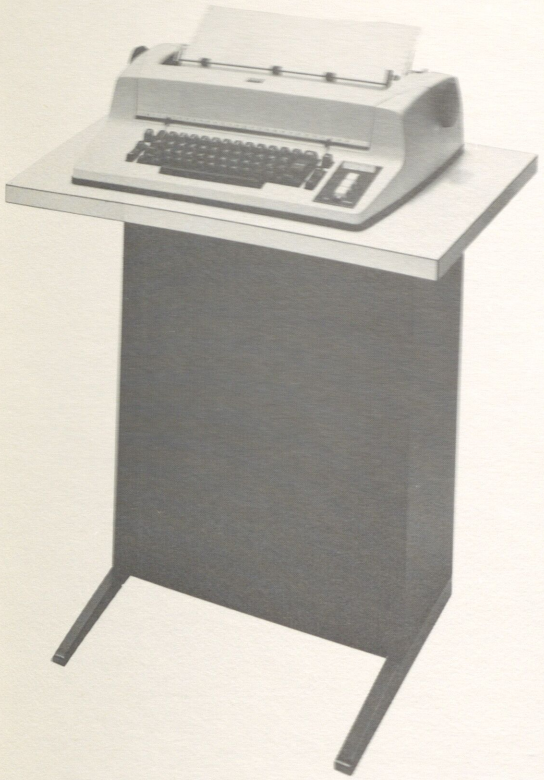




MODEL 603 PAPER TAPE EDITOR

The Model 603 Paper Tape Editor provides off-line punched paper tape users with a convenient and economical means of obtaining calculated output from raw data.

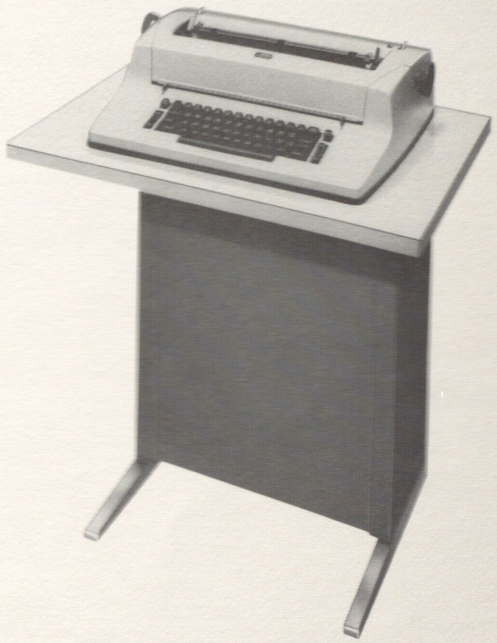
PERIPHERALS (cont.)



MODEL 611 INPUT/OUTPUT WRITER

The Model 611 Input/Output Writer is a modified IBM Selectric® typewriter which can handle input/output applications in conjunction with the 600 calculator. Alpha printout formatting is easily generated directly on the 611 keyboard and loaded into the memory of the calculator for easy access. (It may also be used as a conventional typewriter.)

® Registered trademark IBM

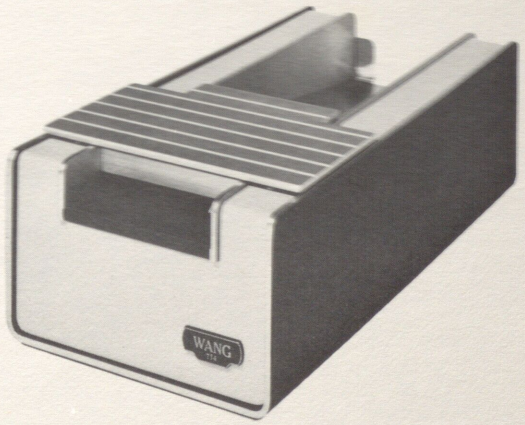


MODEL 602 PLOTTING OUTPUT WRITER

This peripheral adds digital plotting and alphanumeric output writing capabilities to Wang's 600 Series calculators. It is a dual purpose unit that can provide any number of alphanumeric printout and plot combinations either manually or under program control. (It may also be used as a conventional typewriter.)

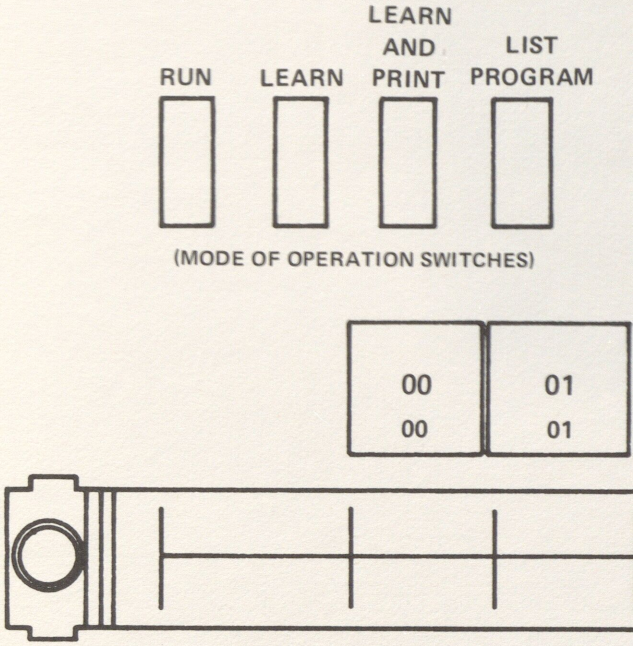
MODEL 612 ANALOG PLOTTER

A flat-bed plotter which provides a continuous line plot with alphanumeric labeling capabilities.



MODEL 614 MARKED SENSE CARD READER

The Marked Sense Card Reader allows off-line programming for software development without the use of the calculator, thus freeing the keyboard for other calculating applications.



S.M. PRIME	SIN^{-1} 0906 SIN 0806	C CO 08
DEG-RAD 0909 RAD-DEG 0809	TAN^{-1} 0908 TAN 0808	RE 1/ 08
S H I F T	$LOG_{10}X$ 0910 LOG_eX 0810	e 08
	INT X 0912 X^2 0812	$\sqrt{X}$ 081

RUN | LEARN | LEARN AND PRINT | LIST PROGRAM

(MODE OF OPERATION SWITCHES)

CLEAR 00 14 | T 01 | + 02 | – 03 | X 04 | ÷ 05 | ST 06 | RE 07 | fx 10 | SP ON 08 | FL Sc | DEG RAD | PRINTER ON | PAPER FEED

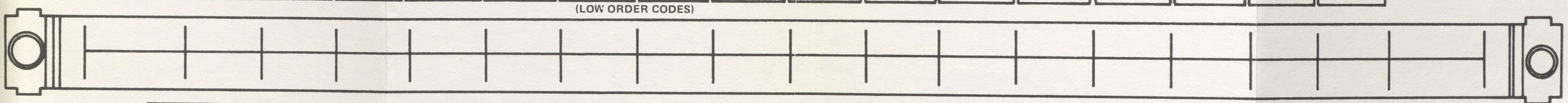
(HIGH ORDER CODES)

RELEASE | FORWARD | TAPE READY | REWIND

PROG ERROR | MACH ERROR

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15

(LOW ORDER CODES)



S.M. PRIME	SIN^{-1} 0906 SIN 0806	COS^{-1} 0907 COS 0807
DEG-RAD 0909 RAD-DEG 0809	TAN^{-1} 0908 TAN 0808	RETURN 0915 1/X 0815
SHIFT	$LOG_{10}X$ 0910 LOG_eX 0810	10^x 0911 e^x 0811
	INT X 0912 X^2 0812	|X| 0913 $\sqrt{X}$ 0813

TOTAL 0115	– 0315
÷ = 0515	+ 0215
X = 0415	
STORE 0615	RECALL 0715

CHANGE SIGN 0012	7 0007	8 0008	9 0009
CLEAR DISP 0015	4 0004	5 0005	6 0006
SET EXP 0011	1 0001	2 0002	3 0003
0 0000		• 0010	

– 0314	TOTAL 0114
+ 0214	÷ = 0514
	X = 0414
RECALL 0714	STORE 0614

END 0914 LOAD PROG 0814	≠0 0904 J IF 0 0804	I/O 1502	INS SET P.C.
MARK 0900 SEARCH 0800	ERROR 0905 J IF + 0805	GROUP 1 1513	B.S. VERIFY PROG
STOP 0903 GO 0803	STORE 0901 RECALL 0801	GROUP 2 1514	DEL RECORD PROG
	α 0902 PRINT 0802	INDIR 1511	STEP

WANG
LABORATORIES, INC

836 North Street
Tewksbury, Massachusetts 01876
TELEPHONE (617) 851-7311
TWX 710-343-6769
TELEX 94-7421

WANG LABORATORIES (CANADA) LTD.
180 Duncan Mill Road
Don Mills, Ontario
TELEPHONE (416) 449-7890
TELEX 06-21-7549

WANG EUROPE, S.A.
Rue Anatole France 115-121
1030 Brussels, Belgium
TELEPHONE 15.48.01/02/03

WANG ELECTRONICS LTD.
40-44 High Street
Northwood, Middlesex, England
TELEPHONE Northwood 27677

WANG FRANCE SARL
47 Rue de la Chapelle
Paris 18, France
TELEPHONE 203.27.94 or 203.25.94

WANG LABORATORIES GMBH
Moselstrasse No. 4
6000 Frankfurt am Main
West Germany
TELEPHONE (611) 23-00-40

WANG SKANDINAVISKA AB
Fredsgatan 17
S-172 33
Sundbyberg, Sweden
TELEPHONE 98-12-45

WANG NEDERLAND N.V.
Damstraat 2
Utrecht, Netherlands
TELEPHONE 030-980947

WANG PACIFIC LTD.
61, King Yip Street, 1st Floor
Kwun Tong, Kowloon, Hong Kong
TELEPHONE K-444536

WANG INDUSTRIAL CO., LTD.
9, Sec. 5, Nanking East Road
Taipei, Taiwan
TELEPHONE 77.96.19 or 71.35.78

WANG MEDICAL SYSTEMS, INC.
836 North Street
Tewksbury, Massachusetts 01876
TELEPHONE (617) 851-7311
TWX 710-343-6769
TELEX 94-7421

PHI COMPUTER SERVICES, INC.
800 Massachusetts Avenue
Arlington, Massachusetts 02174
TELEPHONE (617) 648-8550

WANG ELECTRONIK GESELLSCHAFT MBH
Grinzinger Allee 16
1190 Wien 19
Austria
TELEPHONE (0222) 32.42.43

Printed in U.S.A.
700-0556
4-72-5M
Price $10.00