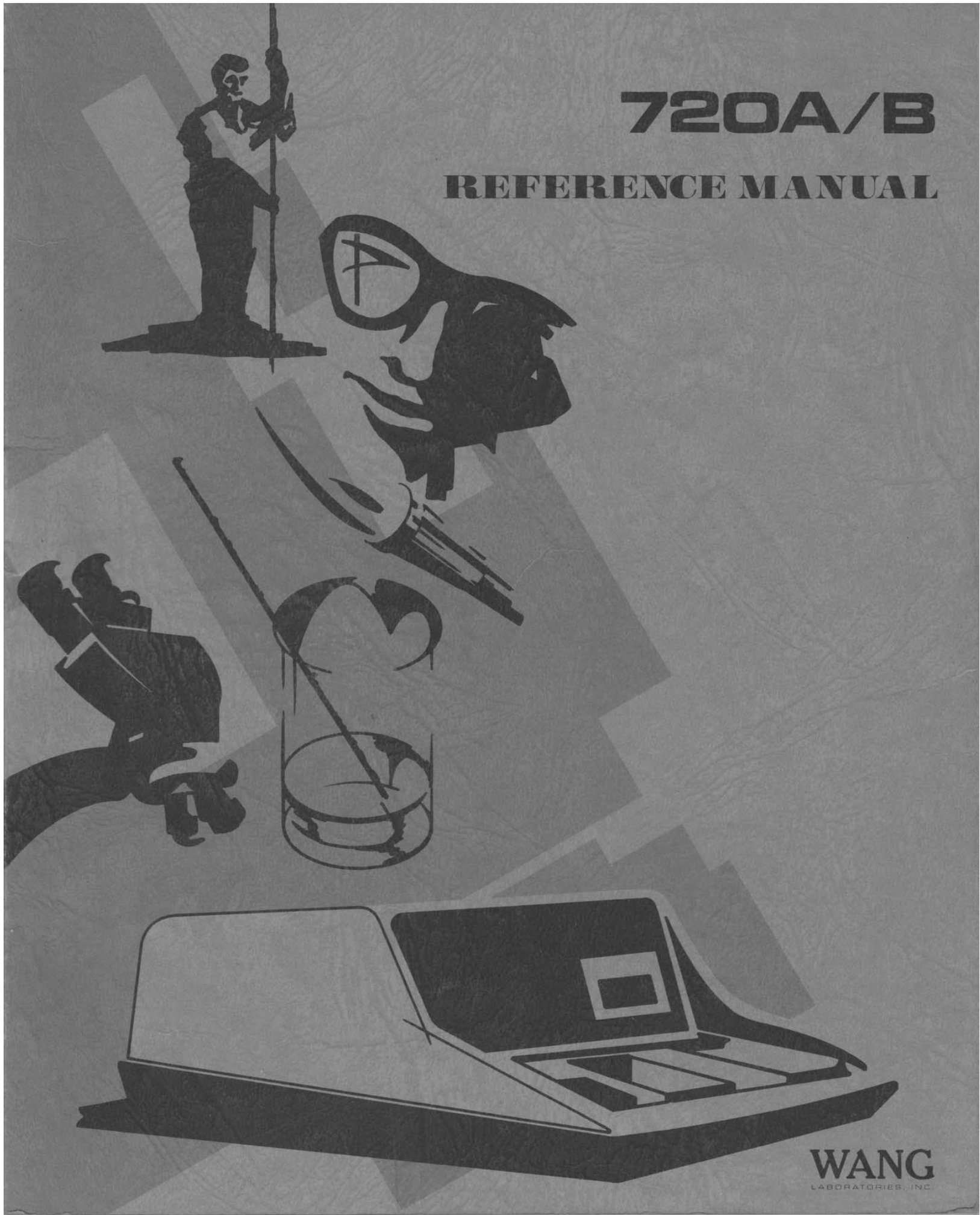


720A/B

REFERENCE MANUAL



WANG
LABORATORIES, INC.

720A/B

REFERENCE MANUAL

© WANG LABORATORIES, INC., 1970
Tewksbury, Mass. 01876
Telephone (617) 851-7311
TWX 710 343-6769

PREFACE

Wang Laboratories, America's leading manufacturer of electronic calculating systems would like to take this opportunity to thank you for your purchase of Wang calculating equipment.

As a company we are committed to a total program of services to meet the needs of our customers. We maintain an extensive library of programs and technical services. Our worldwide service organization complemented by trained personnel assures downtime to be at a minimum if at all.

Wang equipment is designed around our renowned modularity concept, which allows a user to expand his system as his requirements expand. Add-on optional items such as, a printer, display, extra storage, programming, and special function keys are available that can be installed easily and efficiently. These features and many more are designed especially to meet the needs of any calculating requirements that may arise.

When the calculator arrives it is simply a matter of plugging the unit into a power source, turning it "ON", depressing the Clear button, and start the calculating.

This manual provides operating instructions for the Wang Model 720A/B Programmable Electronic Calculator. Illustrations are typical and show the principal design features of the unit.

These instructions have been prepared as a service to our customers and are offered as suggested methods for the customers to follow so that the equipment may be installed, operated and maintained properly.

Wang Laboratories is dedicated to manufacturing the finest calculating equipment in the world. We are ready to serve you in any way that will best meet your requirements.

For further information, contact your local sales office or Wang Laboratories, Inc. 836 North Street, Tewksbury, Massachusetts 01876.

Remember "We do more so you can".

SECTION I – INTRODUCTION

SECTION II – EXPLANATION OF KEYS

SECTION III – PROGRAMMING

SECTION IV – PROGRAM CONCEPTS

SECTION V – DECISION COMMANDS

SECTION VI – PROGRAMMING TECHNIQUES USING A TAPE CASSETTE

SECTION VII – ADDITIONAL COMMANDS NOT FOUND ON THE 720 KEYBOARD

SECTION VIII – TRIGONOMETRIC PACKAGE PROGRAM, STATISTICAL PACKAGE PROGRAM

SECTION IX – SAMPLE PROGRAM

SECTION X – WARRANTY, SERVICE AND MAINTENANCE

TABLE OF CONTENTS

SECTION I – INTRODUCTION

Introduction	1-1
------------------------	-----

SECTION II – EXPLANATION OF KEYS

Modes of Operation	2-1
Run Mode	2-1
Learn Mode	2-1
Learn-Print Mode	2-1
List-Program Mode	2-2
Turning the 720 ON	2-2
Non Programmable Keys	2-2
Prime	2-2
Program Counter and Set PC	2-3
Step	2-3
Verify Program	2-4
Record Program	2-4
The Display	2-4
X-Register	2-5
Entering a Number	2-5
Set Exponent	2-5
Y-Register	2-6
Program-Error Indicator	2-8
Data Storage Registers	2-9
Direct Addressing	2-9
Toggle Switches and Special Function Keys	2-9
Store Direct	2-12
Recall Direct	2-12
Exchange Direct	2-13
Add, Subtract, Multiply, and Divide Direct	2-13
Indirect Addressing	2-14
Indirect Keys	2-14
Advantages of Indirect Addressing	2-16
Recall Residue	2-16
Addition, Subtraction, Multiplication	2-17
Division	2-18
Write Commands	2-20
Group 1 – Group 2	2-21

SECTION III – PROGRAMMING

Coding	3-1
Generating a Code Using Special Function Keys and Toggle Switches	3-2
Core Memory	3-3
Number of Registers Occupied By a Program	3-5

SECTION IV – PROGRAMMING CONCEPTS

Programming Concepts	4-1
Mark and Search Commands	4-2
Branching	4-3
Looping	4-4
Subroutine	4-5
Multi-Level Subroutines (Or a Subroutine within a Subroutine)	4-7

TABLE OF CONTENTS (Continued)

SECTION V – DECISION COMMANDS

DECISIONS	5-1
Skip if $Y = X$	5-1
Skip if $Y \geq X$	5-2
Skip if $Y < X$	5-3
Skip if Error	5-3
PROGRAMMING TECHNIQUES	5-4
Looping Using a Counter	5-4
Looping Without a Counter	5-6
Scanning a Table	5-8
Go	5-9

SECTION VI – PROGRAMMING TECHNIQUES USING TAPE CASSETTE

Tape Cassette	6-1
Tape Drive Operations	6-2
Machine Error Indicator	6-2
Protection of Program on Tape	6-3
What is a Tape Block?	6-3
End Program	6-4
How to "Learn" a Program Into Core From the Keyboard	6-5
How to Transfer a Program From Core to Tape	6-6
How to Load a Program From Tape Into Core	6-7
Bypassing Program Blocks	6-8
Procedure For Correcting Single Program Steps	6-8
Procedure For Inserting Extra Program Steps	6-9
Programming Techniques Using Tape Cassette	6-9
Creating a Multi-Block Tape	6-11

SECTION VII – ADDITIONAL COMMANDS NOT FOUND
ON THE 720 KEYBOARD

Pause Command	7-1
Write a Pause	7-1
Storage Commands (Direct Access To and From the Y-Register)	7-2
Additional Commands for Addressing the 720 Register	7-3
Decisions	7-3
X-Register	7-4
Y-Register	7-4
Shifting Commands	7-5

SECTION VIII – TRIGONOMETRIC AND STATISTICAL
PACKAGE PROGRAMS

The Trig Pack	8-1
Speed and Accuracy	8-2
To Load the Trig Package	8-2
Using the Trig Package	8-3
Program Use	8-4
Design of the Trig Package	8-4
Statistical Package Program	8-5
Assignment of Special Operation Keys for a User's Own Subroutines	8-6

TABLE OF CONTENTS (Continued)

SECTION IX – SAMPLE PROGRAM

Algebra of Complex Numbers $+$, $-$, $\times$, $\div$ (Program)	9-1
--	-----

SECTION X – WARRANTY, SERVICE AND MAINTENANCE

Warranty	10-1
Post-Warranty Service Availability	10-1
Annual Maintenance Contract	10-1
Post-Warranty Service Calls Without Maintenance Contract	10-1
In-House Maintenance Capability	10-2

APPENDIX

Typing Conventions for 700 Series Program Library	A-1
Index	A-4

RUN

LEARN
PRINT

LEARN
PROGRAM

LIST

PROGRAM

RELEASE

FORWARD

TAPE READY

REWIND

00

01

02

03

04

05

06

07

08

09

10

11

12

13

14

15

80

40

20

10

DEGREE
TO RADIANS

RADIANS
TO DEGREE

SINX

COSX

TANX

$\sin^{-1} x$

$\cos^{-1} x$

$\tan^{-1} x$

TO POLAR

TO RECT

SINHx

COSHx

TANHx

$\sinh^{-1} x$

$\cosh^{-1} x$

$\tanh^{-1} x$

MACHINE
ERROR

PROGRAM
ERROR

MACHINE
ERROR

WRITE ALPHA	END ALPHA	RECALL INDIR	$\left(\frac{\circ}{\circ}\right)$ INDIR	$\left(\frac{\circ}{\circ}\right)$ DIRECT	RECALL DIRECT	CHANGE SIGN	$\sqrt{x}$	x^2	CLEAR X	PRIME
WRITE	1/X	STORE INDIR	$\div$ INDIR	$\div$ DIRECT	STORE DIRECT	$\div$	7	8	9	MARK
INTEGER X	x	RECALL RESIDUE	X INDIR	X DIRECT	$\uparrow$	X	4	5	6	RETURN
10^x	$\log_{10} x$	π	$\overline{\text{INDIR}}$	$\overline{\text{DIRECT}}$	$\uparrow$	—	1	2	3	SKIP IF ERROR
e^x	$\log_e x$	$\left(\frac{\circ}{\circ}\right)$	$\div$ INDIR	$\div$ DIRECT	$\uparrow$	*	0	*	SET EXP	SKIP IF $y \geq x$
										GO
										LOAD PROG
										END PROG
										STOP
										SKIP IF $y = x$
										GROUP 1
										RECORD PROG
										SKIP IF $y < x$
										SEARCH
										STEP

720A/B KEYBOARD ILLUSTRATION

SECTION I

INTRODUCTION

The Wang 720 is the ultimate solution to many calculating needs. Simple or complex calculations can be done right at the desk. The 720 keyboard is extremely simple to operate. Once the fundamental operations have been mastered, programming the 720 is easy.

The Wang 720 is a self-contained programmable electronic calculator constructed with integrated circuits on snap-in replaceable printed circuit modules. The 720 is composed of three basic elements:

1. The Central Processing Unit
2. Read-Only Memory
3. Core Memory

The Central Processing Unit (CPU) is the hardware which performs the arithmetic operations. The **Read-Only Memory** guides the CPU in all its operations. In effect, the Read Only Memory is "the brains" of the Wang 720. It directs all arithmetic and logical operations on the 720 and has been programmed to perform all the functions found on the 67 keys of the 720 keyboard.

The 720 Core Memory is organized into 249 data registers; plus a nixie display of the two work registers X and Y; 248 registers can be used for data storage or program storage. All user programs are executed from core memory. The trig functions are also executed from core memory.

Section II explains the modes of operation on the 720, as well as the five non-programmable keys; discusses the dual nixie-type display readout and the basic arithmetic operations; explains direct and indirect addressing of the data storage registers; explains the **RECALL RESIDUE** key — a unique Wang feature that makes double-precision arithmetic a simple operation on the 720.

The usefulness of the Wang 720 comes from its programming capabilities. A program is simply a logical sequence of steps which the calculator can perform automatically over and over again on different variables. If the calculation is to be performed only once, it probably is simple enough to do it manually on the keyboard. However, if the same calculations are to be done repeatedly, it is beneficial to record and save the steps of the calculation in the form of a program and let the calculator perform these repeated operations. The program is loaded into core and executed from core. Programs can also be stored on magnetic tape for later use.

Sections III, IV, V, VI, and VII discuss various techniques to be used in programming the Wang 720. They explain how to introduce a program into core memory and how to save it for later use on tape cassettes. Also, they explain how several parts of a program can share the same part of core memory. Section VIII discusses the TRIG functions of the Wang

INTRODUCTION

720. Section IX gives an example of a 720 Program and further illustrates the concept of indirect addressing. Section X contains warranty, service, and maintenance information.

An appendix is included in this manual which covers typing conventions and contains an index to help the user locate with ease certain items of interest.

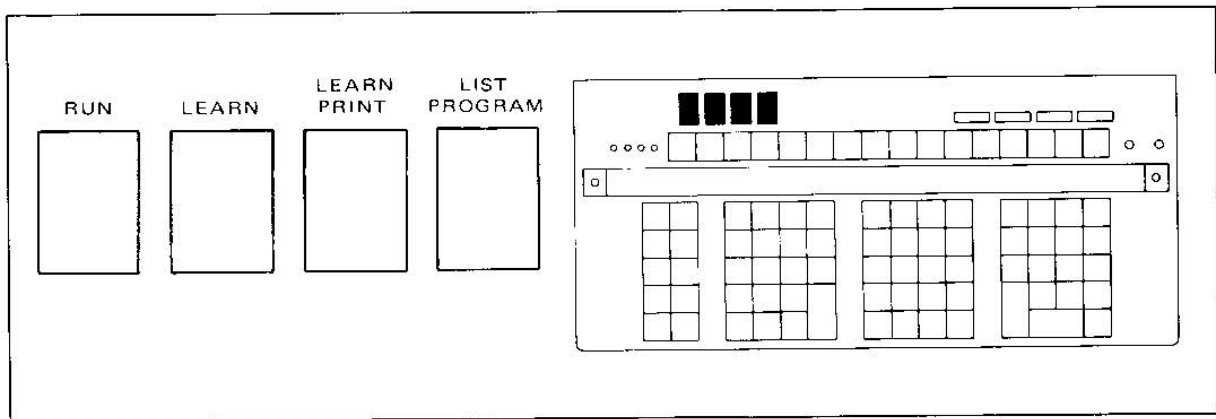
720A – 720B

	PROGRAM STEPS	REGISTERS
720A	1984	000–247 248* (Scratch Pad Only) 5 Level Subroutine Drives 701 Output Writer
720B	1984	000–247 248* (Scratch Pad Only) 5 Level Subroutine Drives 701 Output Writer 702 Plotter

*The 720A register 248 and the 720B register 248 may be used as scratch pads only if subroutines 00-00 through 01-15 are not accessed. When these subroutines are called upon, the Y register is automatically stored in this register.

SECTION II EXPLANATION OF KEYS

MODES OF OPERATION



The 720 has four different modes of operation. The four lock-in switches located above the toggle switches on the keyboard are used to put the 720 into a desired mode of operation.

RUN MODE

The RUN MODE is used for most 720 operations. All keyboard calculations are done in the RUN MODE. In fact, practically all operations except introducing a program into core memory from the keyboard are performed on the 720 in the RUN MODE.

LEARN MODE

The 720 is put into LEARN MODE when a program is to be written into core. Every programmable key which is indexed while the 720 is in LEARN MODE is "learned" or recorded in core. In LEARN MODE the Y-Register is blanked and the X-Register displays the program step number and the program code stored at that step.

LEARN-PRINT MODE

The LEARN-PRINT MODE and the LIST PROGRAM MODE are used only when the output writer is available. In the LEARN-PRINT MODE, each key indexed is "learned" into core and is also listed on the output writer. As each key is indexed, the program step number and the program code of the key are listed on the output writer, giving the user a hard copy of his program as he writes it!

Section II
Explanation of Keys

LIST-PROGRAM MODE

When the 720 is put in the LIST-PROGRAM MODE and the **GO** key is depressed, it automatically lists the program steps and program code in increments of 1000 steps until it encounters an **END PROGRAM** code. The LEARN-PRINT and LIST-PROGRAM modes are discussed in greater detail in the 701 OUTPUT WRITER MANUAL.

TURNING THE 720 ON

The procedure for turning the 720 on consists of three steps:

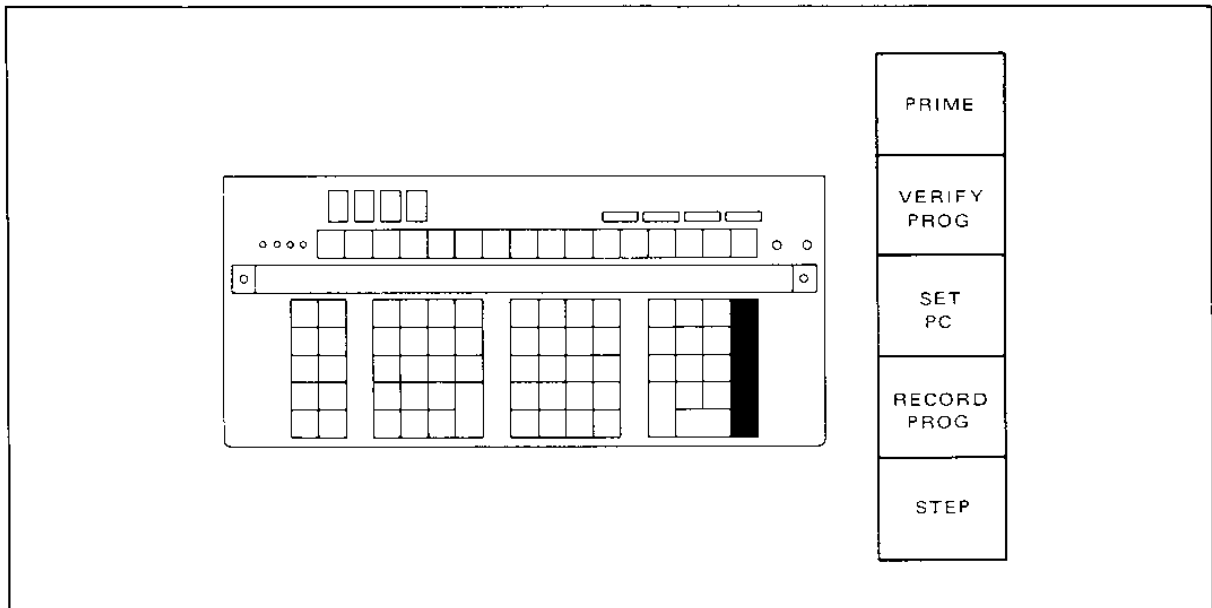
- 1. Turn power switch ON. (located on the rear panel.)*
- 2. Index PRIME to initialize the system.*
- 3. Select mode of operation. (In most instances the RUN mode will be selected. Depress RUN button.)*

The Wang 720 is now in RUN MODE ready to perform your calculations.

NON PROGRAMMABLE KEYS

Because of their function, there are five keys which cannot be programmed on the Wang 720. Each of these commands is discussed briefly in this section. All of their functions and uses will become clear after reading the entire manual.

The five non programmable keys on the 720 are:



PRIME

The **PRIME** key initializes the 720 system and should always be depressed when the 720 is first turned on. It also performs the following operations:

1. Clears Y-Register to zero.
2. Clears X-Register to zero.
3. Sets the program counter to Step 0000.
4. Resets the program-error and machine-error indicators.

The **PRIME** key should be depressed when the 720 is first turned on.

NOTE

*The **PRIME** key should not be depressed when any operation is being executed. If the program is to be stopped during execution, the **STEP** key should be used. This will stop the program after the current step is executed. Also, indexing the **PRIME** key when **RECORD PROGRAM** or **LOAD PROGRAM** commands are being executed will cause difficulty with the tape. If the **PRIME** key is indexed accidentally during a **RECORD PROGRAM** or **LOAD PROGRAM** operation, the operation will be terminated immediately. However, the tape should be rewound before executing any other tape operations.*

PROGRAM COUNTER AND SET PC

The program counter or PC is a counter which counts from 0000 to 1983. It indicates which program step is about to be executed. At all times, it points to the next program step. Thus, when the machine is performing step 0108, the PC is already on step 0109.

The **SET PC** key allows the user to address and set the program counter with the next three keystrokes.

SET PC 0018

This instruction sets the program counter to program step number 0018. To set the program counter requires five keystrokes: **SET PC** followed by four numeric keys. **PRIME** automatically sets the program counter to step number 0000.

STEP

The **STEP** key allows the user to step through his program one step at a time. If the program is running when the **STEP** key is indexed, the program stops at the step it is about to execute. In the RUN MODE, depressing the **STEP** key will cause the 720 to perform the next step in the program. Each time the **STEP** key is indexed, the next program step will be executed.

The **GO** key will take the 720 out of the stepping mode and put it in the continuous mode executing the remaining steps in the program until a **STOP** command is encountered.

NOTE

*In any 2-step command such as **DIRECT ADDRESSING** and **WRITE ALPHA** commands, the **GO** key should not be depressed in the middle of the 2-step command. The entire 2-step command should be executed in step mode before switching to the continuous mode.*

This stepping feature is of tremendous value for debugging programs. The programmer can step through his program and locate his difficulty immediately. By switching to **LEARN**

Section II
Explanation of Keys

MODE he can see the step number and the code of the operation he is **about to execute**. When stepping through a program in **LEARN MODE**, the program step number (the PC) and the program code of the operation are displayed in the X-Register. However, in **LEARN MODE**, the operation is not executed.

VERIFY PROGRAM

The **VERIFY PROGRAM** key decimally adds the high-order and low-order digits of the program codes in core beginning at step 0000 until it encounters an **END PROGRAM** code. The sum is displayed in the right-most digits of the mantissa of the X-Register.

EXAMPLE

STEP	KEY	CODE	If this program is located in core and the VERIFY PROGRAM key is depressed, the sum displayed in X is 59.
0000	MARK	0408	
0001	0700	0700	{ 04 08 07 00 07 13 05 15 59
0002	x ²	0713	
0003	STOP	0515	
0004	END PROG	0512	

After performing this operation, the PC is set at the step where the **END PROGRAM** command is located. (Step 0004 in this example.) Notice the code for **END PROGRAM** is not added into the sum generated by the **VERIFY PROGRAM** key.

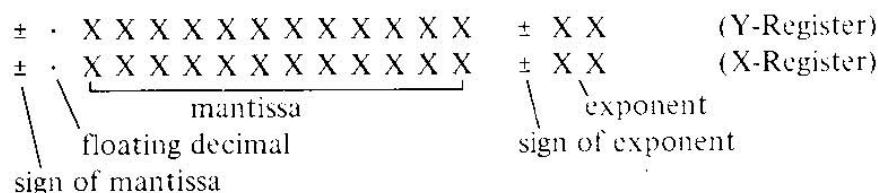
RECORD PROGRAM

This key transfers a program from core to magnetic tape. The PC is set to a specific step and the program steps starting at this step are transferred to the tape until an **END PROGRAM** command is reached. The **END PROGRAM** command is the last step transferred to the tape. After transferring the program to tape, the PC is set to where it was originally set (i.e., the first program step to be transferred to the tape).

The five keys, **PRIME**, **SET PC**, **VERIFY PROGRAM**, **RECORD PROGRAM** and **STEP** are the only keys on the 720 which cannot be used in a program. Therefore, when any of these keys are indexed it doesn't matter whether or not the machine is in **LEARN** or **RUN MODE**.

THE DISPLAY

The display consists of two work registers, X and Y. Both the X and Y Registers are displayed simultaneously by easily readable half-inch nixie-type tubes. Each register has a $\pm$ sign and twelve digit mantissa followed by a two-digit exponent with a range of -99 to $+99$.



For numbers in the range $.1 \leq |N| < 1\,000\,000\,000$, the decimal point retains its natural position. When a number lies outside this range, the decimal automatically relocates to the extreme left, and the exponent of the power of 10 is indicated correctly in modified scientific notation. This property will become clear after a few minutes familiarization with the keyboard.

(A few numbers and how they appear in the display are given below.)

X-REGISTER

The keys 0, 1, 2, . . . 9 and decimal point (.) are used for entering a number into the X-Register. The **SETEXP** key is used to set the exponent value of X. The **CH SIGN** key changes the algebraic sign of the mantissa or exponent of X.

Indexing a number into the 720 keyboard is as simple as writing the number down on paper. The normal sequence of steps is to key in the mantissa followed by the **SETEXP** key and the value of the exponent.

ENTERING A NUMBER

Index the following few numbers on the 720 keyboard. After indexing the number into the X-Register, move it to the Y-Register by depressing the $\uparrow$ key.

NUMBER	SEQUENCE OF STEPS	DISPLAY
a) $.152 \times 10^{21}$	<u>1</u> <u>5</u> <u>2</u> SETEXP <u>2</u> <u>1</u> $\uparrow$	.152000000000 + 21
b) 6.62517×10^{-27}	<u>6</u> <u>6</u> <u>2</u> <u>5</u> <u>1</u> <u>7</u> SETEXP CHS <u>2</u> <u>6</u> $\uparrow$	.662517000000 - 26
c) -2534.5	<u>2</u> <u>5</u> <u>3</u> <u>4</u> <u>.</u> <u>5</u> CHS $\uparrow$ Or <u>2</u> <u>5</u> <u>3</u> <u>4</u> <u>5</u> CHS SETEXP <u>4</u> $\uparrow$	-2534.50000000
d) .0075	<u>0</u> <u>0</u> <u>7</u> <u>5</u> $\uparrow$ Or <u>7</u> <u>5</u> SETEXP CHS <u>2</u> $\uparrow$	+.750000000000 - 02

Index c and d both ways. Does the display appear differently? Notice example b. Why is 26 entered as the value of the exponent?

SET EXP

The **SETEXP** key is used to set the exponent value of X with the next two successive keystrokes. The **SETEXP** key automatically aligns the decimal point in the left-most position of the X-Register; however, it does not blank out the mantissa. This allows us to change the value of the exponent of a number without having to key in the entire number again.

Section II
Explanation of Keys

EXAMPLE

Index 1.75×10^2

Suppose the following sequence of steps is used:

1 . 7 5 SETEXP 2 3

Notice what happens to the decimal point when the **SETEXP** key is indexed. It is not necessary to index the decimal point, as the **SETEXP** key automatically aligns it in the left most position. The value of the exponent will also have to be indexed correctly. If the number is in proper scientific notation, the value of the exponent is simply increased by 1. Thus, the correct sequence of steps would be:

CLEAR X 1 7 5 SETEXP 2 4

All numbers indexed after the **SETEXP** key simply changes the value of the exponent. Since the range of the exponent is -99 to $+99$, normally only 1 or 2 numbers are indexed after the **SETEXP** key. However, if more than 2 numbers are indexed, the exponent takes on the value of the last 2 numbers entered.

EXAMPLE

If the following sequence of steps is performed:

1. **1 2 SETEXP 2 3 4**, the value of the exponent is 34.
2. For **SETEXP CHS 3 5 7**, the value of the exponent is -57 .
3. For **SETEXP 5 0 2**, the value of the exponent should be 2. However, on the display the exponent would be blanked out and the decimal point would assume its natural position.

The 720 will remain in the **SETEXP** mode until a non numeric key or the decimal point key is depressed.

Y-REGISTER

The Y-Register is another work register used in conjunction with the X-Register for basic arithmetic operations and data transfers. A number in the X-Register can easily be transferred to the Y-Register by indexing the $\uparrow$ key or $\downarrow\uparrow$ key.

KEYSTROKE	OPERATION
CLEAR X	Clears X-Register
$\uparrow$	X into Y, X unchanged
$\downarrow$	Y into X, Y unchanged
$\downarrow\uparrow$	X and Y exchanged
+	$Y+X$ into Y, X unchanged
-	$Y-X$ into Y, X unchanged
x	$Y \times X$ into Y, X unchanged

$\div$	$Y \div X$ into Y, X unchanged
X	Absolute value of X into X, Y unchanged
INT X	Disregards decimal part of number in X and puts integer part of number in X, Y unchanged
1/X	1/X into X, Y unchanged
x^2	x^2 into X, Y unchanged
$\sqrt{x}$	$\sqrt{x}$ into X, Y unchanged
$\text{LOG}_{10} X$	$\text{LOG}_{10} X$ into X, Y unchanged
10^X	10^X into X, Y unchanged
$\text{LOG}_e X$	$\text{LOG}_e X$ into X, Y unchanged
e^X	e^X into X, Y unchanged
π	π into X, Y unchanged

Step through the following examples to familiarize yourself with these keyboard operations.

EXAMPLE 1. Calculate $.083 + 17.86 + 32.2 = +50.1430000000$ 1. PRIME 2. . 0 8 3 $\uparrow$ 3. 1 7 . 8 6 + 4. 3 2 . 2 + (Answer in Y, 32.2 in X)			
EXAMPLE 2. Calculate $(5)^2 - (20)^2 + (1/15)^2 + \sqrt{70} = -366.628955291$ 1. CLEAR X 2. 5 x^2 $\uparrow$ 3. 20 x^2 - 4. 1 5 1/x x^2 + 5. 70 $\sqrt{x}$ + (Answer in Y, $\sqrt{70}$ in X)			
EXAMPLE 3. Calculate $\frac{51}{\sqrt{26}} \times 6.2 = +62.0119219307$ <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; vertical-align: top;"> 1. 5 1 $\uparrow$ 2. 6 . 2 x 3. 2 6 $\sqrt{x}$ $\div$ (Answer in Y, $\sqrt{26}$ in X) </td><td style="width: 50%; vertical-align: top;"> 1. 5 1 $\uparrow$ 2. 2 6 $\sqrt{x}$ $\div$ 3. 6 . 2 x (Answer in Y, 6.2 in X) </td></tr> </table>		1. 5 1 $\uparrow$ 2. 6 . 2 x 3. 2 6 $\sqrt{x}$ $\div$ (Answer in Y, $\sqrt{26}$ in X)	1. 5 1 $\uparrow$ 2. 2 6 $\sqrt{x}$ $\div$ 3. 6 . 2 x (Answer in Y, 6.2 in X)
1. 5 1 $\uparrow$ 2. 6 . 2 x 3. 2 6 $\sqrt{x}$ $\div$ (Answer in Y, $\sqrt{26}$ in X)	1. 5 1 $\uparrow$ 2. 2 6 $\sqrt{x}$ $\div$ 3. 6 . 2 x (Answer in Y, 6.2 in X)		
EXAMPLE 4. $A = \pi r^2$ $r = .568 \times 10^{-6} = +.101355318827 - 11$ 1. 5 6 8 SETEXP 6 CHS 2. x^2 $\uparrow$ π . 3. $\times$ (Answer in Y, π in X)			

Section II
Explanation of Keys

EXAMPLE 5. Calculate $(12.8)^{7/3} = +383.256852976$

- | | | |
|-----------------------------------|----|------------------------------------|
| 1. 1 2 . 8 LOG _e X | | 1. 1 2 . 8 LOG ₁₀ X |
| 2. ↑ 7 x | or | 2. ↑ 7 x |
| 3. 3 ÷ | | 3. 3 ÷ |
| 4. ↓ e ^x (Answer in X) | | 4. ↓ 10 ^x (Answer in X) |

EXAMPLE 6. Reduce the angle 865° to an equivalent angle less than 360°.

Formula $\left(\frac{865}{360} - \text{INT} \left[\frac{865}{360} \right] \right) 360 = \text{equivalent (145) value}$

1. 8 6 5 ↑
2. 3 6 0 ÷
3. ↓ INT (X)
4. —
5. 3 6 0 x (Answer in Y, 360 in X)

EXAMPLE 7. Calculate the following:

- a. $c = 2 \pi r$ where $r = .347 \times 10^{-5} = .21802 \dots \times 10^{-4}$
- b. $M = \frac{90 + 87 + 68 + 77}{4} = 80.5$
- c. $\sqrt{M} + \frac{1}{\sqrt{M}}$ where M is the answer of 7 (b)
Answer = 8.984 ...
Hint: Use ↑ ↓ key
- d. $e^{5.3} + 10^{5.7} + \pi^2 = 501397.4 \dots$
- e. Log (-2) What happens? Why? **PRIME** and find $\sqrt{-3}$. What happens? Why?

PROGRAM-ERROR INDICATOR

There are two lights located to the right of the Special Function Keys on the 720 keyboard. These two lights are used as error indicators. The one on the right indicates MACHINE ERROR; the one on the left PROGRAM ERROR. The MACHINE ERROR INDICATOR is discussed later.

The PROGRAM ERROR INDICATOR is turned on whenever an illegal operation is performed (i.e., taking the logarithm or square root of a negative number, or dividing by zero). Also, if a calculated result is greater than 10^{99} , the PROGRAM ERROR INDICATOR will be turned on. Whenever the indicator is on, the arithmetic sign of the X-Register also flashes.

OPERATIONS WHICH TURN PROGRAM ERROR INDICATOR ON

Calculated result greater than 10^{99} (Overflow condition)

Division by 0

$\sqrt{-x}$

LOG₁₀ x where $x \leq 0$

Log e^x where $x \leq 0$
 Searching Non-Existent Flag (See page 4-3).
 Addressing All Illegal Data Register (Any Register Greater than 249).
 Program Overlaps Core (See BYPASSING PROGRAM BLOCKS page 6-8).
 Program Block is Missing An **END PROGRAM** Instruction (See
 Definition of PROGRAM BLOCK page 6-4).

The **PRIME** key or **SKIP IF ERROR** key is used to turn the PROGRAM ERROR INDICATOR off. In programming, a **SKIP IF ERROR** command is available to test for this condition. Performing this test will also turn the indicator off.

DATA STORAGE REGISTERS

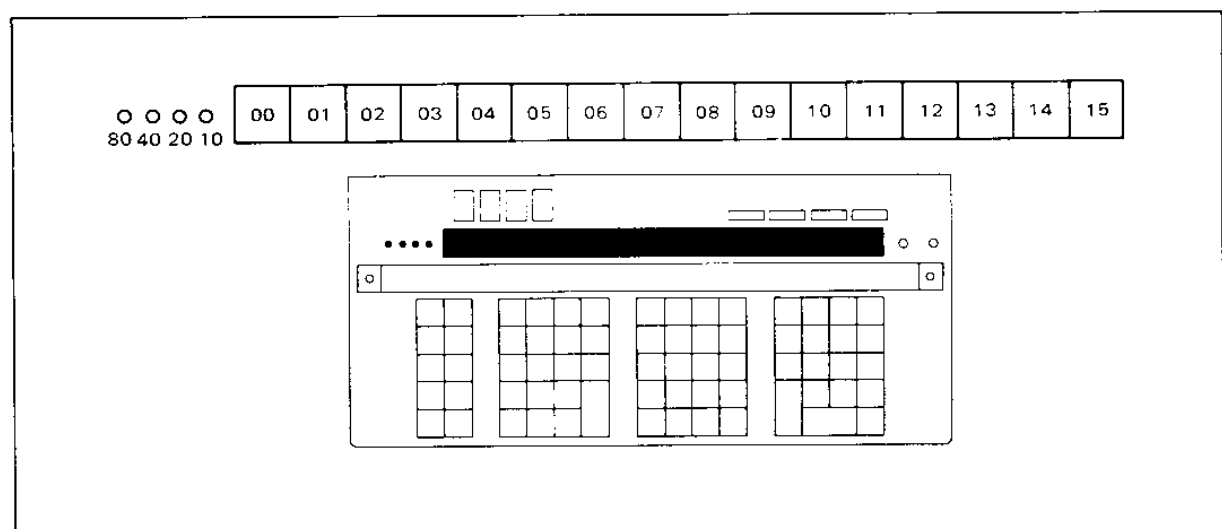
In addition to the X and Y work registers, the Wang 720 has up to 249 storage registers. Each register has a 12-digit mantissa with sign and a two-digit exponent with sign. The registers are numbered consecutively from 000 to 248 and can be addressed both directly and indirectly for maximum convenience. Numbers are stored from and recalled to the X-Register. Each register can be used to add, subtract, multiply and divide. Any number in storage can be exchanged or swapped with any number in the X-Register.

DIRECT ADDRESSING

Direct addressing of registers requires a two-step command. The first keystroke indicates the operation (i.e., to Store, Recall, Add, Subtract, Multiply, Divide, or Exchange). The second keystroke indicates the register in which the operation is to be performed. To store a number, simply index the control key **STORE DIRECT** followed by a second keystroke identifying the register number. For direct addressing of registers above 165 see Section VII.

TOGGLE SWITCHES AND SPECIAL FUNCTION KEYS

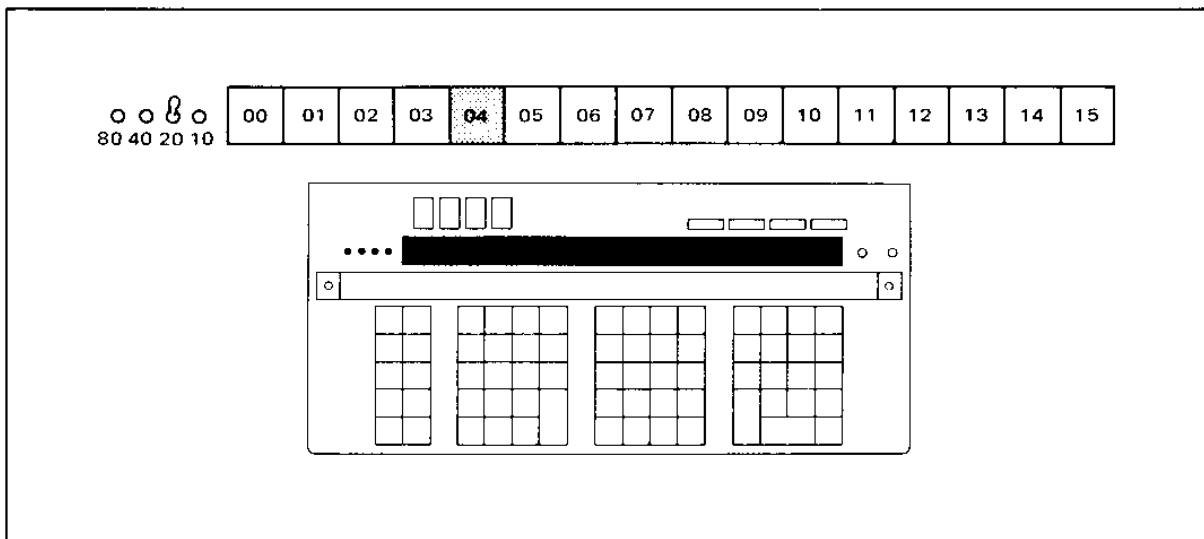
Each register is represented by a combination of toggle switch settings and special function keys.



Section II
Explanation of Keys

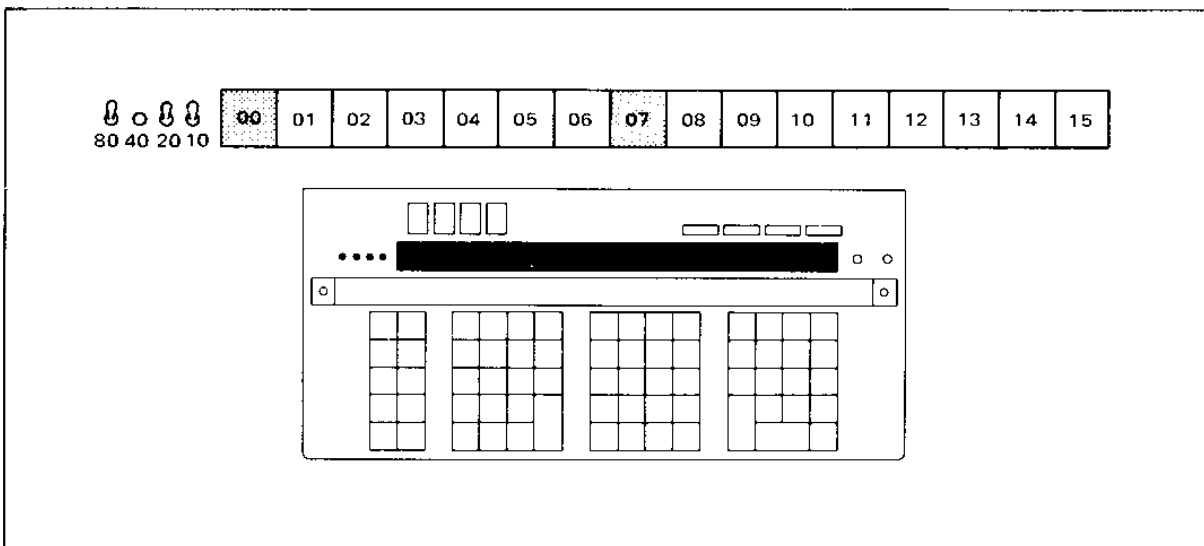
The toggle switches are set to the OFF (down) position. When the toggle switches are in the down position, the special function keys designate the registers 000 to 015. The 4 toggle switches are labeled 80, 40, 20 and 10. When one of these toggle switches is switched to the ON (up) position and a special function key is indexed, the register designated is the sum of the values of the toggle switches and the special function key.

(1)



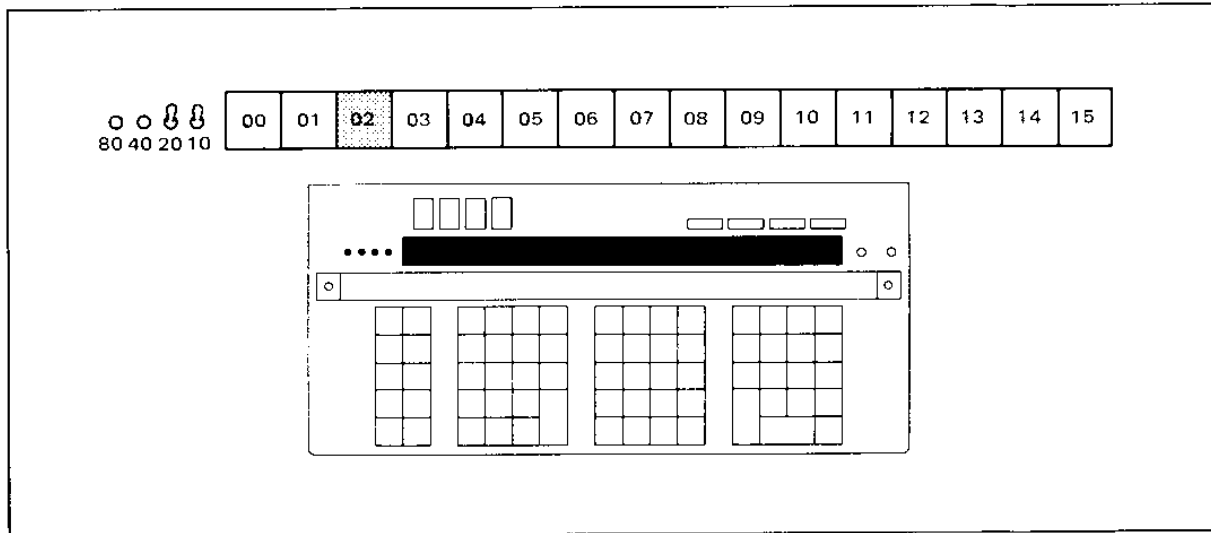
Depressing the special function key 04 while switch setting (20) is flicked ON designates register 24 (20 + 4).

(2)



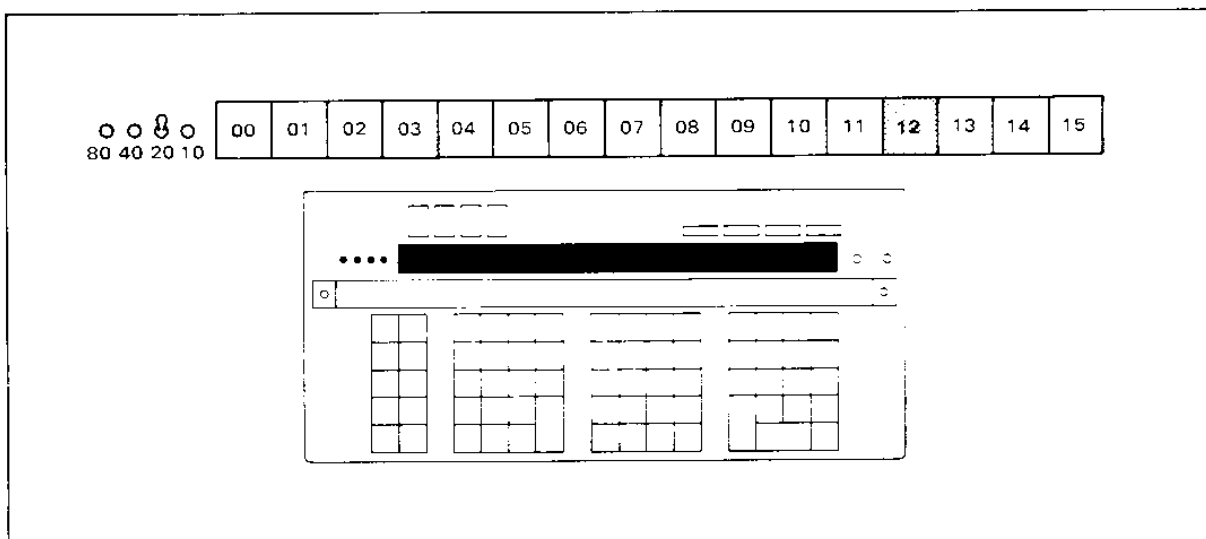
Notice the toggle switch setting. When the 07 key is indexed the register designated is 117 (80 + 20 + 10 + 7). When the 00 key is indexed the register designated is 110 (80 + 20 + 10 + 0).

(3) Designate Register 32 in two different ways:



One way of doing this would be to set toggle switches 20 and 10 to the ON (up) position and press the special function key 02.

Another way is the following:



Set toggle switch 20 in the up position and press the 12 key. Notice both combinations 20 + 10 + 2 and 20 + 12 designate register 32. Thus, different combinations of toggle switch settings and special function keys can be used to identify a particular register. However, in

Section II
Explanation of Keys

LEARN MODE the program code designating Register 32 would be 0302 or 0212, depending on which method was used.

STORE DIRECT

To store a number in a register, simply index the number into the X-Register, press the **STORE DIRECT** key followed by the register number.

EXAMPLE 1: Store π^2 into register 14
* Toggle switches down
Index $\pi \times^2$ STORE DIRECT 14
 π^2 is now stored in register 14 and is still displayed in the X-Register.

EXAMPLE 2: Store $.57 \times 10^{18}$ into Register 32
*Toggle switches 20 and 10 UP
Index 5 7 SETEXP 1 8
STORE DIRECT 02
 $.57 \times 10^{18}$ is now stored in Register 32 and is still displayed in X.

***NOTE**

For problems requiring less than 17 storage registers and for general usage, the toggle switches are kept in the OFF (down) position and the Special Function Keys are used to address Registers 000 to 015.

RECALL DIRECT

RECALL DIRECT recalls the number from the designated register into the X-Register. The number appears in the X-Register and also remains in the storage register. The sequence of steps to follow is the same as with **STORE DIRECT**.

EXAMPLE: Recall π^2 from register 14
Index RECALL DIRECT 14
 π^2 appears in the X-Register and is still in storage register 14.

↺ DIRECT

The **↺ DIRECT** key is a handy command which allows the operator to exchange a number in the X-Register with a number in any of the storage registers. The command simply swaps the values of the X-Register and the internal register. Again the sequence of steps to follow is **↺ DIRECT** followed by the desired register.

EXAMPLE: Suppose 27.8 is in the X-Register and π^2 is in Register 14. To store 27.8 in Register 14 and recall π^2 to the X-Register in one operation:
Index **↺ DIRECT 14**

What happens if the same operation is performed again?

ADD, SUBTRACT, MULTIPLY, AND DIVIDE DIRECT (The X and Y Registers Remain Unchanged.)

In addition to storing a 12-digit mantissa and a 2-digit exponent, the registers can be used as accumulators to add, subtract, multiply and divide. With each of these operations the result is stored in the designated register and the X-Register and Y-Register remain unchanged.

The four arithmetic operations are:

+DIRECT	Adds number in X-REGISTER to value stored in register designated by next keystroke. The X and Y Registers remain unchanged.
-DIRECT	Subtracts number in X-REGISTER from value stored in register designated by next keystroke. The X and Y Registers remain unchanged.
XDIRECT	Multiplies number in X-REGISTER by value stored in register designated by next keystroke. The X and Y Registers remain unchanged.
÷DIRECT	Divides number in X-REGISTER into number stored in register designated by next keystroke. The X and Y Registers remain unchanged.

A simple example will illustrate how each of these commands works.

EXAMPLE: Perform the following in Register 001	
$\frac{(13 \times 2) + 4}{3} - 3 = 7$	
1. <u>1</u> <u>3</u> STORE DIRECT <u>01</u>	Places 13 in Register 01 and the X-Register
2. <u>2</u> X DIRECT <u>01</u>	This sequence of steps places the product equal to 26 in Register 01 and 2 remains unchanged in the X-Register.
3. <u>4</u> + DIRECT <u>01</u>	Adds 4 to the Answer. 30 is now in Register 01, 4 is in X-Register.

Section II
Explanation of Keys

4. <u>3</u> <u>÷</u> <u>DIRECT</u> <u>01</u>	Divides result by 3 putting 10 in Register 01, 3 remains in X-Register.
5. <u>-</u> <u>DIRECT</u> <u>01</u>	Since 3 is in X when the command is given, 3 is subtracted from 10 putting 7 in Register 01, 3 in X-Register.
6. <u>RECALL DIRECT</u> <u>01</u>	Recalls final answer to X. = 7

The fact that the result is put in the storage register rather than the X-Register can be extremely useful if we are using a constant multiplier or divisor.

INDIRECT ADDRESSING

In addition to providing direct access to the internal storage registers, the Wang 720 offers an indirect mode of address. Both display registers are utilized for indirect addressing. The Y-Register designates the register being addressed. As with direct addressing, the X-Register is used as the work register. The command is performed on the number in X and the result is placed in the internal storage register.

Indirect addressing is a valuable programming tool for saving program steps, especially in repetitive matrix-type operations. Remember, indirect addressing requires only one step -- the operation itself. The register on which the operation is performed is identified by the number in Y.

INDIRECT KEYS

The indirect commands are identical to those used in direct addressing. They consist of the following:

KEY	OPERATION
<u>STORE INDIRECT</u>	Stores number in X into Register designated by number in Y.
<u>RECALL INDIRECT</u>	Recalls number to X from register designated in Y. Number also remains in register.
 <u>INDIRECT</u>	Swaps number in X with number in register designated by Y.
<u>+ INDIRECT</u>	Adds number in X to number in register designated in Y. The sum is placed in internal register. Number in X remains unchanged.

<u>- INDIRECT</u>	Subtracts number in X from number in register designated in Y. The difference is placed in internal register. Number in X remains unchanged.
<u>X INDIRECT</u>	Multiplies number in X by number in register designated in Y. The product is placed in internal register. Number in X remains unchanged.
<u>÷ INDIRECT</u>	Divides number in X into number in register designated in Y. The quotient is placed in internal register. Number in X remains unchanged.

The following example illustrates how each of these commands would be used.

Example

Perform the following in Register 002 using Indirect mode of address.

$$\left[\frac{7(5.8) - 7.2}{3} \right]^2 + 3 = 126.951111110$$

KEY	OPERATION
<u>2</u> ↑	Places the register number in Y (The register number is usually computed in the program)
<u>7</u> ST INDIR	Stores 7 in register 002. The value is now in both register 002 and the X Register.
<u>5</u> . <u>8</u> X INDIR	Multiplies the value (7) in Register 002 by 5.8, putting the result in 002 and 5.8 remaining in X.
<u>7</u> - <u>2</u> -- INDIR	Subtract 7.2 from the value in Register 002 and places result in Register 002. 7.2 remains in X.
<u>3</u> ÷ INDIR	Divides the value in Register 002 by 3. The result is put in Register 002 and 3 remains in X.
 INDIR	Exchanges $\frac{7(5.8) - 7.2}{3}$ in Register 002 with 3 in the X-Register.
<u>X</u> ²	Squares the value in X.

Section II
Explanation of Keys

<u>+ INDIR</u>	Adds $\left[\frac{7(5.8) - 7.2}{3} \right]^2$ to 3 in Register 002. The result is placed in Register 002 and $\left[\frac{7(5.8) - 7.2}{3} \right]^2$ remains in X.
<u>RECALL INDIR</u>	Recalls final answer = 126.95111110

ADVANTAGES OF INDIRECT ADDRESSING

Two advantages of the indirect mode of addressing are:

1. It requires only one keystroke to perform the indicated operation.
2. By constructing a loop, a given program step sequence can operate on many different sets of registers. A saving of many program steps can result from this technique.

The illustration below is a simple program which illustrates the advantage of indirect addressing. The program stored 0 in the first 100 registers. Using direct access a minimum of 200 steps would be required. (Two keystrokes per register – **STORE DIRECT** followed by each register number.) In contrast, this program requires only 13 steps to accomplish the same thing. A savings of 187 steps!

OPERATING PROCEDURE: <u>PRIME</u> <u>GO</u>		
STEP	KEY	CODE
0000	MARK	0408
0001	0700	0700
0002	0	0700
0003	ST IND	0504
0004	1	0701
0005	+	0600
0006	1	0701
0007	0	0700
0008	0	0700
0009	SKIP IF Y=X	0509
0010	SEARCH	0407
0011	0700	0700
0012	STOP	0515

RECALL RESIDUE

The **RECALL RESIDUE** key is a unique Wang 720 feature which can be of great value to users who need greater than 12 digit accuracy. The **RECALL RESIDUE** key gives the

user the option of double precision arithmetic for addition, subtraction, multiplication, and division performed in any of the storage registers or the X and Y registers. By indexing the **RECALL RESIDUE** key directly after performing one of these operations, another 12 digits of accuracy is acquired.

ADDITION, SUBTRACTION, MULTIPLICATION

When the **RECALL RESIDUE** key is indexed after performing an addition, subtraction or multiplication, a residue is displayed in the X-Register, which if added to the first 12 digits of the result, gives an additional 12 digits of accuracy. Examples are given to show how the **RECALL RESIDUE** key is used for addition, subtraction, and multiplication.

EXAMPLE 1:

ADD	OPERATION ON 720	DISPLAY
5024873058.28	<u>5 0 2 4 8 7 3 0 5 8 . 2 8</u>	(+5024873058.28 +6.8520987)
<u>+ 6.8520987</u>	<u>6 . 8 5 2 0 9 8 7</u>	
5024873065.1320987	±	(+502487306513 +10 +6.8520927)
	<u>RESIDUE</u>	(+502487306513 +10 +.209870000000 -02)

By indexing the **RECALL RESIDUE** key, the significant digits which would ordinarily be lost in the shifting process are retained. The final result is always the algebraic sum of the values displayed.

In subtraction, however, the residue might be opposite in sign to the answer. This should not cause any difficulty since the residue is always algebraically added to the result.

EXAMPLE 2:

SUBTRACT	OPERATION ON 720	DISPLAY
5024873058.28	<u>5 0 2 4 8 7 3 0 5 8 . 2 8</u>	(+5024873058.28 +6.8520987)
<u>- 6.8520987</u>	<u>6 . 8 5 2 0 9 8 7</u>	
5024873051.4279013	=	(+502487305143 +10 +6.8520987)

Section II
Explanation of Keys

	<u>RESIDUE</u>	+ .502487305143 +10 - .209870000000 -02
--	----------------	---

In this example, the residue is opposite in sign to the result. If these two numbers are added together, the correct result is generated. An easy way of performing this addition is to decrease the 12th digit of the result by 1 (.502487305143 becomes .50248730512), subtract each digit of the residue from 9 so .20987 becomes .79012, and add 1 to the last significant digit (.79013).

Multiplication works the same way as addition.

EXAMPLE 3:

MULTIPLY	OPERATION ON 720	DISPLAY
31415.9254998 .728645297326	$\begin{array}{r} 3\ 1\ 4\ 1\ 5\ .\ 9\ 2\ 5\ 4\ 9\ 9\ 8 \\ \uparrow \\ .\ 7\ 2\ 8\ 6\ 4\ 5\ 2\ 9\ 7\ 3\ 2\ 6 \end{array}$	+31415.9254998 + .728645297326
The answer is	$\times$	+22891.0663764 + .728645297326
22891.0663765732361535348	<u>RESIDUE</u>	+22891.0663765 + .732361535348 -07

The first twelve digits of the product are in Y; the last 12 digits are in X.

DIVISION

Using the **RECALL RESIDUE** key in division is slightly different from addition, subtraction, and multiplication. In division, indexing the **RECALL RESIDUE** key gives us a remainder. Using this remainder and the original divisor, 12 more digits of accuracy can be obtained by performing the division again. Study the following example illustrating the technique:

EXAMPLE 4:

DIVIDE $\frac{22}{7}$	OPERATION ON 720	DISPLAY
$\begin{array}{r} 3.14285714285 \\ 7 \overline{) 22.00000000} \\ \underline{21} \\ 10 \\ \underline{7} \\ 30 \\ \underline{28} \\ 20 \\ \underline{14} \\ 60 \\ \underline{56} \\ 40 \\ \underline{35} \\ 50 \\ \underline{49} \\ 10 \\ \underline{7} \\ 30 \\ \underline{28} \\ 20 \\ \underline{14} \\ 60 \\ \underline{56} \\ 40 \\ \underline{35} \\ \boxed{5} \end{array}$ Remainder $\rightarrow$ $\boxed{5}$	$\begin{array}{c} \underline{2} \underline{2} \\ \uparrow \\ \underline{7} \\ \div \\ \hline \text{RESIDUE} \end{array}$	+22.000000000 +7 +3.14285714285 +7.00000000000 +3.14285714285 +.500000000000 --11

The +.500000000000 -- 11 displayed in X after the **RECALL RESIDUE** key is pressed indicates a remainder of 5 after the first 12 digits of the quotient are generated. Notice the proper decimal position is retained (i.e., $.5 \times 10^{-11}$). Since the decimal position is retained automatically, the original divisor should be expressed with the decimal point in the left most position and an exponent value of 0 before performing the second division. Thus, .7 is divided into the remainder $.5 \times 10^{-11}$ and 12 more digits of the quotient are generated. To preserve the first 12 digits of the quotient, the second division is performed in Register 000.

Since the remainder is now in X

$$\frac{\text{STDIR } 00}{7 \text{ SETEXP*} \div \text{DIR } 00} \text{ RE DIR } 00$$

*This command automatically aligns the decimal point and exponential value of the divisor.

Read .714285714285 – 11 in the X-Register which if added to 3.14285714285 yields 24 digit accuracy for 22/7. If greater accuracy is desired, simply touch the **RECALL RESIDUE** key to obtain the remainder (.5000000000000 – 23) and repeat the process.

This example illustrates the fact that the **RECALL RESIDUE** key performs the same function when any of the 249 internal registers are used to add, subtract, multiply and divide. The **RECALL RESIDUE** key is NOT limited to use solely with the X and Y registers. **IT SHOULD ALSO BE NOTED THAT THE RESIDUE MUST BE SAVED AFTER EACH OPERATION IF IT IS TO BE USED IN FURTHER CALCULATIONS.**

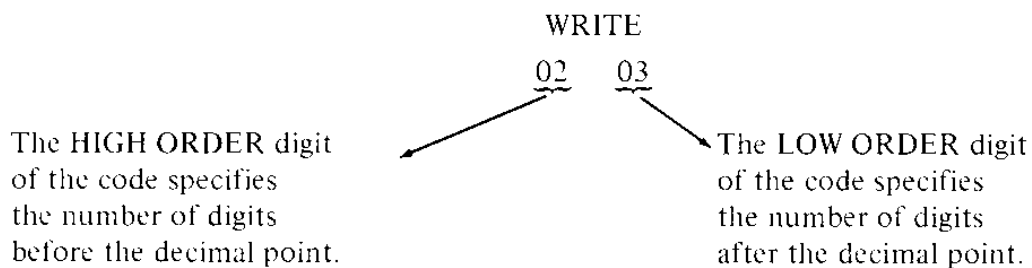
WRITE COMMANDS

The 701 Output Writer provides the user with completely formatted alpha-numeric output of his calculated results.

NUMERIC output consists of a two-step command. The WRITE key followed by a format command will print the contents of the X-Register.

The format command specifies the number of digits to be printed out before and after the decimal point

EXAMPLE



The above command would print two digits before the decimal point and three digits after the decimal point.

An option to always print in modified scientific notation is available.

EXAMPLE

Display:	+12345678123 – 40
Command:	WRITE
	0015
Output will appear as:	.123456789123ex – 40

ALPHABETIC output can be printed under program control by using the **WRITE ALPHA** command. Indexing the **WRITE ALPHA** key places the 720 in alpha mode so that alpha characters can be printed. The **END ALPHA** command takes the 720 out of alpha mode.

EXAMPLE

WRITE ALPHA	(Places 720 in alpha mode)
H – 0201	
E – 0205	
L – 0209	
L – 0209	
O – 0109	
END ALPHA	(Takes the 720 out of alpha mode)

The above example would print the word “**HELLO.**”

Other control commands such as shifting to upper and lower case, carriage return, line feed, spacing, backspace, and tabulation are all available on the Output Writer. All these features are discussed in the 701 OUTPUT WRITER MANUAL.

GROUP 1—GROUP 2

These two keys are reserved for addressing optional peripheral equipment.

SECTION III PROGRAMMING

CODING

All programmed operations are represented by a 4-digit code. A list of the keyboard operations and their respective codes is given below:

720 PROGRAM CODES

CODE	KEY	CODE	KEY	CODE	KEY
0400	+ DIRECT	0506	 INDIR	0612	$\sqrt{x}$
0401	- DIRECT	0507	SKIP if $Y \geq X$	0613	10^x
0402	x DIRECT	0508	SKIP if $Y < X$	0614	e^x
0403	$\div$ DIRECT	0509	SKIP if $Y = X$	0615	$1/x$
0404	STORE DIRECT	0510	SKIP if ERROR	0700	0
0405	RECALL DIRECT	0511	RETURN	0701	1
0406	 DIRECT	0512	END PROG	0702	2
0407	SEARCH	0513	LOAD PROG	0703	3
0408	MARK	0514	GO	0704	4
0409	GROUP 1	0515	STOP	0705	5
0410	GROUP 2			0706	6
0411	WRITE	0600	+	0707	7
0412	WRITE ALPHA	0601	-	0708	8
0413	END ALPHA	0602	x	0709	9
0414	STORE Y *	0603	$\div$	0710	SET EXP
0415	RECALL Y *	0604	$\uparrow$	0711	CHANGE SIGN
		0605	$\downarrow$	0712	DECIMAL POINT
0500	+ INDIR	0606		0713	x^2
0501	- INDIR	0607	X	0714	RECALL RESIDUE
0502	x INDIR	0608	INTEGER X	0715	CLEAR X
0503	$\div$ INDIR	0609	π		
0504	STORE INDIR	0610	$\log_{10} X$		*ENTERED BY TOGGLE
0505	RECALL INDIR	0611	$\log_e X$		SWITCH SETTING

FOR MODEL 720 ONLY

*CODE	OPERATION	*CODE	OPERATION	*CODE	OPERATION
1200	+ DIRECT (+100)	1203	$\div$ DIRECT (+100)	1206	 DIRECT (+100)
1201	- DIRECT (+100)	1204	STORE DIRECT (+100)	1214	STORE Y (+100)
1202	x DIRECT (+100)	1205	RECALL DIRECT (+100)	1215	RECALL Y (+100)

Any of these codes automatically adds 100 to the Storage Register number.

*These codes are generated by toggle switches and special operation keys.

SPECIAL COMMANDS which must be preceded by WRITE ALPHA

(Decimal Point Shifting Commands)

CODE	KEY	OPERATION	CODE	KEY	OPERATION
0401	-DIRECT	Divide X by 10^1	0701	1	Multiply X by 10^1
0402	X DIRECT	Divide X by 10^2	0702	2	Multiply X by 10^2
0403	÷ DIRECT	Divide X by 10^3	0703	3	Multiply X by 10^3
0404	STORE DIRECT	Divide X by 10^4	0704	4	Multiply X by 10^4
0405	RECALL DIRECT	Divide X by 10^5	0705	5	Multiply X by 10^5
0406	↔ DIRECT	Divide X by 10^6	0706	6	Multiply X by 10^6
0407	SEARCH	Divide X by 10^7	0707	7	Multiply X by 10^7
0408	MARK	Divide X by 10^8	0708	8	Multiply X by 10^8
0409	GROUP 1	Divide X by 10^9	0709	9	Multiply X by 10^9
0400	+ DIRECT	Divide X by 10^{10}	0700	0	Multiply X by 10^{10}

DECISIONS

CODE	KEY	OPERATION
0410	GROUP 2	Skip if Y positive
0411	WRITE	Skip if Y = 0
0510	SKIP if ERROR	Skip if Y negative
0511	RETURN	Skip if Y ≠ 0
0610	$\log_{10} X$	Skip if X positive
0611	$\log_e X$	Skip if X = 0

DECISIONS

CODE	KEY	OPERATION
0710	SET EXP	Skip if X negative
0711	CHANGE SIGN	Skip if X ≠ 0
(Additional Commands)		
0615	1/X	Pause
0514	GO	$180/\pi$
0515	STOP	$\pi/180$

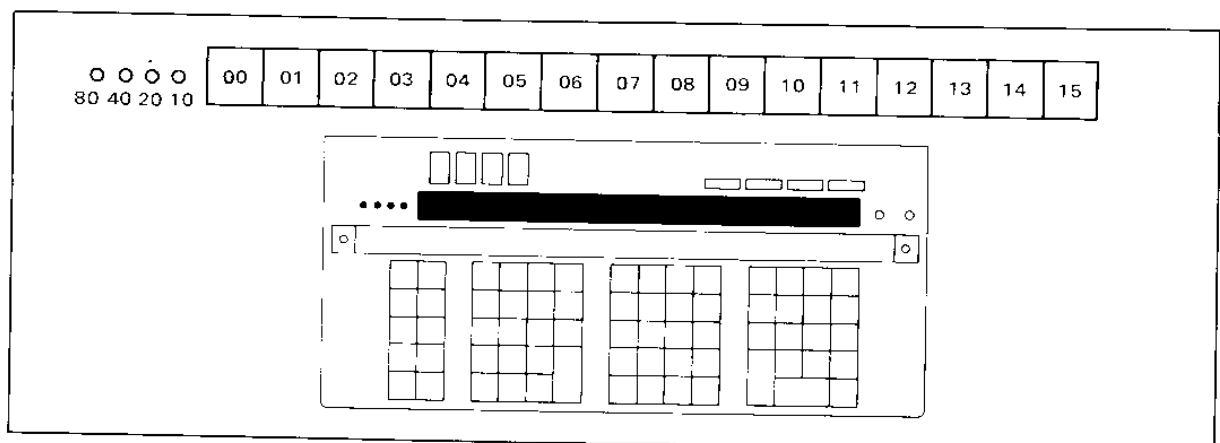
The four-digit code consists of 2 halves: a high-order 2-digit number and a low-order 2-digit number.

HIGH ORDER { X X X X } LOW ORDER

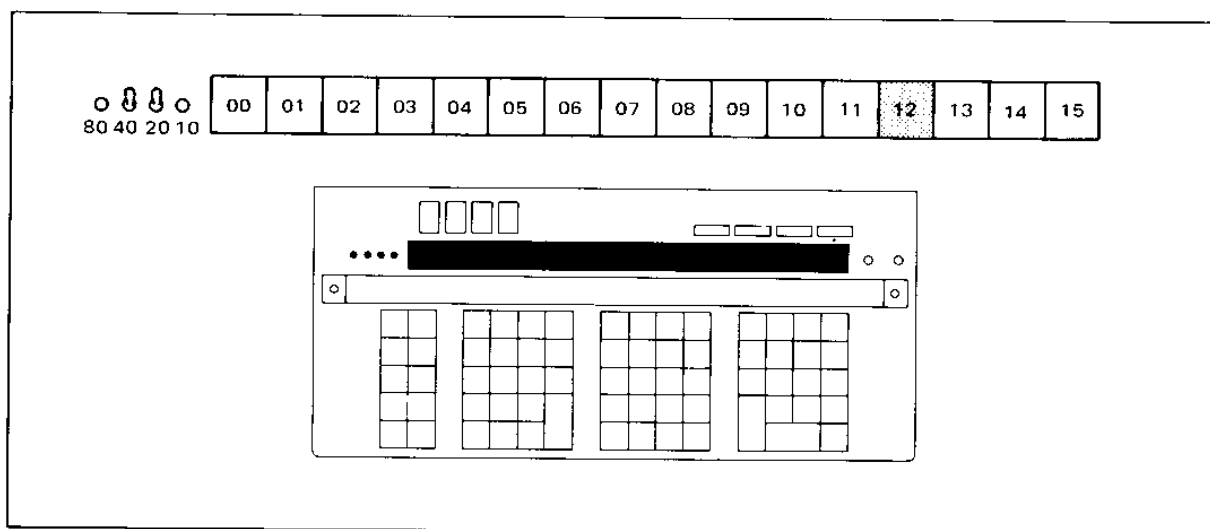
Each of these halves can assume the values 00, 01, 02, ... up to 15. Thus there are 16 different high and low-order digits and a total of $16 \times 16 = 256$ codes.

GENERATING A CODE USING SPECIAL FUNCTION KEYS AND TOGGLE SWITCHES

While this procedure is not recommended for any of the "operation keys," any legal code can be generated using the toggle switches and the special function keys. The special function keys are used to define the low-order digit and a combination of toggle switches is used to define the high order digit.



The toggle switches are labeled 80, 40, 20, and 10 for convenience in selecting the data storage registers discussed in Section II. **THEY CAN ALSO BE VISUALIZED AS REPRESENTING THE NUMBERS 08, 04, 02, AND 01 FOR THE PURPOSE OF GENERATING THE HIGH-ORDER DIGIT OF ANY LEGAL CODE.** When a special function key is indexed, the operation executed by the calculator is the command whose high-order digit is defined by the setting of the toggle switches and whose low-order digit is the special function key indexed.



If the toggle switches are set as in the above figure and the special operation key 12 is indexed, the square root of the number in the X-Register will be generated since the code for square root is 0612. Naturally, the square root of a number would rarely be found using this technique, however, this example is included to explain how to generate any of the 256 codes. This technique is used most often with the Store Y and Recall Y commands.

CORE MEMORY

Core Memory is organized into 249 data registers numbered consecutively from 000 to 248. Registers 000 – 247 are used for storing either program steps or data. 16 program steps occupy 2 data-storage registers. Register 248 is used exclusively for data storage.

As stated previously, each programmed operation is represented by a four-digit code. The four-digit code consists of two halves: a high-order two-digit number and a low-order two-digit number.

<u>XX</u>	<u>XX</u>
HIGH	LOW
ORDER	ORDER

The program code for $\sqrt{x}$ is

<u>06</u>	<u>12</u>
HIGH	LOW
ORDER	ORDER

[illegible]

A program code step occupies two digits of storage, one digit in each of two adjacent registers; the high-order digit of a code occupying one register; the low-order digit the other register.

Program steps 0000 to 0015 occupy registers 246 and 247. The following routine to add $2 + 2$ is loaded into Registers 246 and 247 as illustrated.

STEP	KEY	CODE
0000	2	0702
0001	↑	0604
0002	+	0600
0003	STOP	0515
0004	END PROG	0512

FIGURE 1

Registers

246											05	05	06	06	07	high order
247											12	15	00	04	02	low order
0015 0014 0013 0012 0011 0010 0009 0008 0007 0006 0005 0004 0003 0002 0001 0000																

PROGRAM STEP NUMBERS

The high-order digit of the program code is loaded into register 246, the low-order digit of the code in register 247. Each pair of registers can accommodate 16 program steps. The program steps are numbered 0000 to 1983. Step 0000 is located in data registers 247 and 246, step 1983 is located in data registers 000 and 001. (See page 3-4) It shows exactly what program steps are located in each register. It is advisable to use registers 000, 001, 002, 003, etc., for data storage and registers 247, 246, 245, etc., for program storage. In this way, data will be stored in one end of core and program operations will be stored in the opposite end of core.

NUMBER OF REGISTERS OCCUPIED BY A PROGRAM

If a program is 7 steps long, 2 data registers are being utilized for storing the program. If the program is 35 steps long, 6 data registers are being used for program steps. To determine how many registers are being utilized:

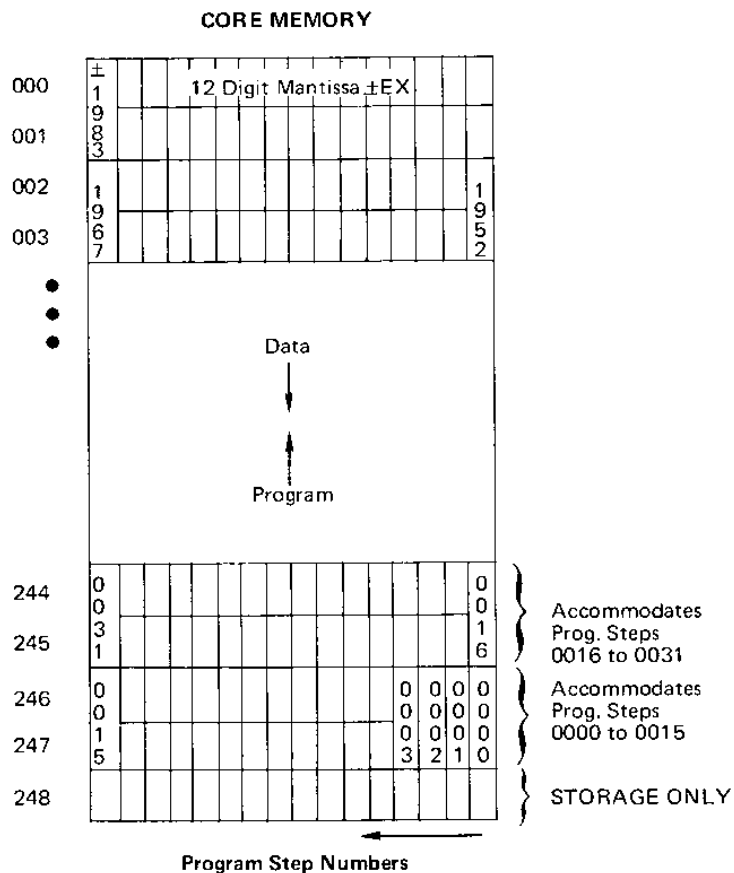
1. Divide the number of program steps by 16.
2. Round the answer to the next whole number.

Example:

$$\frac{33}{16} = 2.0625 \text{ becomes } 3$$

3. Multiply the whole number by 2 to find the equivalent number of registers being used.

Section III
Programming



Example

Program of 88 steps occupies 12 registers

$$\frac{88}{16} = 5.5 \rightarrow 6(2) = 12 \text{ registers}$$

See Section V, Page 5-4 (for short program to perform this calculation.)

SECTION IV PROGRAMMING CONCEPTS

PROGRAMMING CONCEPTS

To exploit the full programming capability of the 720, a few basic programming concepts must be discussed. These are the concept of a branch, a subroutine, a loop and a decision.

Usually, the main part of a program advances one step at a time in a linear and continuous fashion. Each operation is performed consecutively one after the other. A program to evaluate the polynomial $y = 20x^2 + 5x + 7.2$ for different values of x would be:

OPERATION	CODE	REMARK
ST DIR	0404	Key in x
REG 00	0000	
x^2	0713	
↑	0604	
2	0702	
0	0700	$20x^2$ in y
x	0602	
5	0705	
X DIR	0402	
REG 00	0000	
RE DIR	0405	$20x^2 + 5x$ in y
REG 00	0000	
+	0600	
7	0707	
.	0712	
2	0702	$20x^2 + 5x + 7.2$ in y
+	0600	

Notice "the program" is simply the steps the user would perform if he were doing the calculation manually on the keyboard. However, the 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.

Section IV
Programming Concepts

MARK AND SEARCH COMMANDS

Flags (names or marks) in a program are set by the **MARK** key followed by a second keystroke. To set a flag requires 2 keystrokes: **MARK** followed by any of the 256 legal codes. Thus, there are 256 different "names" or flags which can be used in a Wang 720 program. For the simple program we have written to evaluate $Y = 20x^2 + 5x + 7.2$, the number key **1** is used as a distinguishing flag. Thus, the program is preceded by the 2 keystrokes **MARK 1**. To end the calculation simply add a **STOP** command. The complete program thus becomes:

OPERATION	CODE	REMARK
MARK	0408	
1	0701	
ST DIR	0404	
REG 000	0000	
x^2	0713	
↑	0604	
2	0702	
0	0700	
x	0602	
5	0705	
X DIR	0402	
REG 00	0000	
RE DIR	0405	
REG 00	0000	
+	0600	
7	0707	
	0712	
2	0702	
+	0600	
STOP	0515	

Flags tell the 720 where to start its calculations. They indicate the destination of a **SEARCH** command. In the **SEARCH** command, 2 keystrokes are required: **SEARCH** followed by a second keystroke which identifies the flag or mark to find. Thus, the operating procedure for the above program would be:

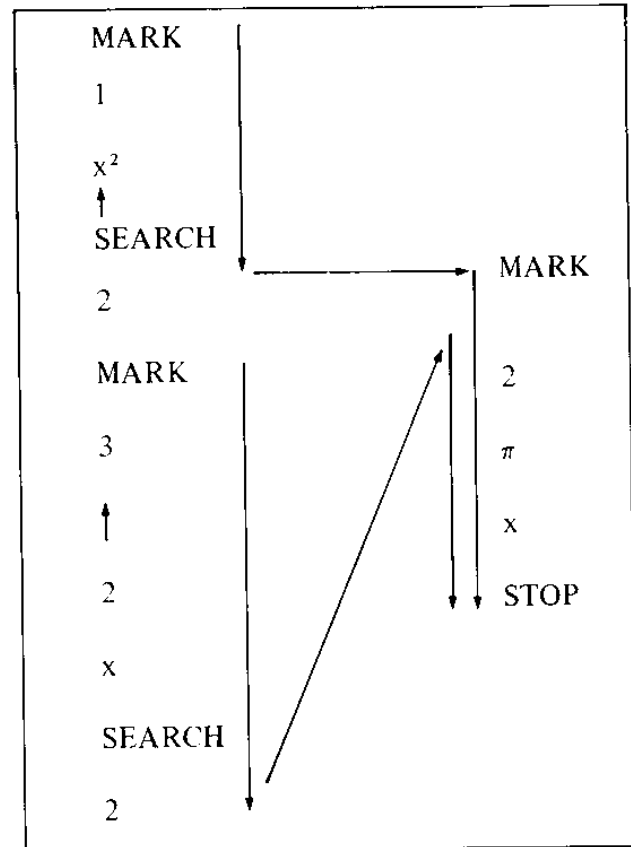
Key X; **SEARCH 1**

and the operations between **MARK 1** and the **STOP** command would be executed in sequence.

Generally, the numeric keys 0, 1, 2, . . . 9 are used as flags or markers for starting general programs. However, any programmable key on the 720 keyboard can be used as a name or marker. A program can start with a **MARK e^x** and to locate this mark, simply **SEARCH e^x** . It should be clear that when the **MARK** and **SEARCH** keys are indexed the calculator interprets the next keystroke as a name or flag and **not** as any other type of operation. When a **SEARCH X** command is given the 720 searches through core to locate the designated marker. If on scanning core it doesn't find the mark, the program stops and the **PROGRAM ERROR INDICATOR** goes on indicating there is no such mark in core.

BRANCHING

The program for evaluating the polynomial follows a linear sequence of steps. The program executes step 0000 then 0001, then 0002, and so on through to the last step. However, the Wang 720 does not have to follow a linear sequence of steps. It is possible for the 720 to start executing commands from step 0025 and go through to step 0052, then jump to step 0075 ignoring all the commands between steps 0052 and 0075. To break out of a linear sequence of steps and to jump about in a program is called branching. Both conditional and unconditional branching are possible on the 720. The **SEARCH** and **MARK** commands are used respectively for branching and for defining the destination of a branch.



The above program evaluates $A = \pi r^2$ or $C = 2\pi r$, depending on which steps are executed in the program.

A. To find $A = \pi r^2$: Index r **SEARCH 1**

The program starts by squaring r and putting the result in Y , it then branches to **MARK 2** ignoring all commands until it encounters the designated flag, and then multiplies r^2 by π for the final result in Y .

B. To find $C = 2\pi r$: Index r **SEARCH 3**

The program ignores the commands preceding **MARK 3** and starts by putting r in Y and multiplying it by 2. It then branches to **MARK 2** ignoring all commands until it encounters the designated flag, and then multiplies $2r$ by π for the final result in Y .

As this program demonstrates, the **SEARCH** command can be part of a program, or can be keyed in by the operator, or both. In either case, upon encountering this command the program branches immediately to the designated mark. **MARK** and **SEARCH** commands can be located at any point or step in the program.

LOOPING

	MARK	OPERATING INSTRUCTIONS
Initialization	1	
	0	
	↑	1. <u>PRIME</u> : <u>SEARCH</u> <u>1</u>
	ST DIR	2. Key X, <u>GO</u>
Loop	REG 00	Repeat 2 for all x
	ST DIR	3. <u>SEARCH</u> <u>2</u>
	REG 01	Read Σx in X
	MARK	Read N in Y
Results	0808	4. <u>GO</u>
	STOP	Read Σx^2 in X
	+ DIR	
	REG 00	
	Σx	
	x^2	
	+ DIR	
	REG 01	
	Σx^2	
	1	
	+	n
	SEARCH	
	0808	
	MARK	
	2	
	RE DIR	
	REG 00	
	STOP	Σx
	RE DIR	
	REG 01	
	STOP	Σx^2

The above program further illustrates the idea of branching and introduces the important concept of looping. The program computes the statistical sums; Σx and Σx^2 for any number of x values. The first set of instructions initializes the registers by storing 0 in Y, Register 000, and Register 001. The second part of the program forms a loop which accumulates the Σx in Register 000, Σx^2 in Register 001, and the number of points entered in the Y-Register. The same operations are performed on each x -value. The program exits from the loop when a **SEARCH 2** command is given. The final set of instructions recalls the answers to the display.

SUBROUTINE

Another idea closely related to branching is the concept of a subroutine. A subroutine is a part of a program (a sub-program) which appears several times within the overall program. Subroutine capability allows the program to branch to a specified routine, perform the calculations, and then return from where the program originally branched.

On the 720, a single keystroke is needed to branch to a subroutine. A set of 64 operation codes is reserved for this purpose. They consist of the 64 combinations which occur when the high-order digit of the 4 digit code assumes the values 00, 01, 02, and 03. A complete list of these codes is given below.

0000	0100	0200	0300
0001	0101	0201	0301
0002	0102	0202	0302
0003	0103	0203	0303
0004	0104	0204	0304
0005	0105	0205	0305
0006	0106	0206	0306
0007	0107	0207	0307
0008	0108	0208	0308
0009	0109	0209	0309
0010	0110	0210	0310
0011	0111	0211	0311
0012	0112	0212	0312
0013	0113	0213	0313
0014	0114	0214	0314
0015	0115	0215	0315

EXAMPLE: Calculate the following for Z

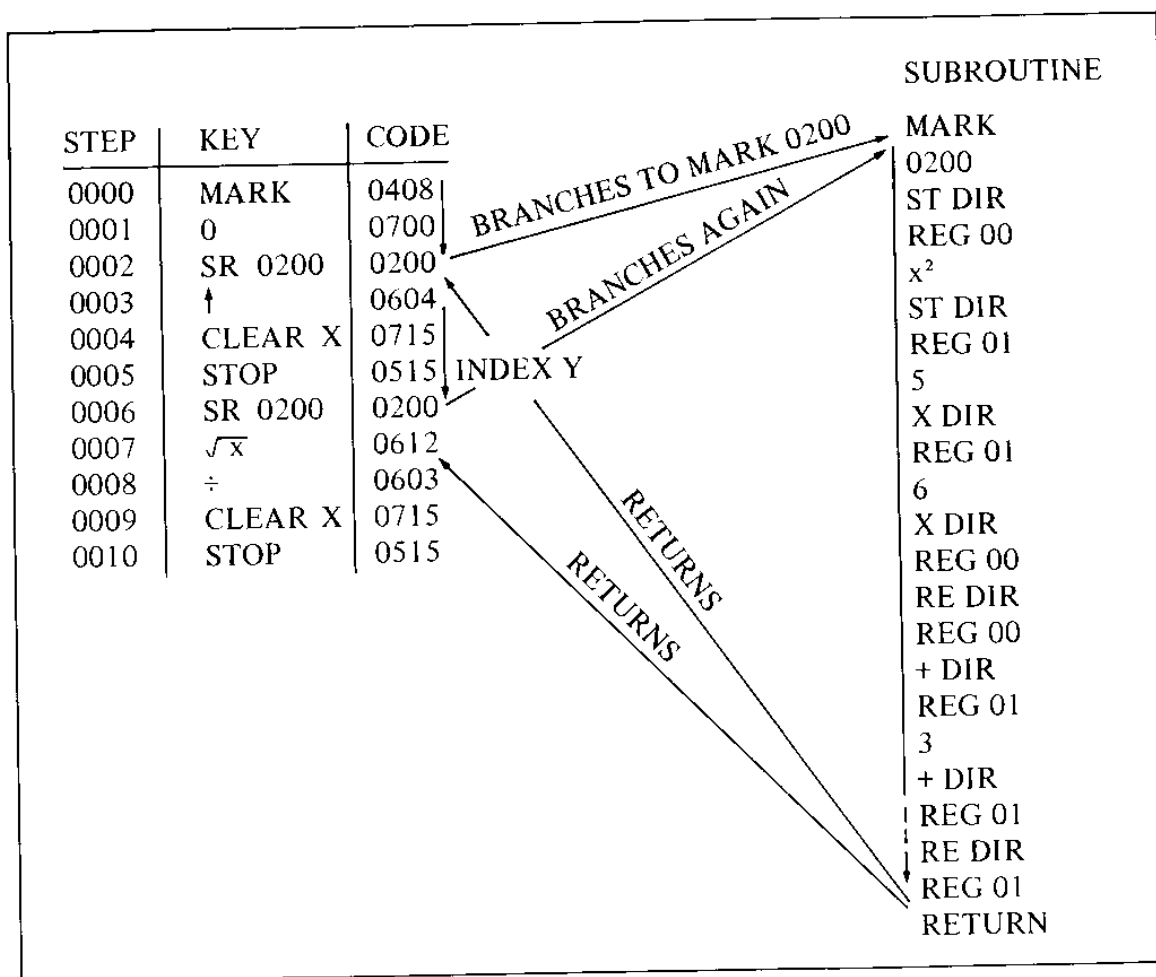
$$Z = \frac{5x^2 + 6x + 3}{\sqrt{5y^2 + 6y + 3}}$$

OPERATING INSTRUCTIONS:

INDEX X SEARCH 0

Y GO

Read Z in Y



When the first 0200 command is encountered at step 0002, the program branches to **MARK 0200**. At the **RETURN** command, the program branches back to step 0003 and continues on with the program. At the second 0200 command, the program again branches to the subroutine defined by **MARK 0200**. However, at the **RETURN** command the program branches back to step 0007. There is no limit to the number of times a subroutine can be addressed and executed. The SR preceding the command in the KEY column is simply a mnemonic device indicating to the reader that a subroutine is being addressed.

It should be noted that the subroutine addressed through one of the 64 designated codes in Table 1 is preceded by a **MARK XXXX** of that same code and terminated by a **RETURN** command; otherwise, the calculator will not know when to return to the spot from which it originally branched.

The 64 codes listed on Page 4-5 do not necessarily have to define a subroutine. They can be used as regular marks and would then be addressed by the 2-step command **SEARCH XXXX**. However, it is generally considered wiser to reserve these codes exclusively for defining subroutines.

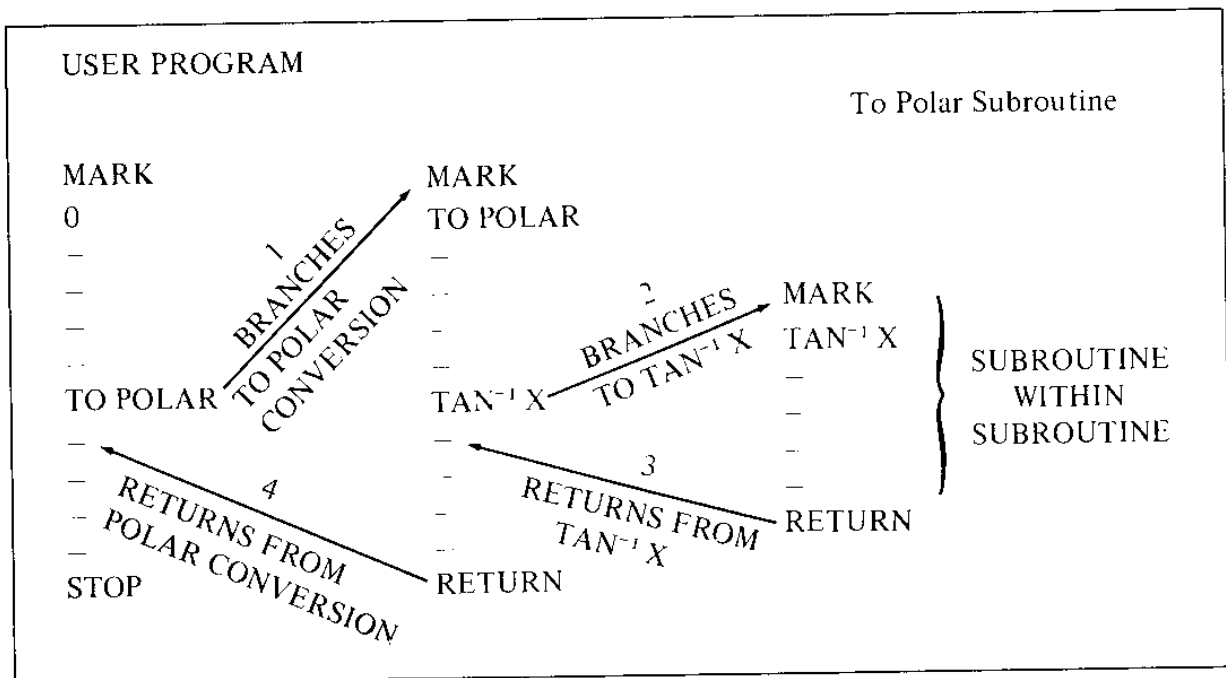
MULTI-LEVEL SUBROUTINES (Or a Subroutine within a Subroutine)

On the WANG 720, multi-level subroutines are possible. What does this mean? An example will best illustrate this concept. In the polar conversion in the TRIG PACK, the following formula is used to find

$$\theta = \tan^{-1} \frac{y}{x}$$

Therefore, the polar conversion subroutine addresses the $\tan^{-1} X$ subroutine. This means the 720 must remember what step to branch back to after each of the two **RETURN** commands is executed.

The 720 is capable of remembering 5 return addresses, thus it has a five level subroutine capability. Figure 2 illustrates this concept graphically. The program branches to subroutine, it immediately branches to this routine, executes it and on encountering the **RETURN** command, branches back to the polar conversion routine which it continues to execute. When the second **RETURN** command is encountered, control branches back to the main program and the remaining steps are executed.



SECTION V DECISION COMMANDS

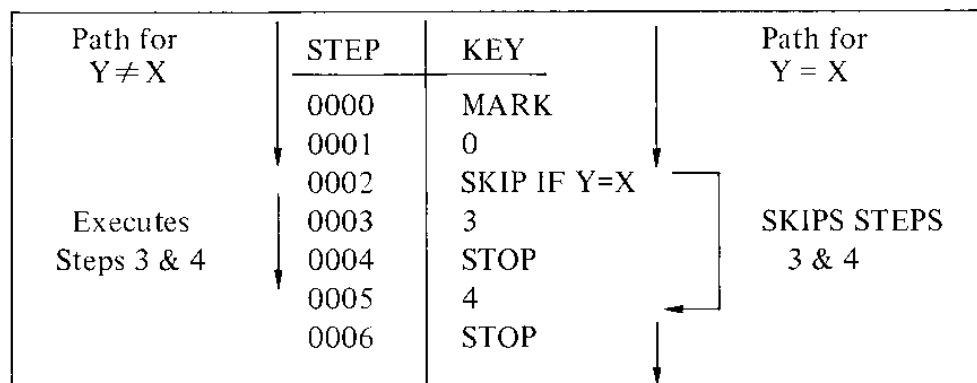
DECISIONS

The Wang 720 has four decisions it can perform. They are used to check for the existence of certain conditions. If the condition is met, the program skips the next two steps. If the condition is not met, the program executes the next step. The four commands are **SKIP IF Y = X**, **SKIP IF Y ≥ X**, **SKIP if Y < X**, and **SKIP IF ERROR**.

(1) Skip if Y = X

This command checks to see if the value in the Y-Register and X-Register are equal. If Y = X the program skips the next two steps. If Y does not equal X the program continues with the next step.

As a simple example:



This program will put 3 in the X-Register if Y is not equal to X, and a 4 in X if Y = X.

NOTE

In testing for the condition Y = X, the programmer should keep in mind the necessity for absolute equality of the numbers in X and Y. A condition which is not ordinarily found in analytical computations. Discrepancies often occur between the true value and the calculated value of a number.

Illustration:

$$\text{Calculate } Y = \left[\frac{1}{3} \right] \times 3 \neq 1$$

Section V
Decision Commands

If a 1 is placed in X and the command **SKIP IF Y = X** is given, the calculator will treat the numbers as being unequal. Any good book on numerical analysis gives a full discussion on these discrepancies which occur in approximation theory.

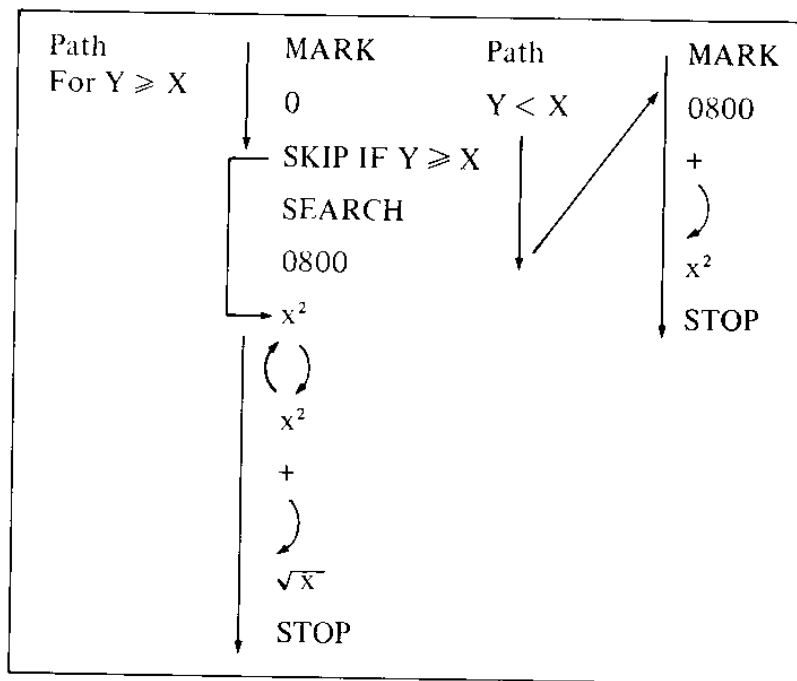
(2) Skip if $Y \geq X$

This command checks to see if the value in the Y-Register is equal to or greater than the value in the X-Register.

In the program below:

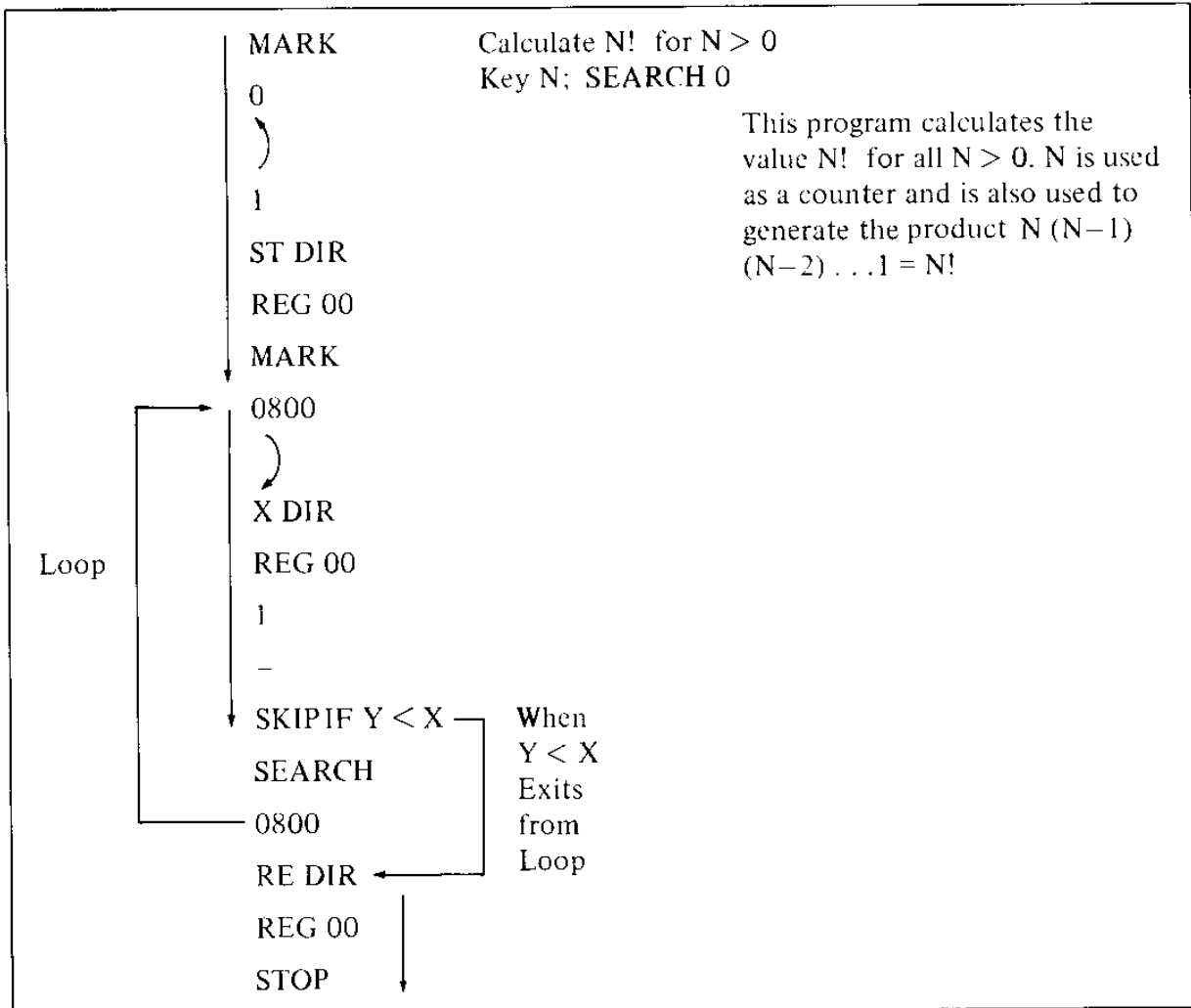
If $Y \geq X$ the value $\sqrt{Y^2 + X^2}$ is calculated.

If $Y < X$ the value $(Y + X)^2$ is calculated.



(3) Skip if $Y < X$

If the value in Y is less than the value in X, the program skips the next two steps. If the value in Y is equal to or greater than X, the next program step is executed.

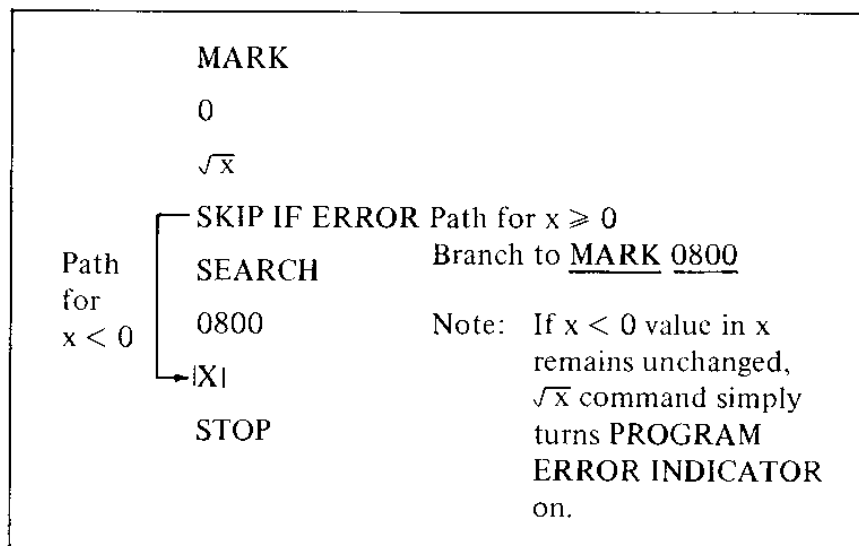


(4) Skip if Error

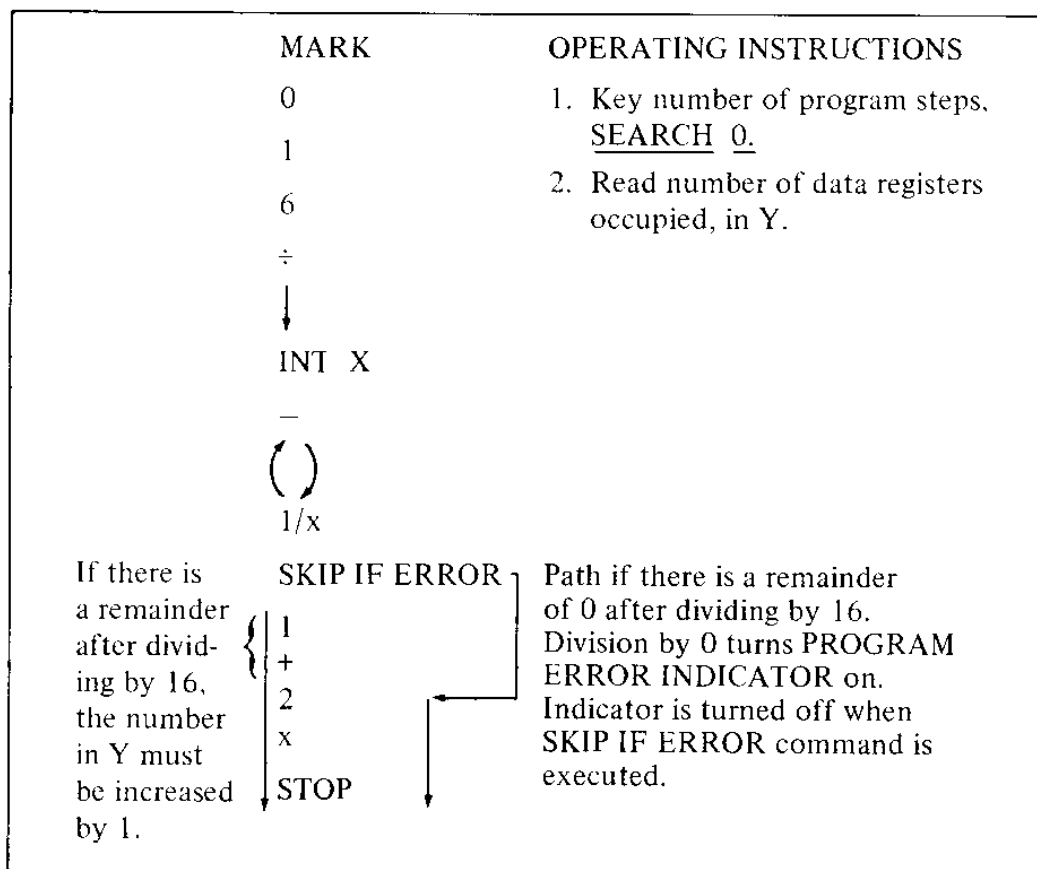
The final decision command **SKIP IF ERROR** can be used in a variety of ways to check for certain conditions (see page 2-8 to review what operations turn the **PROGRAM ERROR INDICATOR** on). Testing for these conditions turns the program indicator off. The following program distinguishes between positive and negative numbers. If the number in X is positive, the program will branch to **MARK 0800**. If it is negative, the program finds the $|X|$ and stops.

(For additional decision commands see Section VII.)

Section V
Decision Commands



The following program uses the **SKIP IF ERROR** command to calculate the number of data registers a program occupies on the Wang 720. It also illustrates the **INTEGER X** command.



PROGRAMMING TECHNIQUES

Looping Using a Counter

Looping is an important programming tool. The decision commands are most frequently

used to set up loops within programs. Counters are set up to "count" the number of times a calculation is performed.

Suppose the sum $Y = x + x^2 + x^3 + \dots + x^n$ is to be calculated for various values of x . The program below sets up a loop to calculate this sum for any number of terms. The value of n determines how many terms in the sum will be calculated.

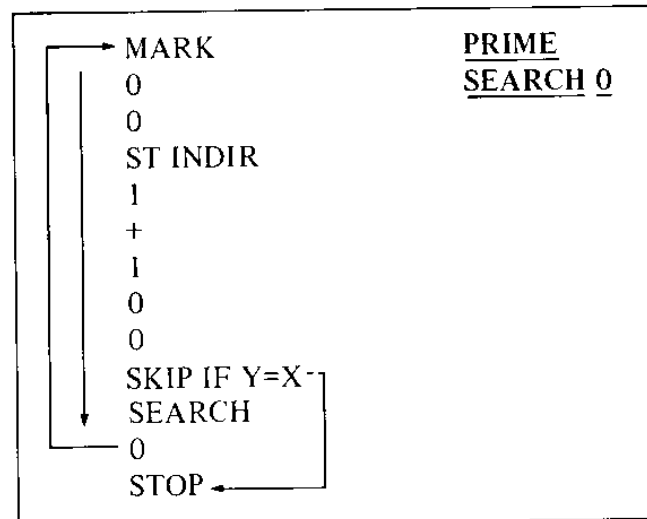
MARK	Key X	OPERATING INSTRUCTIONS
0		Key x <u>SEARCH 0</u>
ST DIR	Initializes	Key N <u>GO</u>
REG 00	Registers	Read x in Y-Register
1		Σx^n in X-Register
ST DIR		To understand the method used
REG 01		in the program, Rewrite the
0		sum as $Y = T_0 + T_1 + T_2 + \dots + T_n$
ST DIR		where $T_0 = x$
REG 02		$T_1 = xT_0$
STOP		$T_2 = xT_1$
↑	Key n	$T_n = x T_{n-1}$
MARK	counter in Y	The program starts the sum with x
0800		and uses the recursive formula
RE DIR	x	$x(x^{n-1})$ to calculate each successive
REG 00		term. N is used as a counter. The
X DIR	$x(x^{n-1})$	program performs the same calculation
REG 01		n times. Each time the loop is executed
RE DIR		the counter is decreased by 1. When
REG 01		N = 0 the loop is terminated and the
+ DIR		final sum is displayed.
REG 02	Σx^n	
1	Decrease N	
-	by 1	
SKIP IF Y < X	Exits from	
SEARCH	loop when	
0800	N = 0	
RE DIR		
REG 00		
RE DIR		
REG 02		
STOP		

Loop is performed n times

Calculates x^n and Σx^n

Section V
Decision Commands

Another slightly different counter is found in the program which stores 0's in the first 100 storage registers. In this example, the counter is constantly increasing until it reaches a value of 100. It also serves to designate the storage register being addressed.



Looping Without a Counter

Often there is no way to predetermine exactly the number of times a loop is to be performed. Other criteria have to be used.

EXAMPLE

In calculating the following sum for $X > 1$

$$1 + \frac{1}{x^2} + \frac{1}{x^4} + \frac{1}{x^6} + \frac{1}{x^8} + \dots + \frac{1}{x^{2n}}$$

for specified accuracy the number of terms to be calculated depends on the value of x . However, it is obvious that each successive term gets smaller and smaller and eventually approaches zero. If 12 digits of accuracy are needed, the calculation can be carried out until the last term gets so small that it does not materially affect the overall sum. This occurs when the term becomes smaller than 10^{-11} and the overall sum of the 720 no longer changes its value when a term is added to it.

To write the program, it is convenient to rewrite the series as

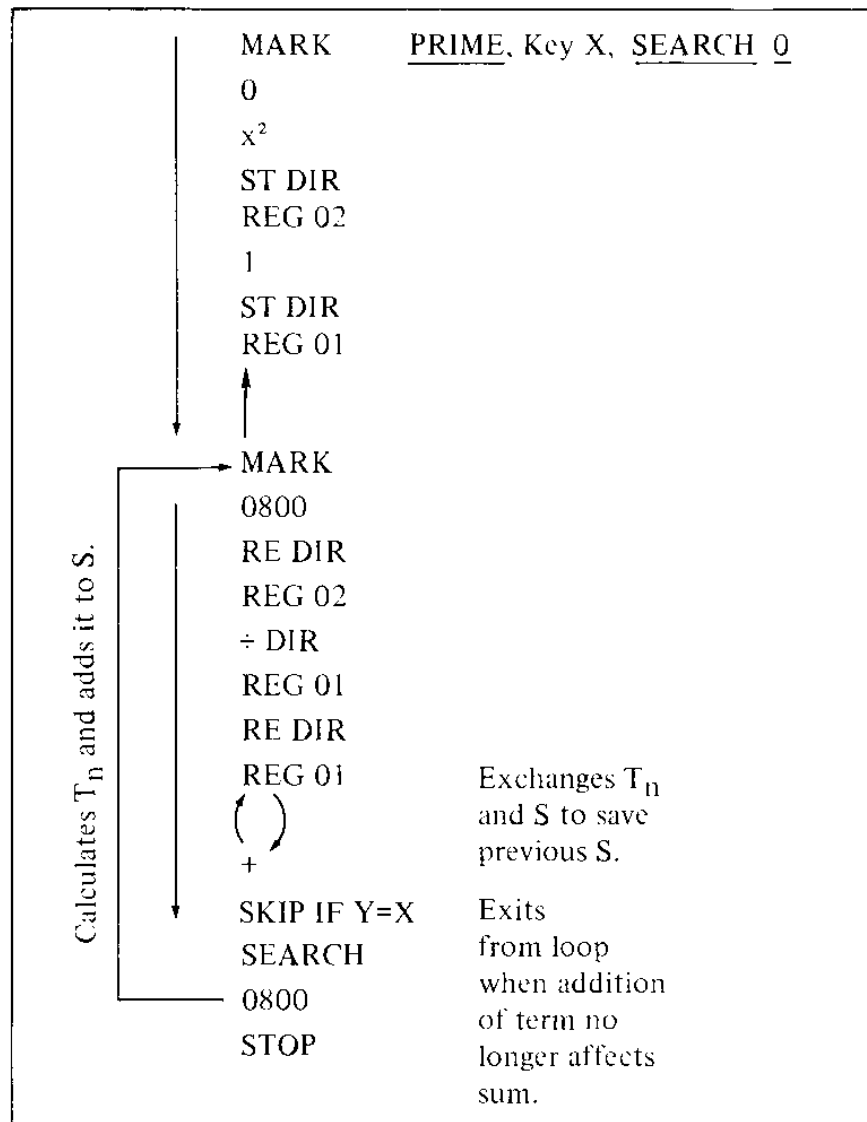
$$S = T_0 + T_1 + T_2 + \dots + T_n \text{ where } T_0 = 1$$

$$T_1 = \frac{1}{x^2} T_0$$

$$T_2 = \frac{1}{x^2} T_1$$

⋮

$$\begin{aligned} & \cdot \\ & \cdot \\ & \cdot \\ & T_n = \frac{1}{x^2} T_{n-1} \end{aligned}$$



The program calculates the sum $\sum_{n=0}^k \frac{1}{x^n}$ to 12 significant digits

for any value of x . The loop is performed many more times for a smaller value of x than it would be for a larger value of x , simply because the series converges faster for large values of x . In all cases when $T_n < 10^{-11}$, the loop is terminated and the final sum is displayed in x and y . If only three digits of accuracy were needed, each successive term could be compared to 10^{-3} ; and when $T_n < 10^{-3}$, the loop could be terminated.

Section V
Decision Commands

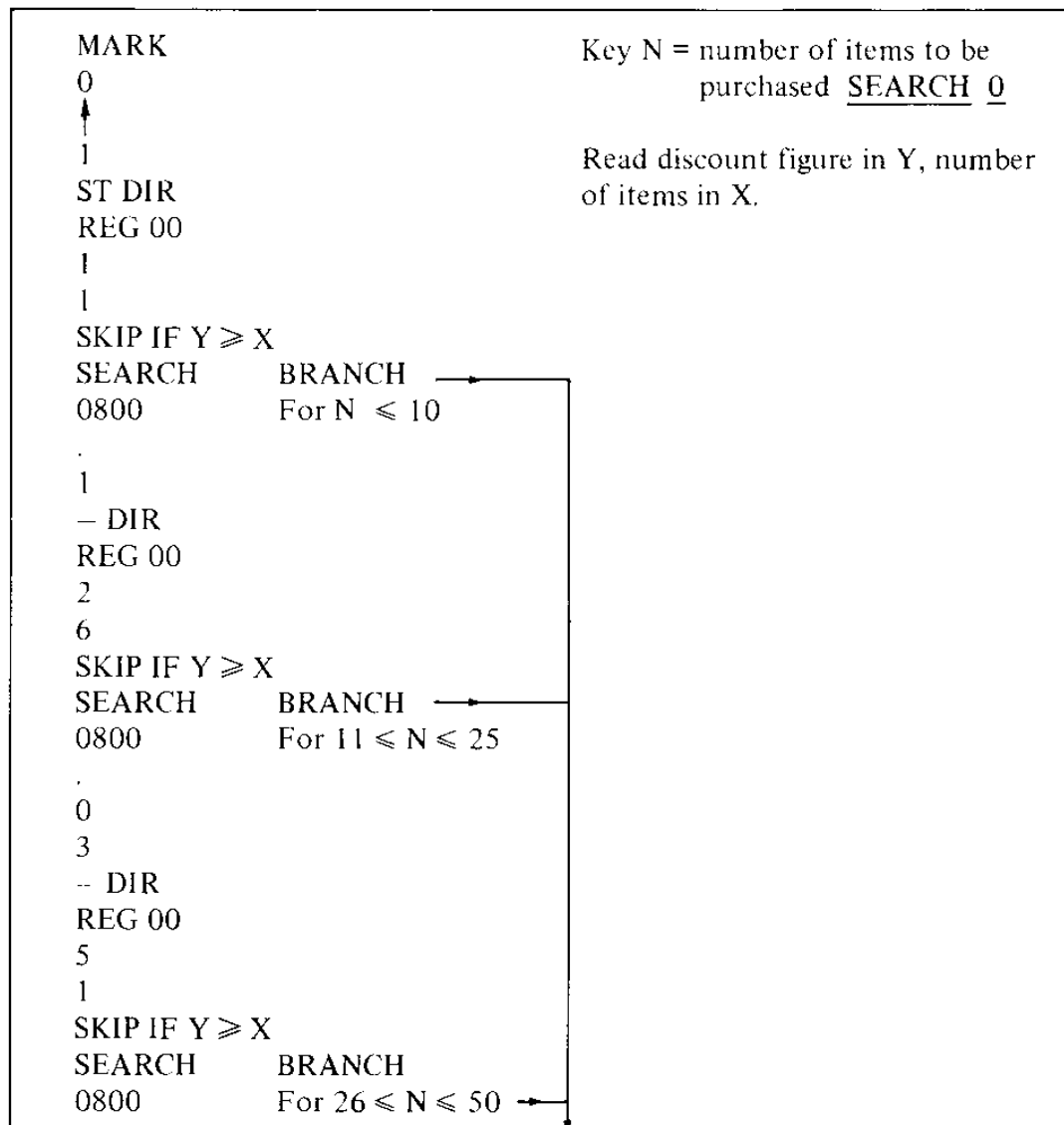
Scanning a Table

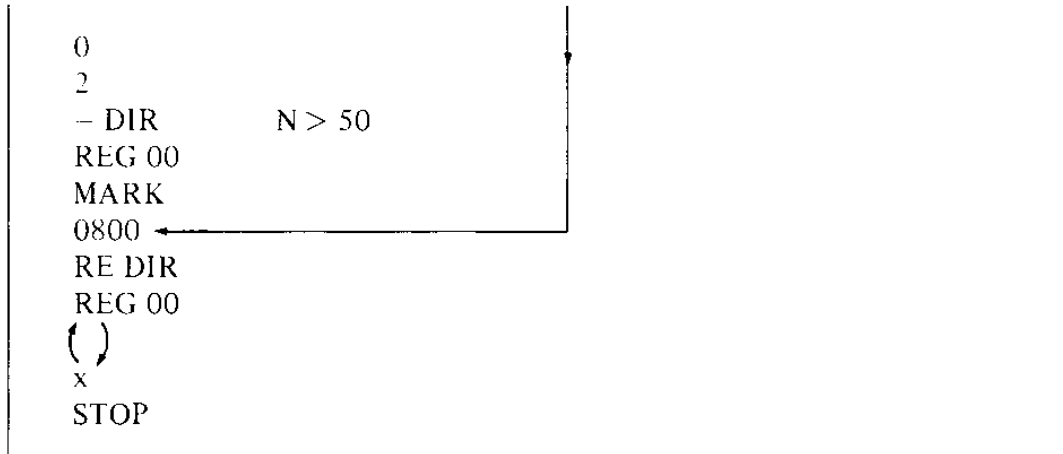
Another frequent use for the decision command is to scan a table or schedule. In many situations, calculations or formulas vary with the class or range the input variable lies in.

A typical example is the pricing of articles. Discounts are often allowed according to the number of articles purchased. Below is a schedule for quantity discounts.

QUANTITY	DISCOUNT
0 to 10	0%
11 to 25	10%
26 to 50	13%
over 50	15%

When given the number of items to be purchased, this program calculates the discount figure which if multiplied by the unit price calculates total cost.





Go

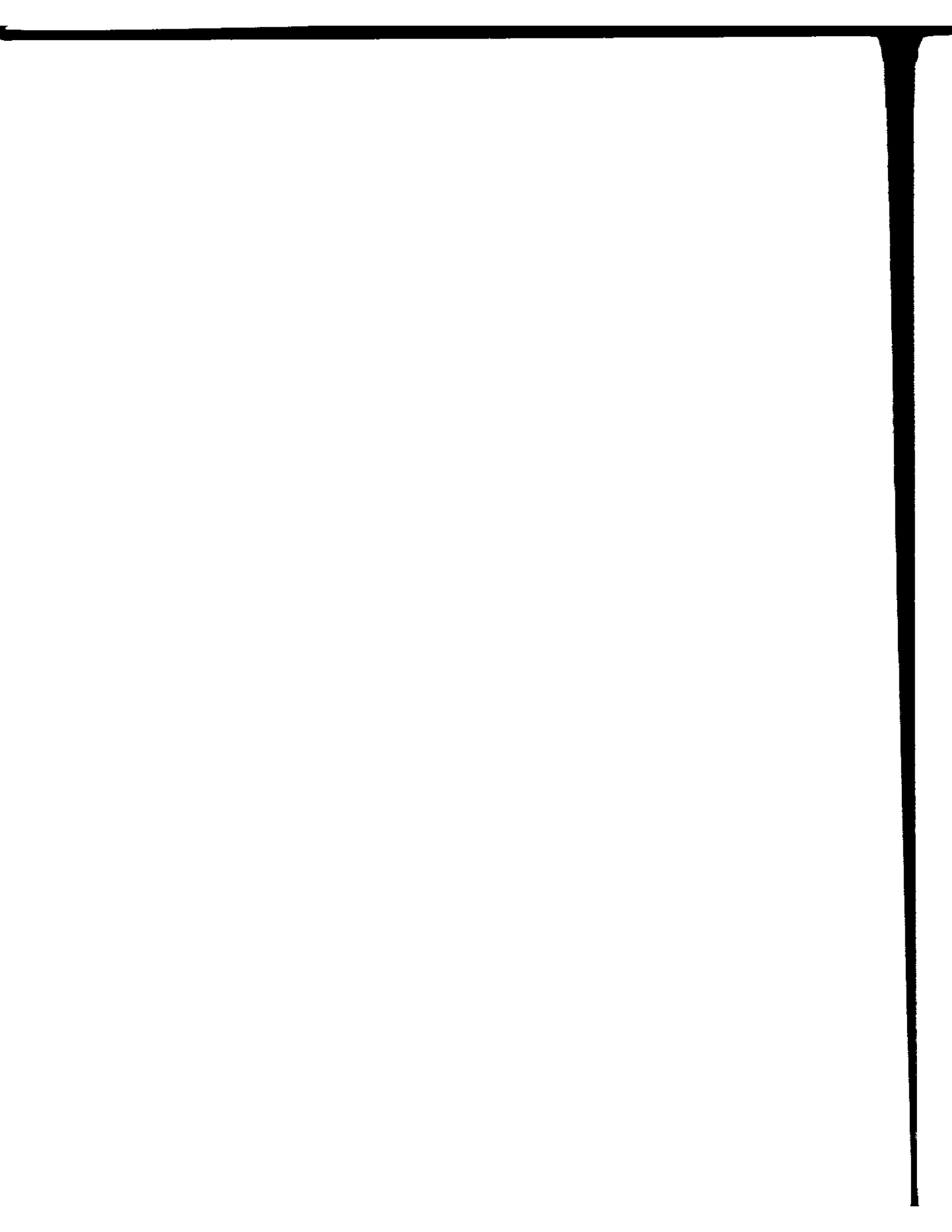
The **GO** key is used to continue the program at the next step after the **STOP** instruction. One important technique that should be pointed out is the idea of using the **GO** command as a do nothing or no-operation instruction.

EXAMPLE

If two angles are unequal, we want to find the sine of the angle in X and use the sine of this angle in future calculations. If the two angles are equal, the angle itself will be used for future calculations. The program would be similar to the following:

	STEP	KEY	
	0000	MARK	
	0001	0700	
	0002	SKIP IF Y = X	
Executes subroutine if $y \neq x$ and returns to Step 0004	0003	SR 0002 (sinx)	
	0004	GO	
	0005	X	Jumps to Step 0005 if $y = x$
	0006	↑	
	0007	π	
	0008	x	
	0009	.	

If the angles are unequal, the command SR 0002 tells the program to execute the sine subroutine. Upon completion of the subroutine, the program branches back to Step 0004. At this point, we do not wish to perform any operation because the **SKIP** command will skip two program steps if the condition is met. We want to perform the same calculation on the variable in X whether it is the sine of the angle or the angle itself. Therefore, a **GO** command is placed in Step 0004 which simply tells the program to continue on the the next step. In this way, the **GO** command can be used as a no-operation command which simply causes the program to continue on without changing or destroying any values.



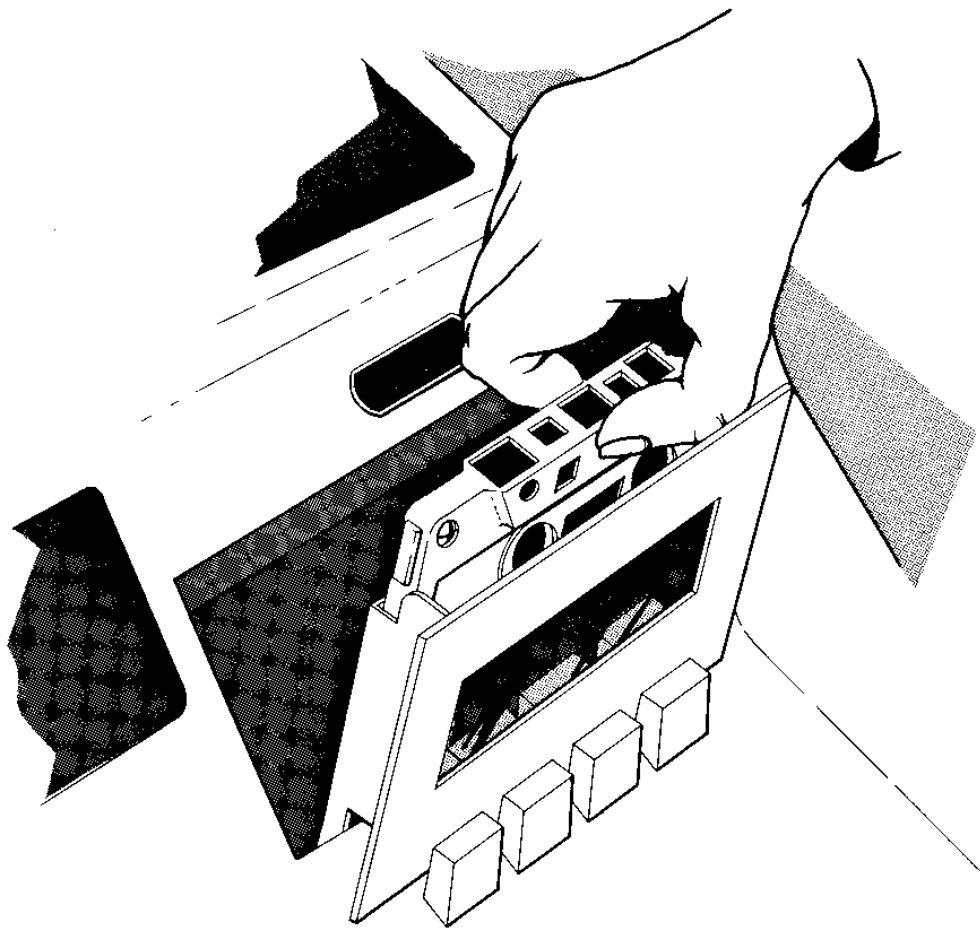
SECTION VI

PROGRAMMING TECHNIQUES USING TAPE CASSETTE

TAPE CASSETTE

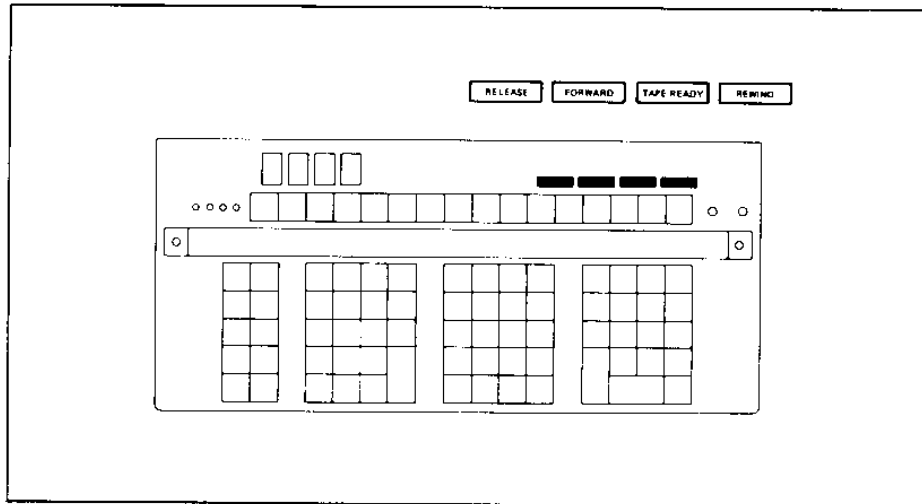
Programs are saved on standard 4" x 2 1/2" x 1/2" magnetic tape cassettes for later use. Up to 20 blocks of programs can be saved on one tape cassette. The tape cassette consists of two tracks and each track can accommodate ten "program blocks."

INSERTING TAPE CASSETTE



Section VI
Programming Techniques

TAPE DRIVE OPERATIONS



There are four switches associated with the tape-drive mechanism.

1. The RELEASE button allows the operator to remove or insert his tape cassette.
2. The FORWARD button moves the tape in a forward direction when depressed.
3. The TAPE-READY button should be pushed when the 720 is to execute a tape instruction. This button places the head of the tape reader in contact with the tape.
4. The REWIND button rewinds the tape when depressed.

MACHINE ERROR INDICATOR

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 X register 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 (See page 2-8). PRIME will turn both error indicators off.

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 insure that a good program stored on tape is not written over or lost accidentally, each track of tape can be protected.



Figure 1

Figure 1 shows a top-view of the tape cassette. There are two small openings with a small plastic covering shown in Figure 1. 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 tape can be used to cover the opening if the tape is to be used for recording other programs.

WHAT IS A PROGRAM BLOCK?

A program block consists of any part of a program (up to 1984 steps) which can be loaded into core at one time. It must be terminated by an **END PROGRAM** instruction. If an **END PROGRAM** instruction is not given and the **RECORD PROGRAM** key is indexed, the **PROGRAM ERROR INDICATOR** will go on after transferring all of core to the tape.

In this instance, the **PROGRAM ERROR INDICATOR** goes on because there is an error in programming (i.e., no **END PROGRAM**) and not a machine malfunction.

A program block **must** contain:

1. 1984 program steps or less.
2. An **END PROGRAM** as a final instruction.

NOTE

*If a program is 1984 steps, the **END PROGRAM** is located at Step 1983. Even though the **END PROGRAM** command is not missing, the **PROGRAM ERROR INDICATOR** will go on when this program is transferred to tape. If this is the case (i.e., a 1984 step program), simply **PRIME** and ignore the **PROGRAM ERROR INDICATOR**.*

END PROGRAM

The **END PROGRAM** key defines a "program block." It is used to signal the end of a **RECORD PROGRAM** or **LOAD PROGRAM** operation. An **END PROGRAM** command is the last instruction to be transferred in a **RECORD PROGRAM** or **LOAD PROGRAM** instruction. Therefore, each program must be terminated by an **END PROGRAM** command if it is to be transferred onto tape.

It is recommended that only one **END PROGRAM** instruction be loaded into core at one time. The primary reason for this is due to the **VERIFY PROGRAM** instruction. When the **VERIFY PROGRAM** key is indexed, the 720 always starts summing at Step 0000 and continues until an **END PROGRAM** instruction is encountered. Therefore, if additional programming instructions are located after the **END PROGRAM**, they will not be included in the sum generated by the **VERIFY PROGRAM** instruction. Therefore, when adding additional programming steps, write over the **END PROGRAM** instruction. This can be accomplished quite easily if it is remembered that the PC is set to the step that the **END PROGRAM** instruction occupies after a **VERIFY PROGRAM** has been executed. Therefore, after indexing **VERIFY PROGRAM**, simply index **LOAD PROGRAM** to load the additional steps or switch to **LEARN MODE** and start indexing them.

Always remember to end your program with an **END PROGRAM** instruction. This instruction is required for transferring the program from core to tape.

NOTE

*An **END PROGRAM** command must not be preceded by a program code whose high-order digit is 04 or 12. Logically, an instruction whose program code is 04XX or 12XX would never precede an **END PROGRAM** command. (See code listing, page 3-1.) One instance which might occur is the following:*

PROGRAM

•
•
•

MARK

0402

•
•
•

SEARCH

0402

END PROG

*If the above program were loaded in core, the program would execute properly. However, if the program were to be transferred from core to tape the **END PROGRAM** instruction would not be recognized as an "END PROGRAM" command. All of core would be transferred to tape and the **PROGRAM ERROR INDICATOR** would be turned on indicating there was no **END PROGRAM** terminating the program block.*

HOW TO "LEARN" A PROGRAM INTO CORE FROM THE KEYBOARD

A program is recorded into core by the following:

1. Place the 720 in LEARN MODE.
2. SET PC at the desired step where the first program command will be stored.
3. Index the program commands.

Remember to always end your program with an **END PROGRAM** instruction. This instruction is required for transferring the program from core to tape.

EXAMPLE

Program to find C: $C = \sqrt{a^2 + b^2}$ (Pythagorean Theorem)

STEP	KEY	CODE
0000	MARK	0408 Key a
0001	0	0700 <u>SEARCH 0</u>
0002	x^2	0713 Key <u>b</u> <u>GO</u>
0003	$\uparrow$	0604 Read c in X
0004	STOP	0515
0005	x^2	0713
0006	+	0600
0007	$\downarrow$	0605
0008	$\sqrt{x}$	0612
0009	STOP	0515
0010	END PROGRAM	0512

The following instructions will introduce the above program into core:

1. Place 720 in LEARN MODE.
2. SET PC to the step you want the first program command to occupy. (To put the first program command at Step 0000, SET PC 0 0 0 0.)

NOTE

An easy way to set the PC at 0000 is by depressing the **PRIME** key.

3. Now simply index the program commands

```

MARK
0
x2
↑
STOP
x2
+
↓
√x
STOP
END PROGRAM

```

Notice while the program is introduced into core the PC displays the program step number and program code currently located at this step. Indexing a key causes the program code of the keystroke to replace the existent code. The PC is increased by one and displays the next step and current code. To see what is now loaded into core beginning at Step 0000, **PRIME** and step through your program.

KEYSTROKE	READ IN X		
PRIME	0000	04	08
STEP	0001	07	00
STEP	0002	07	13

The above indicates that **MARK** is now stored at Step 0000, **0** at Step 0001, x^2 at Step 0002 etc.

To execute the program, place the 720 in RUN MODE

```

Key  a = 3 SEARCH 0
Key  b = 4 GO
Read c = 5 in X

```

By stepping through the program in RUN MODE, each step will be executed one step at a time. In **LEARN MODE** the program is not executed. While stepping through the program in **RUN MODE**, one can see the step number and program code of the instruction about to be executed if the 720 is placed in **LEARN MODE**. Simply remember to put the 720 back in **RUN MODE** before indexing the **STEP** key; otherwise, the instruction will not be executed.

HOW TO TRANSFER A PROGRAM FROM CORE TO TAPE

RECORD PROGRAM

A program can be stored for later use on a magnetic tape.

To transfer a program from core to tape:

1. Place 720 in **RUN MODE**.
2. Insert the Tape cassette; push **TAPE READY** button.

3. **SET PC** to the first step of the program.
(For the above example simply depress **PRIME** key.)
4. Index **RECORD PROGRAM** key, and all the steps from where the PC is set up to and including **END PROGRAM** will be loaded onto the tape. (For this example, Steps 0000 to 0010 are transferred to tape.)

NOTE

RECORDING DATA

Pairs of data storage registers can be recorded on magnetic tape for later use. See diagram (page 3-4). To transfer data from core to tape:

- (1) Place 720 in RUN MODE.*
- (2) Insert Cassette, push TAPE READY.*
- (3) SET PC to program step number corresponding to the data registers. An END PROGRAM command must be located immediately following the data.*
- (4) RECORD PROG*

The same procedure to load the data in any pair of registers is used, except LOAD PROG replaces RECORD PROG in step (4).

HOW TO LOAD A PROGRAM FROM TAPE INTO CORE

The **LOAD PROGRAM** key transfers a program block on magnetic tape to core.

The procedure is to set the PC and index the **LOAD PROGRAM** key. The first program command located on the tape is then stored in the step designated by the PC and the program continues loading until an **END PROGRAM** instruction is encountered. The **END PROGRAM** instruction is the last step to be loaded into core. When the loading is terminated, the PC is automatically set to the first step just loaded into core.

In general, four steps are required for loading a program:

1. Place 720 in RUN MODE.
2. Insert Tape Cassette, push TAPE READY button.
3. **SET PC** to first step in core than program will occupy.
4. Index **LOAD PROGRAM**.

Since ten program blocks can be saved on one side of a tape cassette, what happens if the third block is desired?

If the third block is to be loaded beginning at Step 0000:

1. Place 720 in RUN MODE.
2. Insert tape cassette, REWIND tape completely, push TAPE READY button.
3. **PRIME** (Sets PC to Step 0000).
4. Index **LOAD PROGRAM** (loads 1st block and sets PC to 0000).
 LOAD PROGRAM (loads 2nd block and sets PC to 0000).
 LOAD PROGRAM (loads 3rd block and sets PC to 0000).

The third block on the tape is now loaded into core and is ready to be executed.

Normally, the sequence of steps to follow in loading the nth block of the tape into core is to index the **LOAD PROGRAM** key "n" times. As a final check, the **VERIFY PROGRAM**

key can be indexed and the verify program number checked to be completely sure the correct program is loaded. The **VERIFY PROGRAM** key starts summing from Step 0000 until it encounters an **END PROGRAM** instruction.

BYPASSING PROGRAM BLOCKS

In some instances, loading the preliminary program blocks on a tape will destroy parts of core which must be saved. The problem arises as to how to bypass a program block without loading it into core and destroying data which will be needed for later calculations.

If the PC is set to 1968 and the **LOAD PROGRAM** key indexed, the program will only load into Steps 1968 to 1983. If the program is greater than 16 steps, it will continue loading within those steps (1968 to 1983). In other words, when a program exceeds Step 1983 in core, the remaining part of the program is simply loaded over itself in the first two data registers 000 and 001. The **PROGRAM ERROR INDICATOR** will go on when this occurs.

Therefore, by destroying only the contents of Registers 000 and 001, the first three blocks of the tape can be bypassed.

1. Load tape and place 720 in RUN MODE.
2. **SET PC 1 9 6 8**.
3. **LOAD PROGRAM** (bypasses 1st block).
LOAD PROGRAM (bypasses 2nd block).
LOAD PROGRAM (bypasses 3rd block).
4. **PRIME** (to turn off **PROGRAM ERROR INDICATOR**).
5. **SET PC** to first step that the desired program will occupy.

NOTE

If the program is to be loaded beginning at Step 0000, PRIME will have already set the PC to 0000.

The fourth block of tape is now stored in core and only registers 000 and 001 have been altered.

PROCEDURE FOR CORRECTING SINGLE PROGRAM STEPS

It is quite easy to correct any part of a 720 program. Suppose the following program to accumulate the sum Σx in Register 28 and Σx^2 in Register 29 is loaded into core as shown.

STEP	KEY	CODE	Notice at Step 0023 the - DIR key has been entered by mistake. Correcting this error requires three steps:
0020	MARK	0408	
0021	1	0701	1. Place 720 in LEARN MODE. 2. Set PC at step to be corrected: <u>SET PC 0 0 2 3</u>
0022	STOP	0515	
0023	- DIR	0401	3. Index correct key: <u>+ DIR</u>
0024	REG 28	0208	
0025	x^2	0713	The correct step + DIR is now located at Step 0023. In a similar way, any step in core can be directly assessed using the SET PC key and the correct step keyed in.
0026	+ DIR	0400	
0027	REG 29	0209	
0028	SEARCH	0407	
0029	1	0701	
0030	END PROGRAM	0512	

PROCEDURE FOR INSERTING EXTRA PROGRAM STEPS

Suppose a 400 step program has been introduced into core and three steps which occur in the middle of the program have been omitted. These steps can be inserted without having to key in the entire program again. Using the same program as on the previous page, suppose the two steps 1 + should appear between Steps 0021 and 0022 to indicate which x is about to be entered. The procedure for inserting the steps is as follows:

1. Insert a tape cartridge. REWIND tape completely, and push TAPE READY.
2. **SET PC 0 0 2 2 RECORD PROGRAM.** (This instruction saves the second half of the program. Steps 0022 to 0030, by transferring these steps to tape. The PC is set to Step 0022 when the instruction is completed.)
3. Be sure 720 is in LEARN MODE and index the steps to be inserted:
 - 1 (Loaded at Step 0022, PC increases to 0023.)
 - + (Loaded at Step 0023, PC increases to 0024.)

(Notice that the added steps are loaded into core at the proper place because the PC is set to the first instruction transferred to tape by the **RECORD PROGRAM** instruction.)

4. Put 720 in RUN MODE, REWIND tape, and push TAPE READY.
5. **LOAD PROGRAM** (The steps saved on the tape are now loaded into core beginning at Step 0024.)

The extra steps have been inserted and the program is ready to be executed. The program appears in core as illustrated:

STEP	KEY	CODE
0020	MARK	0408
0021	1	0701
0022	1	0701
0023	+	0600
0024	STOP	0515
0025	+ DIR	0400
0026	REG 28	0208
0027	x^2	0713
0028	+ DIR	0400
0029	REG 29	0209
0030	SEARCH	0407
0031	1	0701
0032	END PROGRAM	0512

PROGRAMMING TECHNIQUES USING TAPE CASSETTE

An interesting feature of the **LOAD PROGRAM** key is that it is programmable. This allows different parts of a program to use the same place in memory at different times. When a **LOAD PROGRAM** instruction is encountered in a program, the next program block is loaded into core beginning at the step immediately following the **LOAD PROGRAM** command. Immediately after the **END PROGRAM** instruction is loaded, control switches to the first instruction loaded and the program is executed.

To take advantage of this command a loop has to be formed. An example illustrates this idea.

Section VI
Programming Techniques

EXAMPLE		
PROGRAM IN CORE		PROGRAM ON TAPE
STEP	COMMAND	
0000	MARK	2
0001	0	↑
0002	LOAD PROG	+
0003	END PROG	↓
0004		ST DIR
0005		REG 00
0006		SEARCH
		0
		END PROG
		3
		↑
		+
		↓
		+ DIR
		REG 00
		STOP
		END PROG

This program will illustrate how to use the **LOAD PROGRAM** instruction in programming and also points out the importance and need for the **END PROGRAM** instruction. The program consists of three parts or blocks. The first part is loaded into core and consists of four instructions.

The program is started by a **SEARCH 0** from the keyboard. The **LOAD PROGRAM** is the first instruction. The 720 immediately starts loading the next program block into core. After the **LOAD PROGRAM** instruction is executed, core will look like the following:

STEP	COMMAND
0000	MARK
0001	0
0002	LOAD PROG
0003	2
0004	↑
0005	+
0006	↓
0007	ST DIR
0008	REG 00
0009	SEARCH
0010	0
0011	END PROG

The program will immediately add $2 + 2$ and store the sum in Register 000. The program then encounters a **SEARCH 0**. Control switches back to **MARK 0** and the **LOAD PROGRAM** instruction is encountered causing the next block to be loaded in replacing the last block. Notice the **END PROGRAM** instruction was never used in executing the program: however, it was needed when the **LOAD PROGRAM** instruction was first executed. If an **END PROGRAM** command had not been located after the **SEARCH 0** command, the 720 would have continued to load programming steps into core. The **END PROGRAM** instruction tells it where to stop.

STEP	COMMAND
0000	MARK
0001	0
0002	LOAD PROG
0003	3
0004	↑
0005	+
0006	↓
0007	+ DIR
0008	REG 00
0009	STOP
0010	END PROG

The program adds $3 + 3$ and adds the sum to Register 000 and stops.

The program is only used to demonstrate how to program the **LOAD PROGRAM** instruction. This technique of a **LOAD PROGRAM** within a program should only be used in long programs which require many program steps and data registers. A valid example might be in a multiple regression analysis where registers are needed for storing sums. In addition, a program for generating the sums and solving simultaneous equations is needed. Therefore, the first program block could initialize all registers and generate the numerous sums needed. When the routine was finished, the second block to solve the simultaneous equations and find the coefficients could be called and loaded into the same part of memory that the first block occupied. In this way, memory can be shared and utilized to its fullest extent.

CREATING A MULTI-BLOCK TAPE

The idea of sharing core storage presents the problem of creating a multi-block tape. The simplest way to explain this procedure is by creating the 3-block program.

Section VI
Programming Techniques

I. To create the first block:

A. Key the first program block into core.

1. Set **LEARN** mode
2. **PRIME** (more generally SET PC to location of first program step)
3. **MARK**
0
LOAD PROG
END PROG.

B. Transfer this block to tape

1. Insert tape cassette, **REWIND**
2. Set **TAPE READY**
3. **PRIME** (or **SET PC** to first step)
4. **RECORD PROG**

II. To create the second block:

A. Key second program block into core

1. **PRIME** (or **SET PC** to first step)
2. 2
↑
+
↓
ST DIRECT
00
SEARCH
0
END PROG

B. Transfer this block to tape

1. **PRIME** (or **SET PC** to first step)
2. **RECORD PROG** .

III. To create the third block:

A. Key third program block into core

1. **PRIME** (or **SET PC** to first step)
2. 3
↑
+
↓
+ **DIR**
00 (Toggles down)
STOP
END PROG

B. Transfer block to tape

1. **PRIME** (or **SET PC** to first step)
2. **RECORD PROG**

- To run the program:
1. **PRIME**
 2. **REWIND** the tape to beginning
 3. Set **TAPE READY**
 4. **LOAD PROG**
 5. **SEARCH 0**

The program will stop with a 6 in X and Y. To recall the sum, **RECALL DIRECT 00** and 10 will appear in X.

NOTE

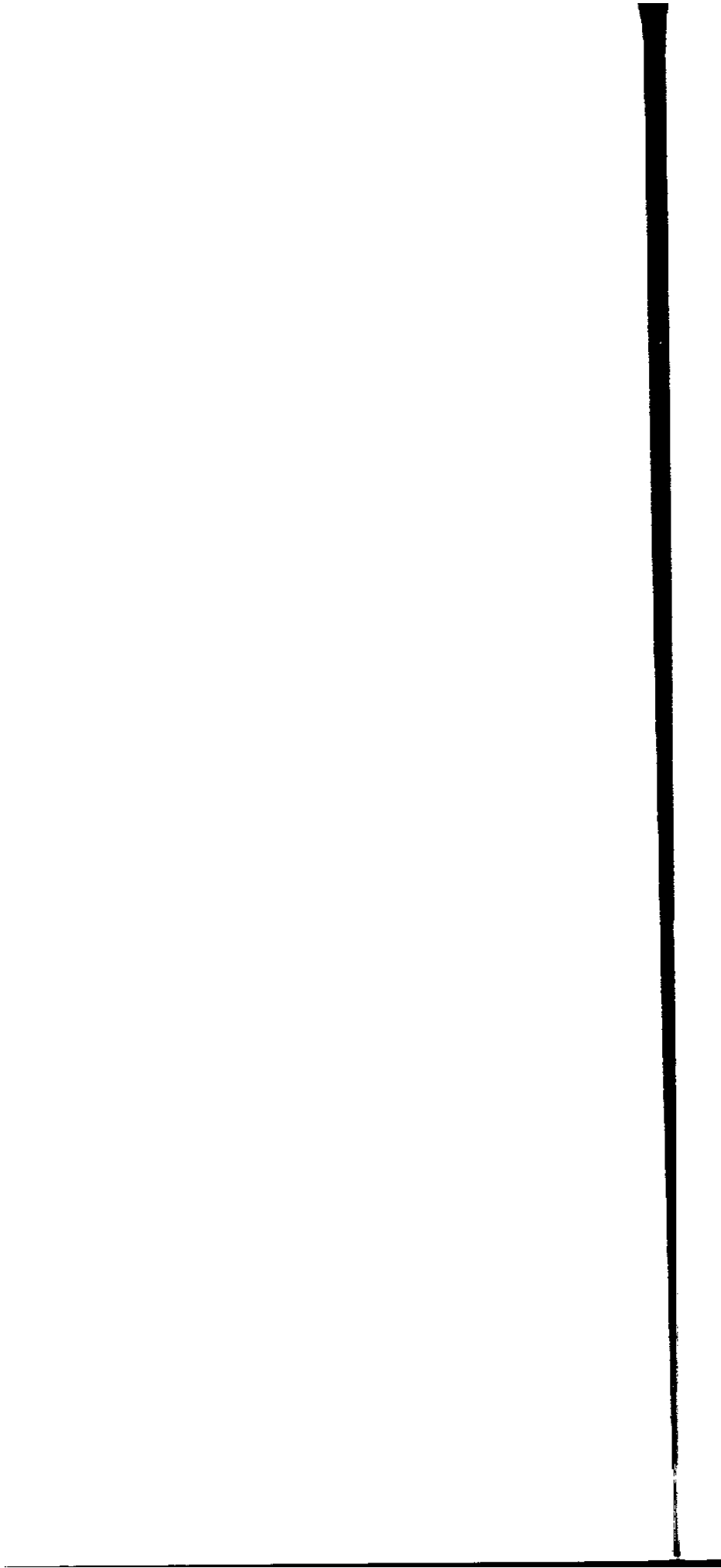
The tape is not moved manually while creating the multi-block tape. If an entirely unrelated program were to be added to this tape as the fourth block, initializing the tape would consist of bypassing the first three program blocks as discussed in Section VI, page 6-8

EXAMPLE

Suppose a program for calculating the t-test for paired variates was located in core from Steps 0100 to 0155 and it is to be recorded as the fourth block on the multi-block tape just created.

1. Insert tape cassette, **REWIND**
2. Set **TAPE READY**
3. Set **RUN** mode
4. **SET PC 1 9 6 8**
5. **LOAD PROG** (bypasses first three program blocks)
LOAD PROG
LOAD PROG
6. **PRIME** (to turn PROGRAM ERROR INDICATOR off)
7. **SET PC 0 1 0 0**
8. **RECORD PROG**

The t-test for paired variates is now recorded as the fourth block on this tape.



SECTION VII

ADDITIONAL COMMANDS NOT FOUND ON THE 720 KEYBOARD

PAUSE COMMAND

One remaining function not previously discussed is the 720 pause command.

WRITE A PAUSE

The WANG 720 has a pause command which allows the user to display the X and Y Registers for .5 seconds at any predetermined point within a program.

The command is a two-step instruction. Since it should only be used within a program, it has not been assigned a regular key on the 720 keyboard.

The two-step command is:

WRITE A followed by the code 0615 which corresponds to the 1/x key.

The following program will count from 0 to 10 displaying each integer in Y for .5 seconds.

<u>KEY</u>	<u>CODE</u>	<u>COMMENT</u>	<u>OPERATING INSTRUCTIONS</u>
MARK	0408		<u>SEARCH</u> <u>O</u>
0	0700		
1	0701		
+	0600		
WRITE A	0412 }	2-Step command causes .5 second pause	
1/x	0615 }		
1	0701		
0	0700		
SKIP IF Y = X	0509		
SEARCH	0407		
0	0700		
STOP	0515		

Multiple PAUSE commands can be used if a longer pause is required. Simply repeat the two-step command for each half second pause.

The pause command is only operational under programming mode. It cannot be used to cause a stop or pause in a program that is executing. If a program is executing and the user desires to stop it at any point, simply index the **STEP** key and the program will stop instantly (See Section II, page 2-3).

Section VII
Additional Commands

In addition to the various commands found on the 720 keyboard, there are several powerful commands which have not been assigned special keys on the Wang 720. These commands are used primarily in programming applications. One of these commands, PAUSE, was discussed above.

These special programming commands can be divided into three basic categories:

1. Storage commands
2. Decision commands
3. Shifting commands

STORAGE COMMANDS (DIRECT ACCESS TO AND FROM THE Y-REGISTER)

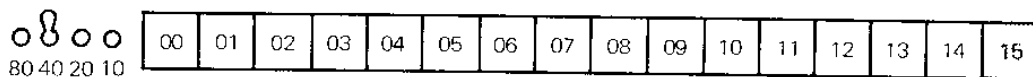
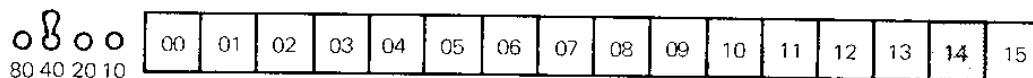
It is possible to transfer data directly to and from the Y-Register and any of the 248 data storage registers. The two-step command is similar to the **RECALL DIRECT** and **STORE DIRECT** commands except that the Y-Register is used in the data transfer rather than the X-Register. The first instruction specifies whether to store or recall, the second instruction designates the internal data register.

STORE Y	0414	Stores the number in Y into the data register designated by the next key-stroke. Y unchanged, X unchanged.
DATA REGISTER	XXXX	
RECALL Y	0415	Recalls to Y the number from the register designated by the next key-stroke. Designated register unchanged, X unchanged.
DATA REGISTER	XXXX	

Since the results of the arithmetic operations using X and Y are placed in Y, the **STORE Y** command saves the program step (↓) of moving the number down into the X-Register when the result is to be saved.

NOTE

The program codes 0414, 0415, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1214, and 1215 must be entered by using the toggle switches and special function keys.



Additional Commands for Addressing the 720 Registers:

EXAMPLE

STORE DIRECT	1204*
DATA REGISTER	XXXX

Stores the number from the X Register to the register designated by the next keystroke. The register pointed to by this keystroke is the numeric sum of the toggle switch settings plus the special function key plus 100.

Store the number 12 in Register 241

Key 1	1
Key 2	2

Set Toggle Switches 80, 40

Key Special Function 04 ST DIR (+100)*

8800 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

Set Toggle Switches 80, 40, 20

Key Special Function 01
$$100 + 80 + 40 + 20 + 01 = 241$$

Reg. 241

8880 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

CODE	KEY
*1200	+ DIRECT (+100)
*1201	- DIRECT (+100)
*1202	XDIRECT (+100)
*1203	÷ DIRECT (+100)
*1204	ST DIRECT (+100)
*1205	RE DIRECT (+100)
*1206	 DIRECT (+100)
*1214	STORE Y (+100)
*1215	RECALL Y (+100)

Any of the codes automatically adds 100 to the storage register number.

*Toggle Switch Settings

DECISIONS

In addition to the four decisions available from the keyboard (See Section V, page 5-1), there are eight other conditions which can be tested. These commands require two instructions. They are each preceded by a **WRITE ALPHA** command and use an existing key on the 720 keyboard for the second half of the command. They are listed as follows. They test for a positive, negative, zero, and non-zero value in the X and Y Registers.

X-REGISTER

WRITE ALPHA SET EXP	Skips next 2 instructions if X is negative
WRITE ALPHA $\text{LOG}_{10} X$	Skips next 2 instructions if X is positive
WRITE ALPHA CHANGE SIGN	Skips next 2 instructions if X is not zero
WRITE ALPHA $\text{LOG}_e X$	Skips next 2 instructions if X is zero

Y-REGISTER

WRITE ALPHA SKIP IF ERROR	Skips next 2 instructions if Y is negative
WRITE ALPHA GROUP II	Skips next 2 instructions if Y is positive
WRITE ALPHA RETURN	Skips next 2 instructions if Y is not zero
WRITE ALPHA WRITE	Skips next 2 instructions if Y is zero

If the condition is met, the next 2 programming instructions are **skipped**. If the condition is not met, the program continues with the next step.

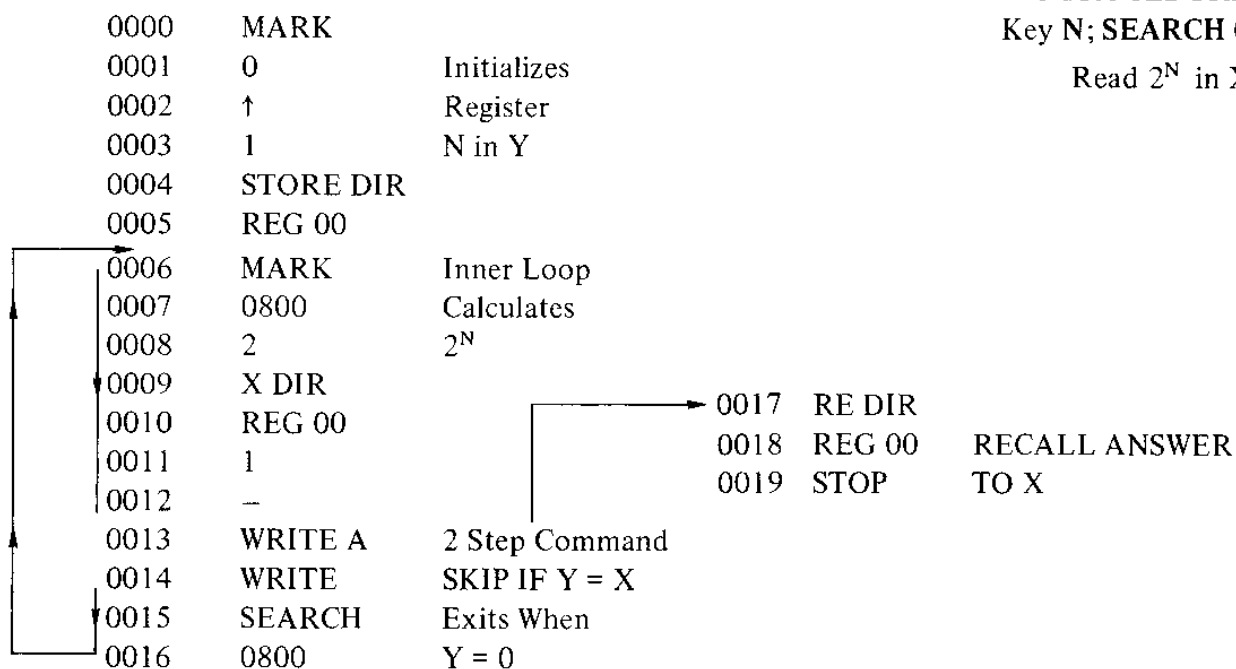
EXAMPLE:

The following program calculates 2^N and illustrates the two-step command **WRITE ALPHA, WRITE** which checks for Y equal to zero.

OPERATING PROCEDURE

Key N; **SEARCH 0**

Read 2^N in X



SHIFTING COMMANDS:

The following two-step commands give the user an easy way to shift the decimal point of the X-Register from 1 to 10 places, left or right.

The first group shifts the decimal point n-places to the right and effectively multiplies the X-Register by 10^n .

WRITE ALPHA 1	Multiplies X by 10^1
WRITE ALPHA 2	Multiplies X by 10^2
WRITE ALPHA 3	Multiplies X by 10^3
WRITE ALPHA 4	Multiplies X by 10^4
WRITE ALPHA 5	Multiplies X by 10^5
WRITE ALPHA 6	Multiplies X by 10^6
WRITE ALPHA 7	Multiplies X by 10^7
WRITE ALPHA 8	Multiplies X by 10^8
WRITE ALPHA 9	Multiplies X by 10^9
WRITE ALPHA 0	Multiplies X by 10^{10}

The second group shifts the decimal point n-places to the left and effectively divides the X-Register by 10^n .

WRITE ALPHA - DIRECT	Divides X by 10^1
WRITE ALPHA X DIRECT	Divides X by 10^2
WRITE ALPHA ÷ DIRECT	Divides X by 10^3
WRITE ALPHA STORE DIRECT	Divides X by 10^4
WRITE ALPHA RECALL DIRECT	Divides X by 10^5
WRITE ALPHA EXCHANGE DIRECT	Divides X by 10^6

Section VII
Additional Commands

WRITE ALPHA SEARCH	Divides X by 10^7
WRITE ALPHA MARK	Divides X by 10^8
WRITE ALPHA GROUP I	Divides X by 10^9
WRITE ALPHA + DIRECT	Divides X by 10^{10}

EXAMPLE

If X contains 12.3456781245 and the command **WRITE ALPHA 3** is given X will then contain 12345.6781245.

If the command **WRITE ALPHA SEARCH** is given, X will contain .123456781245-02

These commands are extremely useful in applications where scaling of input and/or output must be accomplished.

WRITE ALPHA GO	$180/\pi$
WRITE ALPHA STOP	$\pi/180$

SECTION VIII

TRIGONOMETRIC AND STATISTICAL PACKAGE PROGRAM

THE TRIG PACK

The TRIG PACK on the WANG 720 consists of the following 16 trigonometric functions:

SPECIAL OPERATION KEY	TRIG FUNCTION	INPUT RANGE
00	DEGREES TO RADIANS	$ x < 10^{99}$
01	RADIANS TO DEGREES	$ x < 10^{98}$
02	SINE X	$ x < 10^{99}$
03	COSINE X	$ x < 10^{99}$
04	TANGENT X	$ x < 10^{99}$
05	$\sin^{-1} X$	$ x \leq 1$
06	$\cos^{-1} X$	$ x \leq 1$
07	$\tan^{-1} X$	$ x < 10^{99}$
08	TO POLAR	$ x < 10^{50}; y < 10^{50}, \neq 0$
09	TO RECTANGULAR	$0 \leq R < 10^{99} \quad \theta < 10^{99}$
10	SINHX	$ x < 227.9$
11	COSHX	$ x < 227.9$
12	TANHX	$ x < 227.9$
13	$\sinh^{-1} X$	$-10^7 < x < 10^{50}$
14	$\cosh^{-1} X$	$ x \geq 1$
15	$\tanh^{-1} X$	$ x < 1$

These functions are loaded into core memory from a tape cassette which is provided with the machine. The standard TRIG PACK consists of 384 program steps (or 48 Data Registers).

NOTE

The TRIG PACK also uses five data registers: 000, 001, 002, 003 and 248. Care should be taken in using these registers – information stored in these five registers would be lost after an execution of one of the trig functions.

The Y-Register is always preserved and remains unchanged except in the POLAR and RECTANGULAR conversions. When the entire TRIG PACK is loaded into core, core storage for the 720A is as follows:

CORE MEMORY USAGE BY THE TRIG PACK

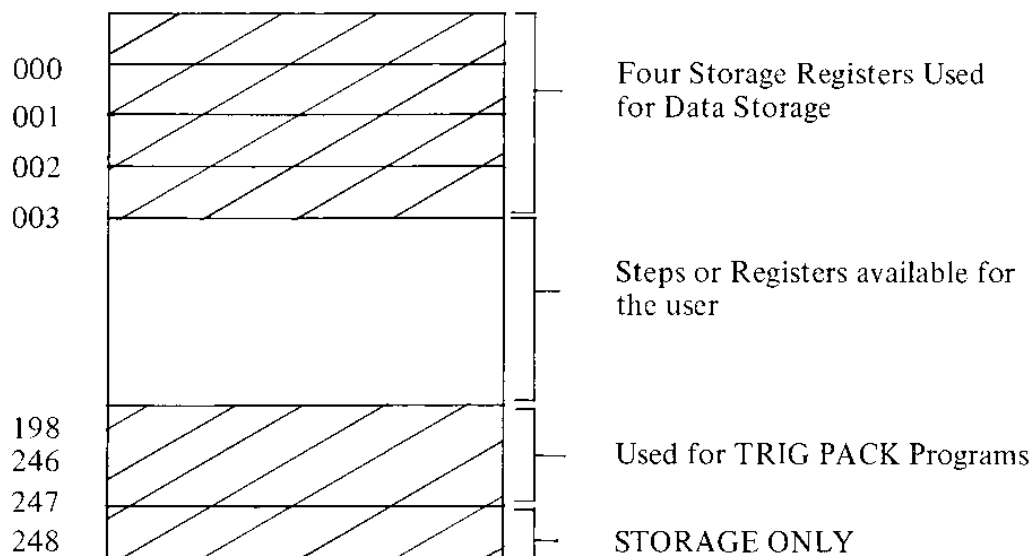


FIGURE 1

SPEED AND ACCURACY

The speed for each function varies. In the worst case, it is no longer than 250 milliseconds. Accuracy is 10 significant digits.

TO LOAD THE TRIG PACKAGE

Like any other tape-to-core operation, the TRIG PACK is loaded as follows:

1. Insert trig-tape, **REWIND**
2. Set **RUN**
3. Set **TAPE READY**
4. **PRIME, LOAD PROGRAM**

By following these steps, the TRIG PACK is loaded into core starting at Step 0000 and utilizes core as indicated in Figure 1. It is recommended that the TRIG PACK always be loaded starting at Step 0000.

In addition to the TRIG PACK, most users will want to load their own programs into core. In order not to erase any of the TRIG PACK, other programs should be introduced into core **beyond** the TRIG PACK. The **VERIFY PROGRAM** key allows us to bypass the TRIG PACK quite easily. After depressing the **VERIFY PROGRAM** key, the PC is set to the step the **END PROGRAM** instruction occupies.

Since it is advisable to have only one **END PROGRAM** instruction in core at any one time, additional programs should start at this step where the **END PROGRAM** instruction is located.

TWO CASES exist:

1. Indexing additional programming steps from the keyboard.
2. Loading another program into core from another tape.

In each case the procedure is basically the same.

CASE 1: Adding Steps from the Keyboard.

1. After loading the TRIG PACK, place 720 in LEARN MODE
2. **VERIFY PROGRAM.**
3. Key program steps desired.

The first step indexed, most likely a **MARK**, will replace the **END PROGRAM** command of the TRIG PACK. Therefore, after completing your own program, an **END PROGRAM** command has to be given. This will complete a new block consisting of the TRIG PACK plus your own program. The **VERIFY PROGRAM** number will then total the codes of the TRIG PACK and your own program.

CASE 2: Loading Another Program into Core from Tape

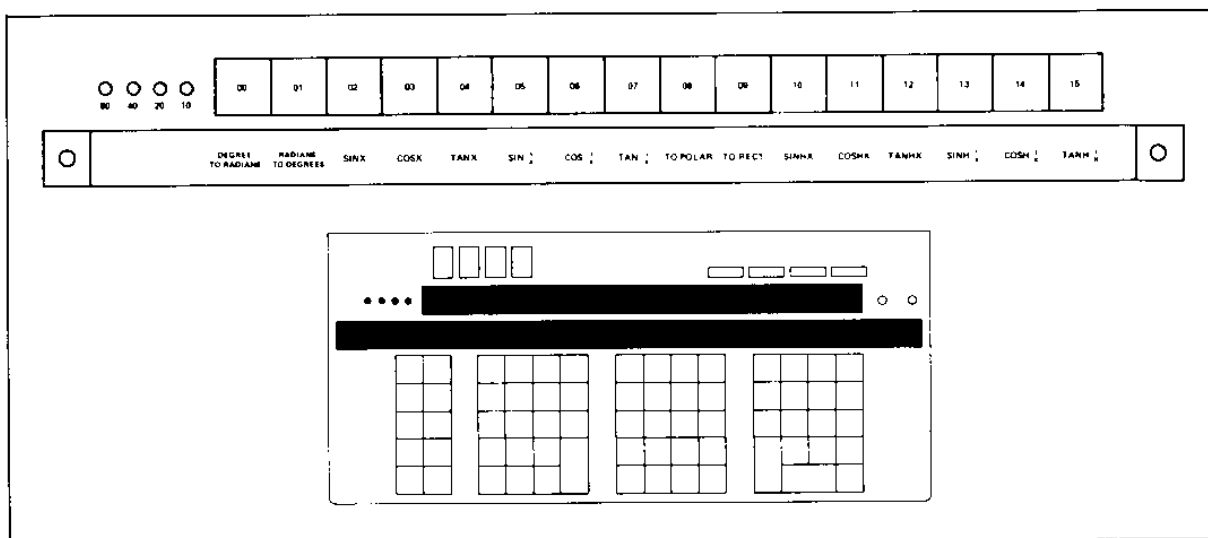
1. After loading the TRIG PACK, insert TAPE CASSETTE which has desired program .
2. Leave 720 in RUN MODE, Set **TAPE READY**
3. **VERIFY PROGRAM** (bypasses TRIG PACK) .
4. **LOAD PROGRAM.**

The program will be loaded into core directly following the TRIG PACK.

USING THE TRIG PACKAGE

The TRIG PACK consists of 16 subroutines which can be addressed from the keyboard or under program control (Sec page 8-6).

KEYBOARD USE: BE SURE THE TOGGLE SWITCHES ARE IN THE OFF (DOWN) POSITION WHEN ADDRESSING THE TRIG FUNCTIONS FROM THE KEYBOARD.



PROGRAM USE

It should be clear that the TRIG PACK is using the special function keys to address the first 16 subroutine codes discussed in Section IV, page 4-5. For instance, the SINE routine is prefixed by a **MARK 0002** and terminated by a **RETURN** command. If the subroutine is addressed within a program, the **RETURN** command transfers control back to the main program. If the subroutine is addressed from the keyboard, control is transferred back to the keyboard. The following shows a user's program utilizing the SINE and COSINE routines.

EXAMPLE: Find $Y = 2 \sin \theta \cos \theta$			
KEY	CODE		
MARK	0408		OPERATING INSTRUCTIONS Key θ in Degrees SEARCH 0 Read $2 \cos \theta \sin \theta$ in Y
0	0700		
↑	0604	STORES θ in Y	
SINX	0002	FINDS SINE θ	
↑↑	0606	SINE θ in Y θ in X	
COSX	0003	FINDS COS θ	
X	0602	$Y = \sin \theta \cos \theta$	
2	0702		
X	0602	$Y = 2 \sin \theta \cos \theta$	
STOP	0515		

The program makes use of the fact that the Y-Register is preserved by storing θ in Y and then the $\sin \theta$ in Y.

DESIGN OF THE TRIG PACKAGE

The TRIG PACK has been designed to give the user greater flexibility. Since the TRIG PACK resides in core memory and is not a "hardware" feature, certain functions which are not used often can be easily deleted. For example, a user may only need SINX, COSX, and $\tan^{-1} X$ for his calculations.

By setting the PC to the step number following these functions, the rest of the TRIG PACK can be deleted and more core storage for other programs and data storage can be gained.

NOTE	
<i>The following steps must be included with any of the trig pack routines:</i>	
MARK	0408
1515	1515
RECALL Y (+ 100)	1215
148	1408
RETURN	0511

STATISTICAL PACKAGE PROGRAM

The STATISTICS PACK is loaded into core the same way that users of the TRIG PACK load the TRIG PACK into core.

FUNCTION	KEY
Mean, Variance, Standard Deviation (ungrouped)	00
Mean, Variance, Standard Deviation (grouped)	01
Normal Distribution	02
Inverse Normal Distribution	03
X ² Statistic	04
X ² Distribution	05
Error Function	06
Binomial Distribution	07
N!	08
Linear Regression	09
Gamma Function	10
Negative Binomial Distribution	11
Poisson Distribution	12
Random Number Generator	13

ASSIGNMENT OF SPECIAL OPERATION KEYS FOR A USER'S OWN SUBROUTINES

The concept of a subroutine was discussed briefly in Section IV, page 4-5. Sixty-four codes are reserved for subroutines on the WANG 720. A subroutine is addressed by a **single** keystroke or a **single program** step. It is prefixed by a **MARK XXXX** chosen from the 64 reserved codes and terminated by a **RETURN** command. The TRIG and STATISTICS packages use the first 16 subroutine codes listed on page 4-5, and are easily addressed by the SPECIAL FUNCTION keys when the Toggle Switches are all in the OFF (DOWN) position.

If Toggle Switch 10 were placed in the ON (UP) position, indexing the **03** key would cause the 720 to look through core for the subroutine beginning with **MARK 0103**. Remember there are 64 codes which can be used as subroutines – not merely 16.

The SPECIAL FUNCTION keys can be used to address the user's own custom-made functions rather than those found in the TRIG and STATISTICS PACKS. Any subroutine which requires only one piece of input data can be addressed by any of the SPECIAL FUNCTION keys.

While in LEARN MODE, the user simply depresses **MARK** followed by one of the SPECIAL FUNCTION keys. This will set the **MARK** flag in core for direct access to the assigned routine. At the end of the subroutine a **RETURN** is given.

For example, assigning 12 to $y = \pi r^2$

KEY	CODE
MARK	0408
12	0012
x^2	0713
$\uparrow$	0604
π	0609
x	0602
RETURN	0511

To call for this function simply index r into X, and press 12. The answer will be given in Y. The subroutine can also be addressed under program control in the same way as the TRIG functions. In this way the user may assign and label any of his own functions to the special operation keys.

SECTION IX

SAMPLE PROGRAM

700 PROGRAM DESCRIPTION

PROGRAM TITLE				NUMBER 1004A/MA6				
ALGEBRA OF COMPLEX NUMBERS +, -, x, ÷				PROGRAMMED BY C. M. TANG				
PROGRAM ABSTRACT				DATE SEPTEMBER, 1969				
+, -, x, and ÷ complex numbers				NO. OF STEPS 084				
BLOCKS								
NO.	NO. OF STEPS	DATA REGISTERS	MARK USED	VERIFY NUMBER				
	084	000, 005	0201, 0600, 0601, 0603, 0602	636				
				SET P.C. 000				
TO "LEARN" PROGRAM (keyboard to core)		TO RECORD PROGRAM (core to tape)		TO LOAD PROGRAM (tape to core)				
1. Set <u>LEARN</u> mode. 2. <u>SET PC</u> to desired step. 3. Index keys in program. 4. Index <u>END PROGRAM</u> as last step in program.		1. Insert tape cartridge. <u>REWIND</u> if necessary. 2. Set <u>TAPE READY</u> . 3. <u>SET PC</u> to first step of program. 4. Index <u>RECORD PROGRAM</u> .		1. Set <u>RUN</u> mode. 2. Insert tape cartridge. <u>REWIND</u> if necessary. 3. Set <u>TAPE READY</u> . 4. <u>SET PC</u> to desired step number. 5. Index <u>LOAD PROGRAM</u>				
PROGRAM DESCRIPTION:								
This program can perform simple +, -, x, and ÷ as well as chain operations. This is because the answer of the previous operation is saved, the real part in storage 001 and imaginary part in storage 000. When reading the answer, the real part is in Y register and imaginary part in X register. Same formula is used for entering the complex numbers.								
OPERATING PROCEDURE:		EXAMPLE:						
1. <u>PRIME</u> ; <u>VERIFY PROGRAM</u>		1. <u>PRIME</u> ; <u>VERIFY PROGRAM</u>						
2. Index first number, real part in y and imaginary part in x Set the Toggle Switches to 20 ; Key <u>01</u>		2. <u>2</u> <u>↑</u> <u>5</u> ; <u>CH SIGN</u> Set the Toggle Switches to 20 ; Key <u>01</u>						
3. Index second number the same way		3. <u>4</u> <u>↑</u> <u>3</u>						
4. <u>SEARCH</u> <u>+</u> if addition <u>-</u> if subtraction <u>x</u> if multiplication <u>÷</u> if division		4. <u>SEARCH</u> <u>+</u>						
5. Read answer If (2 - 5i) + (4 + 3i)		5. Read <table border="1"> <tr> <td>Y</td> <td>+6.0000000000</td> </tr> <tr> <td>X</td> <td>-2.0000000000</td> </tr> </table>			Y	+6.0000000000	X	-2.0000000000
Y	+6.0000000000							
X	-2.0000000000							

Section IX
Sample Programs

PROGRAM TITLE ALGEBRA OF COMPLEX NUMBERS +, -, x, ÷	NUMBER 1004A/MA6
PAGE 2	

If $(2 - 5i) - (4 + 3i)$

If $(2 - 5i) \times (4 + 3i)$

If $\frac{(2 - 5i)}{(4 + 3i)}$

4. SEARCH -
5. Read

$$\begin{array}{r|l} Y & -2.0000000000 \\ X & -8.0000000000 \end{array}$$
4. SEARCH x
5. Read

$$\begin{array}{r|l} Y & +23.0000000000 \\ X & -14.0000000000 \end{array}$$
4. SEARCH ÷
5. Read

$$\begin{array}{r|l} Y & -.280000000000 \\ X & -1.040000000000 \end{array}$$

For chain operations repeat steps 3, 4, and 5

EXAMPLE:

$$\frac{[(1 + 2i)(3 + 4i)] + (6 - 9i)}{3 + 4i}$$

2. 1 ↑ 2 ; Key 0201
3. 3 ↑ 4 ; SEARCH x
- 4 & 5. Read

$$\begin{array}{r|l} Y & -5.0000000000 \\ X & +10.0000000000 \end{array}$$
3. Key 0201
- 6 ↑ 9 ; CH SIGN
4. SEARCH +
5. Read

$$\begin{array}{r|l} Y & +1.0000000000 \\ X & +1.0000000000 \end{array}$$
3. Key 0201
- 3 ↑ 4
4. SEARCH ÷
5. Read

$$\begin{array}{r|l} Y & +.280000000000 \\ X & -.400000000000-01 \end{array}$$

700 PROGRAM TITLE: ALGEBRA OF COMPLEX NUMBERS

NO. 1004A/MA6 Page 1 of 2

Step	Key	Code	Comment
00 0	MARK	0408	
1	0201	0201	
2	ST DIR	0404	
3	REG 00	0000	
4	STORE Y	0414	
5	REG 01	0001	
6	CLEAR X	0715	
7	↑	0604	
8	STOP	0515	
9	MARK	0408	
01 0	-	0601	
1	CH SIGN	0711	
2	↙	0606	
3	CH SIGN	0711	
4	↘	0606	
5	MARK	0408	
6	+	0600	
7	+ DIR	0400	
8	REG 00	0000	
9	RE DIR	0405	
02 0	REG 01	0001	
1	+	0600	
2	RE DIR	0405	
3	REG 00	0000	
4	STOP	0515	
5	MARK	0408	
6	÷	0603	
7	CH SIGN	0711	
8	ST DIR	0404	
9	REG 03	0003	
03 0	x^2	0713	
1	ST DIR	0404	
2	REG 05	0005	
3	↓	0605	
4	x^2	0713	
5	+ DIR	0400	
6	REG 05	0005	
7	RE DIR	0405	
8	REG 05	0005	
9	÷	0603	

Step	Key	Code	Comment
04 0	÷ DIR	0403	
1	REG 03	0003	
2	RE DIR	0405	
3	REG 03	0003	
4	MARK	0408	
5	x	0602	
6	ST DIR	0404	
7	REG 003	0003	
8	RE DIR	0405	
9	REG 00	0000	
05 0	ST DIR	0404	
1	REG 04	0004	
2	↓	0605	
3	X DIR	0402	
4	REG 00	0000	
5	RE DIR	0405	
6	REG 01	0001	
7	ST DIR	0404	
8	REG 02	0002	
9	↓	0605	
06 0	X DIR	0402	
1	REG 01	0001	
2	RE DIR	0405	
3	REG 03	0003	
4	↑	0604	
5	X DIR	0402	
6	REG 04	0004	
7	RE DIR	0405	
8	REG 04	0004	
9	- DIR	0401	
07 0	REG 01	0001	
1	RE DIR	0405	
2	REG 01	0001	
3	↘	0606	
4	X DIR	0402	
5	REG 002	0002	
6	RE DIR	0405	
7	REG 02	0002	
8	+ DIR	0400	
9	REG 00	0000	

Section IX
Sample Programs

700 PROGRAM TITLE: ALGEBRA OF COMPLEX NUMBERS NO. 1004A/MA6 Page 2 of 2

Step	Key	Code	Comment
08 0	RE DIR	0405	
1	REG 00	0000	
2	RETURN	0511	
3	END PROG	0512	
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

Step	Key	Code	Comment
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

SECTION X

WARRANTY, SERVICE AND MAINTENANCE

WARRANTY

Wang electronic equipment is warranted to be free from defects in workmanship and materials for 90 days from delivery to the original purchaser; parts only are warranted for one year, exclusive of labor. Readout tubes, transistors, and fuses are subject to the RETMA guarantee (substituted tubes should be returned to Wang Laboratories). This warranty is in lieu of all other warranties expressed or implied, except as specifically modified in writing by a document signed by an officer of WANG LABORATORIES, INC. Except for such a document, no representative or other person is authorized to represent or assume for WANG LABORATORIES, INC. any warranty liability beyond that set forth herein. Use limits and time between overhaul hours may be specified for mechanical and rotary elements of a Wang system. During the warranty period, Wang equipment is serviced free of charge except for occasional freight cost to and from a service center if equipment is located beyond a 75-mile radius.

POST-WARRANTY SERVICE AVAILABILITY

Wang Service Centers are located in many major cities throughout the world. It is a product service policy to restore the operation of a customer's unit in the shortest time possible after receipt of call. For remotely located users, equipment turnaround is normally within one day after arrival at the center. Spare parts, as well as circuit board repair capability are available at all service centers.

ANNUAL MAINTENANCE CONTRACT

An annual maintenance contract is available that consists of adjusting, replacing parts when required and keeping the equipment in first class operating condition. The contract includes all necessary service calls. It does not include repair necessitated by accident, fire, current fluctuations, abuse, or negligence.

POST-WARRANTY SERVICE CALLS WITHOUT MAINTENANCE CONTRACT

All service calls made to customers' facilities not having service contracts will be charged on an hourly basis point to point between the Wang Service Center and equipment location. Automobile charges per mile and material costs will also be included.

NOTE

Users who attempt to repair Wang equipment, without receiving prior Wang equipment training, run the risk of causing further damage to their equipment. Also, and more important, internal equipment voltages are present that could cause severe electrical shock.

IN-HOUSE MAINTENANCE CAPABILITY

Wang Laboratories offers product familiarization lessons for customers who desire to build up an in-house capability for maintaining their equipment. The customer, of course, is expected to defray the travel and living expenses of his service representative while in training. Please write including details to Product Service Manager, Wang Laboratories, Inc., 836 North Street, Tewksbury, Massachusetts for a price quotation on the course.

APPENDIX

TYPING CONVENTIONS

TYPING CONVENTIONS FOR 720 PROGRAM LIBRARY

Certain keyboard instructions have been abbreviated for typing convenience. The following is a listing of the keyboard instructions and their abbreviations.

KEY	ABBREVIATION	CODE
+ DIRECT	+ DIR	0400
DIRECT	-- DIR	0401
X DIRECT	X DIR	0402
÷ DIRECT	÷ DIR	0403
STORE DIRECT	ST DIR	0404
RECALL DIRECT	RE DIR	0405
 DIRECT	EX DIR	0406
SEARCH	SEARCH	0407
MARK	MARK	0408
GROUP 1	GROUP 1	0409
GROUP 2	GROUP 2	0410
WRITE	WRITE	0411
WRITE ALPHA	WRITE A	0412
END ALPHA	END A	0413
STORE Y	STORE Y	0414
RECALL Y	RECALL Y	0415
+ INDIR	+ INDIR	0500
- INDIR	- INDIR	0501
X INDIR	X INDIR	0502
÷ INDIR	÷ INDIR	0503
STORE INDIR	ST INDIR	0504
RECALL INDIR	RE INDIR	0505
 INDIR	EX INDIR	0506
SKIP IF Y ≥ X	SKIP IF Y ≥ X	0507
SKIP IF Y < X	SKIP IF Y < X	0508
SKIP IF Y = X	SKIP IF Y = X	0509
SKIP IF ERROR	SKIP ERROR	0510
RETURN	RETURN	0511

Appendix

END PROG	END PROG	0512
LOAD PROG	LOAD PROG	0513
GO	GO	0514
STOP	STOP	0515
+	+	0600
—	—	0601
x	x	0602
÷	÷	0603
↑	↑	0604
	↓	0605
↑ ↓	↑ ↓	0606
x	x	0607
INTEGER X	INT X	0608
π	π	0609
$\text{LOG}_{10} X$	$\text{LOG}_{10} X$	0610
$\text{LOG}_e X$	$\text{LOG}_e X$	0611
$\sqrt{x}$	$\sqrt{x}$	0612
10^x	10^x	0613
e^x	e^x	0614
$1/x$	$1/x$	0615
0	0	0700
1	1	0701
2	2	0702
3	3	0703
4	4	0704
5	5	0705
6	6	0706
7	7	0707
8	8	0708
9	9	0709
SET EXP	SET EXP	0710
CHANGE SIGN	CH SIGN	0711
decimal point (·)	decimal point (·)	0712
x^2	x^2	0713
RECALL RESIDUE	RESIDUE	0714
CLEAR X	CLEAR X	0715

NOTE

For typing convenience the exchange keys,  **DIRECT** and  **INDIR**, will be typed as **EX DIR** and **EX INDIR**. **SKIP IF ERROR** has been shortened to **SKIP ERROR**.

In designating the 249 data registers in the KEY column, the register numbers will be preceded by REG.

EXAMPLE

To store into register 58 the coding sheet will appear as follows:

ST DIR
REG 58

A subroutine will be designated in the KEY column by a SR preceding the subroutine code.

EXAMPLE

To address a subroutine beginning with **MARK 0303** the KEY column will appear as follows:

SR 0303

INDEX

- Additional Commands for Addressing the 720 Registers 3-2, 7-3
- Branching 4-3
- Bypassing Program Blocks 6-8
- Central Processing Unit 1-1
- Change Sign 2-5
- Clear X 2-6
- Coding 3-1
- Core Memory 3-3, 3-4, 3-6
- Data Storage Registers 2-9
- Decisions 5-1
 - Skip if $Y = X$ 5-1
 - Skip if $Y \geq X$ 5-2
 - Skip if $Y < X$ 5-3
 - Skip if Error 2-9, 5-3
- Direct Addressing 2-9
- Display 2-4
- End Program 2-2, 6-4, 6-5
- Entering a Number 2-5
- Exchange 2-13
- Go 2-3, 5-9
- Group 1 2-21
- Group 2 2-21
- High-Order Digit 3-2
- Indirect Addressing 2-14, 2-16
- Indirect Keys 2-14
- Learn Mode 2-1, 2-4, 2-12, 6-5, 6-6
- Learn-Print Mode 2-1
- Learn a Program Into Core From the Keyboard 6-5
- List Program Mode 2-2
- Load Program 2-3, 6-4, 6-7
- Looping 4-4
- Looping With a Counter 5-4
- Looping Without a Counter 5-6
- Low-Order Digit 3-2
- Machine-Error Indicator 2-8, 6-2
- Mark 4-1, 4-3, 4-4
- Mark Assignment 4-2
- Mode of Operation 2-1
- Multi-Level Subroutines 4-7
- Non Programmable Keys 2-2
- Number of Registers Occupied by a Program 3-5
- Pause 7-1
- Prime 2-2, 2-3, 6-5
- Procedure for Inserting Extra Program Steps 6-9
- Program Block 6-3
- Program Counter and Set PC 2-3
- Program-Error Indicator 2-8, 2-9, 4-2, 5-3, 6-2
- Programming Concepts 4-1
- Read-Only Memory 1-1
- Recall Direct 2-12, 7-2
- Recall Residue 2-16, 2-20
- Record Program 2-3, 2-4
- Return 4-6, 4-7
- Rewind 6-2
- Run Mode 2-1, 6-6
- Scanning a Table 5-8
- Search 4-1, 4-3, 4-4
- Set Exp 2-5
- Shifting Commands 7-5
- Special Function Keys 2-9
- Starting Address 8-4
- Statistics Functions 8-6
- Statistical Pack 8-6
- Step 2-3
- Stop 4-2
- Storage Assignment 3-5
- Storage Commands (Direct Access to and from the Y-Register) 7-2
- Store Direct 2-9, 2-12, 7-2
- Subroutine 4-5
- Tape Cassette 6-1
- Tape-Drive Operation 6-2
- Tape Ready 6-2
- Toggle Switch 2-9
- Trig Functions 8-5
- Trig Pack 8-1, 8-3
- Turning the 720 On 2-2
- Typing Conventions A-1
- Verify Program 2-4, 6-4
- Working Storage Area 3-5
- Write Alpha 2-3, 2-21, 7-1, 7-2
- Write Alpha Pause 7-1
- Write Command 2-20
- X-Register 2-5
- Y-Register 2-6

NOTES:

NOTES: