



MONROE

Beta 326 Scientist





















Table of Contents





































Section 1 Introduction







The Monroe 326 Scientist	3
The Monroe 392 Tape Cassette Drive	4
How to Use This Manual	6

Section 2 Basic Operations

Getting Started	9
Entering Numbers	9
Clear CLEAR	10
Change Sign CHG SIGN	10
Setting the Decimal Point SET D.P.	11
Exponent EXP	12
Summary of Rules for Entering Numbers	14
Basic Arithmetic and the Number/Operator Register	15
Reset RESET	17
Differences Between Reset and Clear	17
Equal Sign =	17
Basic Arithmetic Procedure	18
Addition	18
Subtraction	19
Multiplication	19
Division	20
Chain Arithmetic	20
Parentheses (and)	22
Summary of Rules for Using Parentheses	23
Raising a Number to a Power a^x	24
Constant Arithmetic	26
Addition Using a Constant	27
Subtraction Using a Constant	27
Multiplication Using a Constant	27
Division Using a Constant Dividend	27
Division Using a Constant Divisor	28
Raise a Constant to a Power	28
Repeated Use of + × or =	28

Angles.....	30
Entering Angles.....	30
Degrees.....	30
Grads.....	32
Radians.....	33
Arithmetic with Angles.....	33
Converting Angles.....	34
Data Storage Registers.....	34
Store 	36
Recall 	37
Exchange  	37
Using the Contents of Registers in a Calculation Sequence.....	38
Direct Register Arithmetic.....	39
 with Arithmetic Operator.....	40
  with Arithmetic Operator.....	42
 with Arithmetic Operator.....	44
Sample Application of Register Arithmetic.....	46
Section 3 Functions	
Functions.....	51
Reciprocal Function 	51
Factorial  	51
Square Root 	52
Square  	52
Logarithm (Base 10) 	52
Antilogarithm (Base 10)  	53
Logarithm (Base e) 	53
Antilogarithm (Base e)  	53
Hyperbolic Functions.....	54
Functions Involving Angles.....	56
The GRAD/DEG switch.....	56
Converting Degrees or Grads to Radians.....	56
Converting Radians to Degrees or Grads.....	57
Converting D/M/S Angles to Degrees with Decimal Fraction.....	57

Sine 	58
Arc Sine  	58
Cosine 	59
Arc Cosine  	59
Tangent 	59
Arc Tangent  	60
Polar to Rectangular Coordinates.....	61
Rectangular to Polar Coordinates.....	62
Statistical Functions.....	63
Clear Registers 1, 2 and 3  	63
Using  with Ungrouped Data.....	63
Using  with Grouped Data.....	64
Deleting Data from a Data Group.....	65
Mean Value  	66
Standard Deviation  	67
Summary of Rules for Statistics Calculations.....	68
Metric/English Conversions.....	68
Other Conversions.....	70
The Number e  	70
Absolute Value  	70
Fraction  	71
Integer  	71
The Number π  	71
Round  	72
Identifier  	72
Pause  	72
Clear All Registers  	73
Summary of Functions Using  Key.....	73
Sample Applications of Functions.....	74
Distance Traveled by Projectile.....	74
Converting Three-Dimensional Rectangular Coordinates to Spherical Coordinates.....	75
Mass of a Moving Electron.....	77

Great Circle Distance	78
Statistical Application	80
Section 4 Programming	
What is a Program?.....	82
Programming and the 326 Scientist.....	84
Program Controls.....	84
The RUN/STEP/LOAD switch	85
Display During Program Loading.....	85
The START/STOP Key	86
The Jump Key	87
Loading a Program.....	87
Executing a Program.....	87
Flowcharting.....	88
Introductory Program.....	88
Sample Program: Volume of a Hollow Sphere.....	90
Labels	93
Sample Program Using Labels.....	96
Modifying a Program.....	100
The Forward Key 	100
The Backspace Key 	100
The Insert Key 	100
The Remove Key 	100
Accessing a Step in Memory	101
Inserting an Instruction.....	102
Changing an Instruction	102
Removing an Instruction.....	102
Sample Program Modifications.....	103
Subroutines and Branching	105
Branch 	106
Return 	106
Rules for the Use of Subroutines	107
Sample Program Using a Subroutine	107
Decisions and Conditional Testing.....	111

Flowchart Symbol for Test.....	111
Conditional Jump.....	112
Conditional Branch	113
Sample Program Using a Conditional Jump	113
Sample Program That Tests Whether an Entry Has Been Made.....	116
Conditional Tests on Numbers Other Than Zero	118
Stepping Through a Program	123
The Pause Instruction.....	125
Look Ahead Feature.....	126
Display of Program Listing	126
Some Programming Hints and Techniques.....	127
Handling Variables and Constants	127
Stop Instructions in the Middle of Arithmetic Sequences.....	127
Use of Identifiers	128
Direct Register Arithmetic in Programs	128
Sample Program: Evaluation of a Polynomial	128
Section 5 Using the 392 Tape Cassette Drive	
Why Use a Tape Cassette Drive?.....	135
Connecting the Tape Drive.....	136
Tape Cassettes Used	137
Monroe Endless Tape Cassette	137
Monroe Tape Cassette	138
Protection of Tape Against Accidental Erasure	138
Tape Controls.....	139
Tape Movement Counter	140
Tape Blocks and Preconditioning Tape	140
Block Search.....	143
Using Prepared Programs with the 326.....	144
Writing a Program onto a Block of Tape	145
Reading a Program from the Tape into Program Memory	147
Writing Data Registers onto Tape	148
Reading from Tape into the Registers of the 326	149

Programmed Tape Control	151
Programmed Tape Control Instructions.....	154
Example: Programmed Tape Control for Overlays	154
Example: Programmed Tape Control for Writing from Data Registers onto Tape	160
Appendix A	
Accuracy and Number Sizes	168
Appendix B	
Batteries and Recharging	169
Appendix C	
Summary of 326 and 392 Keys and Switches.....	170
Appendix D	
Instruction Codes for the 326 Scientist.....	174
Appendix E	
326 Scientist Error Conditions	175
Appendix F	
392 Tape Cassette Drive Error Conditions	176
Appendix G	
Parentheses and the Number/Operator Register	177
Appendix H	
Auto Test Routine	180
Appendix I	
Environmental Conditions.....	181
Appendix J	
Peripheral Devices.....	182
Appendix K	
Notes on the Tape Cassette Drive.....	184
Index	185

List of Figures

Figure Number	Subject	Page
1	Monroe Beta 326 Scientist	2
2	Monroe 392 Tape Cassette Drive	5
3	Keys Used for Entering Numbers into the 326 Scientist.....	8
4	Keys Used in Performing Arithmetic Operations.....	16
5	Reduction of Angles to Equivalent Value.....	32
6	Keys Used to Store, Recall and Exchange Numbers.....	35
7	Store with Arithmetic Operation into a Data Storage Register.....	40
8	Store with Arithmetic Operation into a Data Storage Register and the Entry Register.....	42
9	Recall with Arithmetic Operation.....	44
10	Geometric Solution of Vector Addition Problem	46
11	Keys Used with Functions	50
12	Relationship Between Rectangular and Polar Coordinates.....	62
13	Distance Traveled by a Projectile.....	74
14	Converting Three Dimensional Rectangular to Spherical Coordinates.....	75
15	Great Circle Distance Between Los Angeles and New York.....	79
16	Keys and Switches Used in Programming.....	83
17	Flowchart of Introductory Program	88
18	Hollow Sphere.....	90
19	Flowchart: Calculating the Volume of a Hollow Sphere	91
20	Flowchart: Labeled Program for Sinh(X).....	96
21	Flowcharts: Labeled Programs for Cosh (X) and Tanh(X)	97
22	Branching to a Subroutine	106
23	Flowchart: Main Program and Subroutine.....	108
24	Flowchart: Sum of Integers from 1 to 17.....	114
25	Flowchart: Has an Entry Been Made?.....	117
26	Flowchart: Testing Values Against a Nonzero Number	120
27	Flowchart: Evaluating a Polynomial.....	129
28	Monroe 392 Tape Cassette Drive	134

Section 1 Introduction

29	Connecting the 326 Scientist and the 392 Tape Cassette Drive	136
30	Monroe Endless Tape Cassette: Labeled and Warning Sides	137
31	Write Protect Tabs on Conventional and Endless Tape Cassettes	138
32	Flowchart: One-Block Overlay	151
33	Flowchart: Selecting One of Two Possible Overlay Blocks.....	152
34	Flowchart: Reading and Writing More Than 12 Data Storage Registers...	153
35	Flowchart: Program That Uses Overlay Techniques.....	155
36	Flowchart: Programmed Writing of Data Registers onto Tape.....	161

The Monroe 326 Scientist

The Monroe 326 Scientist (Figure 1) is an incredibly powerful yet compact computer. You can hold it in your hand, yet it will assist you to simplify and solve a wide range of scientific, mathematical and engineering problems.

The 326 Scientist is easy to use. Computations can be performed directly on the keyboard using clearly marked keys and switches. It handles and computes numbers to an accuracy of 13 significant digits plus a sign and exponent. Results are shown in a 16-digit luminous display. Your 326 Scientist may be operated either with rechargeable batteries or a standard AC power supply, whichever is more convenient for you.

You may perform the following types of keyboard computations:

- Arithmetic (including Add, Subtract, Divide, Multiply and Raise to a Power).
- Programmed Functions (including reciprocal, square root, factorial, trigonometric functions and their inverses, logarithmic functions and their inverses, conversion between rectangular and polar coordinates, statistical functions and summation, and many others).
- Metric/English and English/Metric Conversions (including conversions for units of distance, mass, weight, volume, area, temperature, density, pressure, velocity, acceleration and others).

A wide variety of computational results can be achieved by pressing one key or a simple sequence of keys.

Computed data may be stored in any of 12 available data registers and recalled at the touch of a key.

The 326 Scientist is very easy to use and problems are solved from the keyboard much as you would organize and solve the problem yourself. You may compute expressions using up to four levels of parentheses and many features are provided that enable you to simplify otherwise long and complicated computations. The keyboard capabilities of the 326 Scientist are described completely in Section 2, *Basic Operations*, and Section 3, *Functions*, of this manual.

The true power of the 326 Scientist lies in the ease of use of its extensive programming capabilities. You don't have to learn a complicated computer language; you write a program simply by pressing the keys corresponding to the operations you

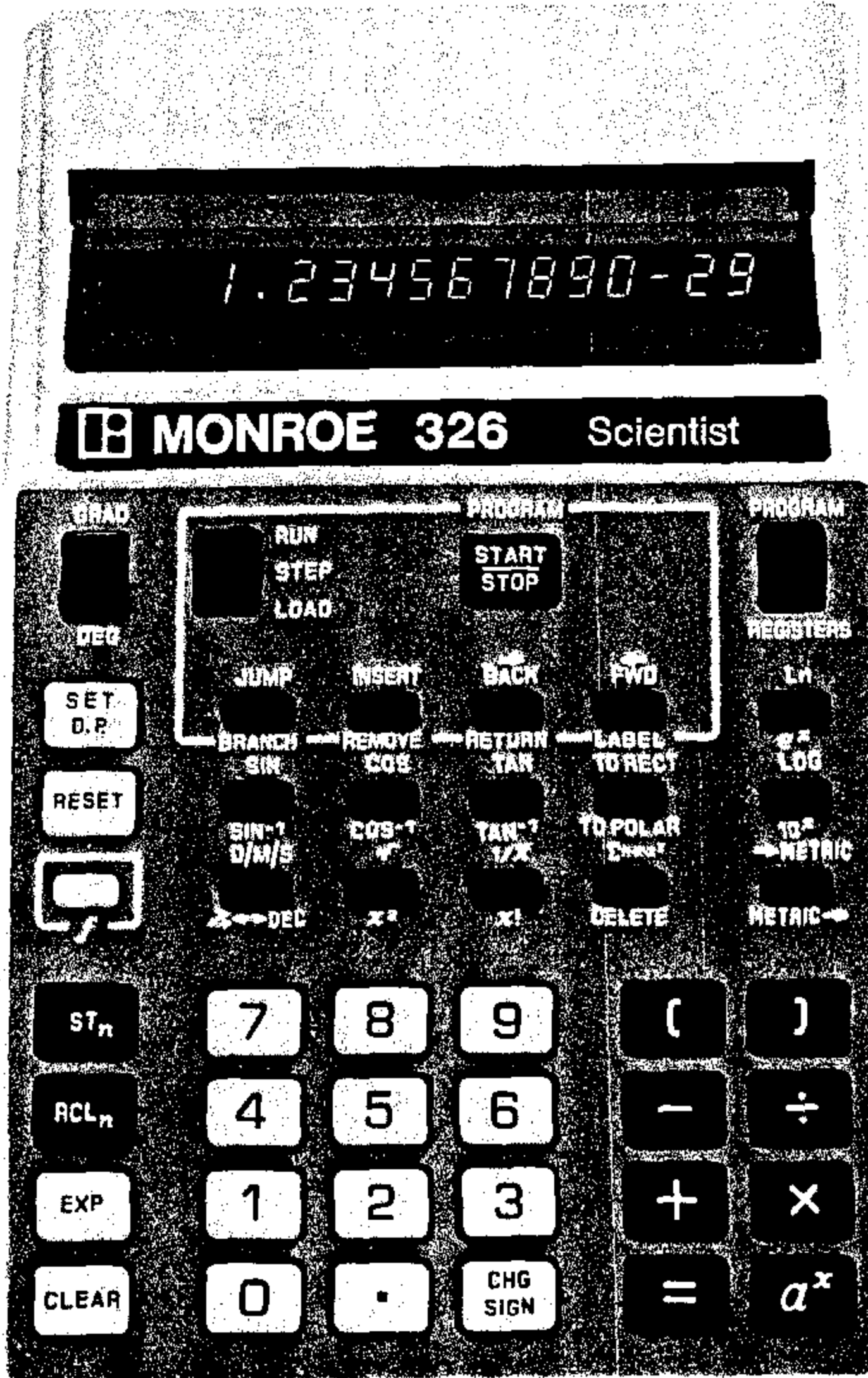


Figure 1

desire. You can specify a program or exact sequence of operations for the 326 Scientist to store in its memory. It can then perform this program automatically at your command. Computations can be easily repeated with different sets of data and choices of different computational procedures can be made automatically, depending on the value of calculated results. The full programming capabilities and features are described in Section 4 of this manual, *Programming*.

The Monroe 392 Tape Cassette Drive

Programming capability is extended greatly by the use of the Monroe 392 Tape Cassette Drive in conjunction with the 326 Scientist (see Figure 2). Prepared programs, from Monroe's extensive Program Library that covers many scientific, mathematical and engineering applications, may be entered into the computer from magnetic tape cassettes. In addition, the programs you can create from the keyboard may be written onto a magnetic tape cassette for later entry into the computer's memory. Since the Tape Cassette Drive's reading and writing operations may be controlled automatically by a program, you may use one program to cause another to be read into the computer's memory. In this way the *effective* size of the computer's memory can be increased to tens of thousands of instructions.

Computed results can also be written from data storage registers onto a magnetic tape cassette, or read from a tape into the computer's data registers. The full flexibility and extent of programming with the Tape Cassette Drive is described in Section 5 of this manual, *Using the 392 Tape Cassette Drive*.



Figure 2

How to Use This Manual

This manual is intended to help you learn to use the full keyboard, programming and magnetic tape capabilities of the 326 Scientist. Sections 2-5 of this manual present the basic principles of operation and how to use the 326 Scientist and 392 Tape Cassette Drive. A series of Appendixes provide further technical information on such matters as the accuracy of computed results, battery recharging, error conditions, and a summary of keys and switches describing their use and pointing to the appropriate page in the text for further information. A complete index, alphabetical by subject, will enable you to use this manual easily as a reference work, after you have become familiar with the 326 and 392.

Some hints on how to study this manual effectively may assist you in becoming expert in the use of the 326 more rapidly.

Do not pass by any undefined terms. Educational research has shown that full comprehension of the basic terminology of a subject is vital to understanding the subject. Thus, while you are reading this book, if your attention drifts or you become aware that you have not understood the material you have just read, it is virtually certain that you have passed one or more *words or symbols* for which you had no definition or an inadequate definition. You can locate the misunderstood word or symbol in this fashion. Go back to the point in the text where you understood the material and "felt good" about it. Then read forward from there. Shortly after that point you will find the misunderstood word or symbol. Get it defined, either by reference to the index of this book to find where the term is defined or by looking in a good dictionary. Try this technique. It works and saves a considerable amount of time in assimilating information.

Secondly, make constant use of your 326 Scientist as you read the manual for the first time. Work the examples and sample problems provided in the manual, and if you wish, generate additional examples from your particular type of application.

Become comfortable with each feature and technique described before moving on the next topic. Study the features of the computer from a personal viewpoint. Ask yourself when reading about a topic "How can I apply this feature to solving the problems of particular interest to me?"

And now you are ready to learn the basic techniques for using the 326 Scientist.

May you enjoy learning how to operate and get the most out of your 326 Scientist.

Getting Started

Turn the Scientist on using the ON/OFF switch located on the back of the computer. The display will have the following appearance:

0.0000

The Model 326 displays four decimal places when it is first turned on. Other decimal settings can be specified using the **SET D.P.** key. (**SET D.P.** is described later in this section.)

Entering Numbers

When a number is entered from the keyboard, it is placed in a register called the *Entry register* (E-register) and also shown in the display.

The Entry register always contains 13 digits. The display will show up to 12 digits.

The area indicated in Figure 3 includes all the keys that are used to enter numbers in the 326 Scientist.

Enter numbers by simply pressing the digit keys 0-9 in the appropriate sequence one at a time. If a second key is pressed before the first key is released, both entries will be accepted in proper sequence if the first key is released before the second.

The decimal point **.** may be pressed when needed. The entered number goes directly into the E-register and appears in the display.

Example:

To enter the number 17.026

Do This

1 7 . 0 2 6

See This

17.026

For large numbers, the 326 Scientist punctuates the displayed value with commas in the appropriate places.

Example:

To enter the number 12,345,678

Do This

1 2 3 4 5 6
7 8

See This

12,345,678.0000

If a decimal point is not entered, it is assumed to follow the last digit entered.

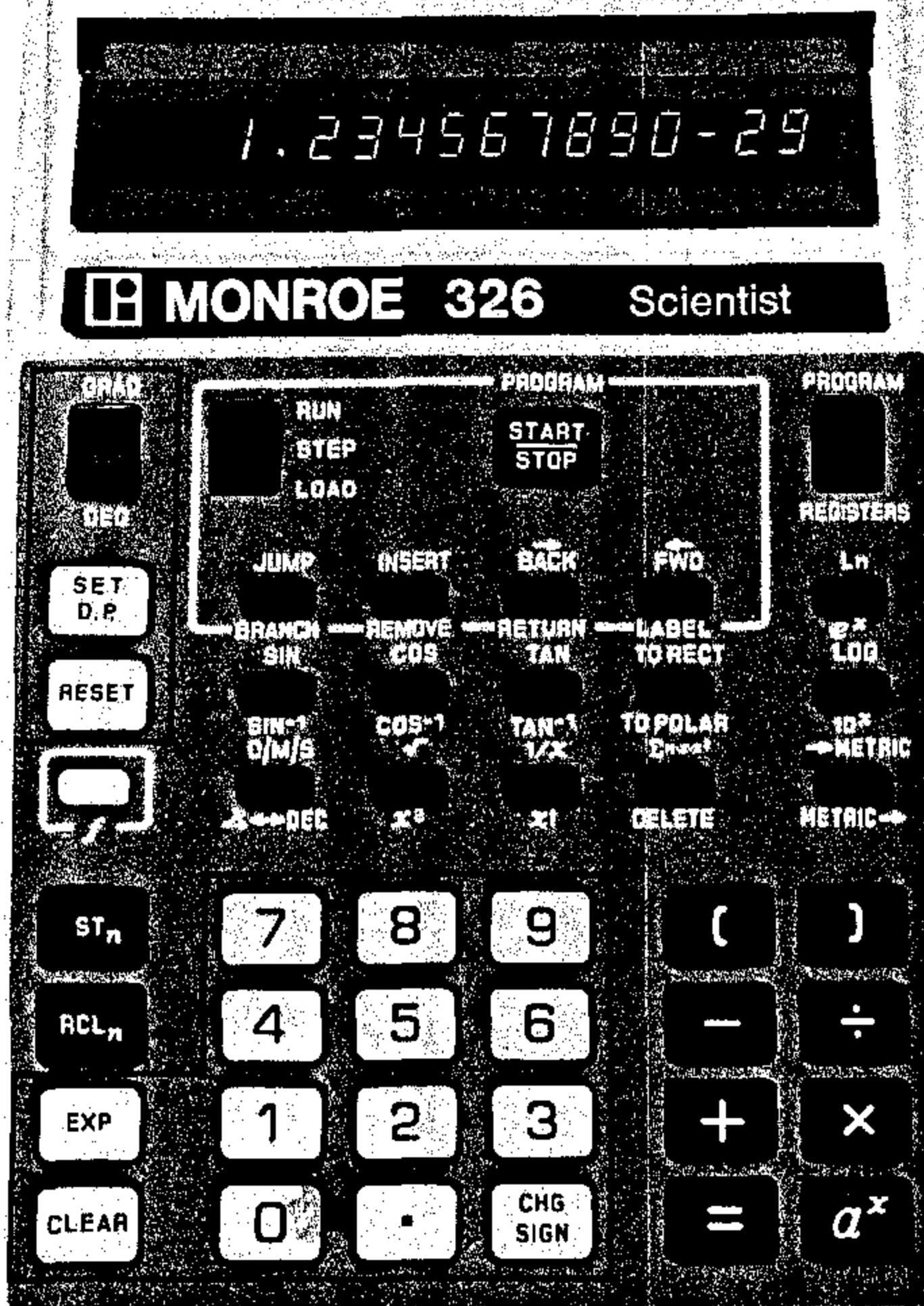

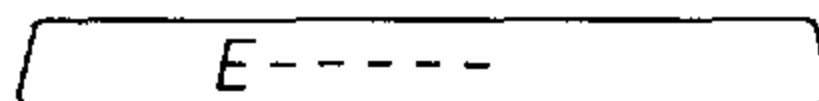



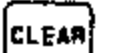
Figure 3

Clear





This key fills the Entry register and display with zeros. Use  if you have made an incorrect entry and wish to re-enter the number correctly.

Also, if you obtain an error indication in the display,

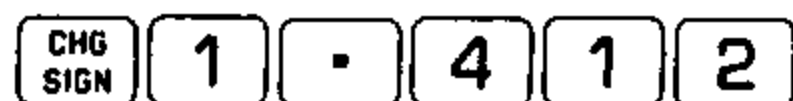


press  to display the number that was in the Entry register prior to the error. Errors are the result of illegal operations which are listed in Appendix E. The  key will be discussed further in the section on Arithmetic Operations.

Change Sign

The  key is used to change the sign of a number in the Entry register. To enter a negative number, you may press  at any time while entering the digits of the number. Repeated use of  during the entry of a number will cause the number to alternate between positive and negative. In the following examples, the  key is pressed at different points during numeral entry.

Do This



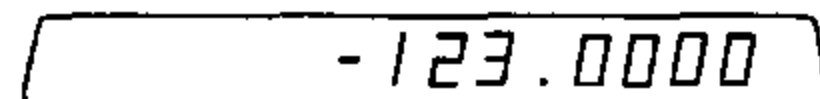
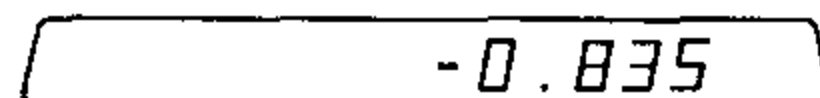
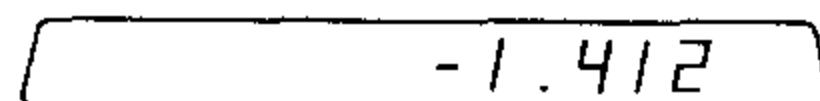













See This



Setting the Decimal Point

 followed by one of the digit keys ( - ) establishes how many digits to the right of the decimal point are normally displayed. For example   will cause two decimal places to be displayed. The full 13-digit number is retained in the Entry register and is used in subsequent calculations. Therefore, the decimal point setting has no effect on the accuracy of calculations. Suppose the number

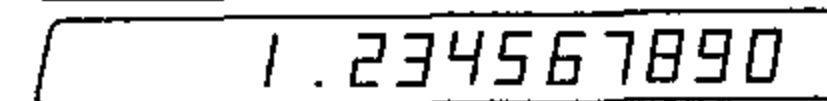
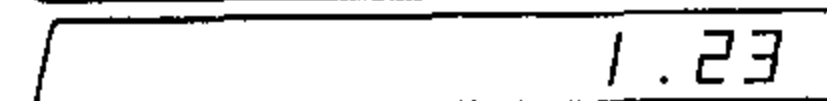
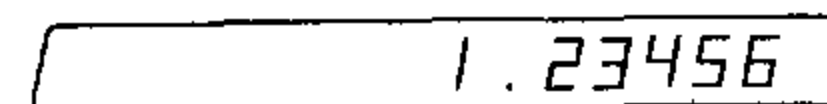
1.234567890



is in the Entry register. Different decimal point settings will affect the display as follows:

Do This



See This



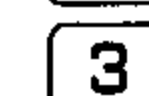
The Scientist automatically moves the decimal point if the present decimal setting will not display any significant digit. For example, if   has been specified and the number 0.000002 results from a calculation or is entered from the keyboard, the 326 will temporarily move the decimal point to allow one significant digit to be displayed.

For numbers between zero and one, you can enter up to 11 leading zeros after the decimal point and still have room to enter 13 digits.

Example:

Enter 0.000001234567890123 (Five leading zeros and 13 significant digits)

Do This



See This



All 13 significant digits are placed in the Entry register and the first six significant digits are shown in the display.

Exponent EXP

You may enter numbers in scientific notation, that is expressed as a *base number* multiplied by ten raised to an *exponent*:

$$a \times 10^b$$

where *a* is the base number and *b* is the exponent. For example, the number 243 could be expressed as 2.43×10^2 . The base number is 2.43 and the exponent is 2.

The procedure is as follows:

1. Enter the base number, digit by digit (with CHG SIGN if the base number is negative).

You may enter as many digits as you like. The first 13 significant digits will be retained in the Entry register.

2. Press EXP

3. Enter the one or two digit exponent (with CHG SIGN if the exponent is negative).

Note that pressing CHG SIGN after EXP changes only the sign of the exponent. A maximum of two digits can be entered for the exponent. If you attempt to enter more than two, the last two digits entered are retained, which enables you to correct a mistake in entering the exponent without having to clear the base number.

If you do not enter a base number before pressing EXP, a base number of one is assumed.

If you enter a base number and press EXP, but then do not enter an exponent, an exponent of 0 is assumed.

Example:

Numbers in scientific notation are entered as follows:

Number	Do This	See This
1	1 EXP	1.0000000000 00
6.023×10^{23}	6 . 0 2 3 EXP 2 3	6.0230000000 23
-35,780	CHG SIGN 3 . 5 7 8 EXP 4	-3.5780000000 04
0.00035	3 . 5 EXP CHG SIGN 4	3.5000000000 -04
0.00035	3 5 EXP CHG SIGN 5	35.0000000000 -05

In the last two examples, the number .00035 is entered and displayed in two different ways to emphasize that you may enter numbers in the form most convenient to you. The 326 Scientist will correctly interpret the value of the number that you enter.

As an added convenience, the 326 Scientist can accept numbers that you enter in any format and express them in standard scientific notation. This is described in the next paragraph.

SET D.P. EXP causes all numbers placed in the entry register to be automatically transformed to standard scientific notation, that is, having the form

$$a \times 10^b$$

where the base number, *a*, is a positive or negative number of absolute value greater than or equal to one and less than ten ($1 \leq |a| < 10$), and *b* is a positive or negative integer exponent.

The 326 Scientist *displays* numbers in scientific notation as follows:

$$n.nnnnnnnnn pp$$

where *n.nnnnnnnnn* are the first 10 significant digits of the 13-digit base number. If the base number is negative, a minus sign precedes the first digit.

pp is the two digit integer exponent. If the exponent is negative, a minus sign is displayed immediately preceding the exponent. Thus, you may enter the number

$$43,870$$

using the key sequence,

4 3 8 7 0

and it will be displayed as;

4.3870000000 04

which means;






$$4.387 \times 10^4$$

Some examples of the entry and display of numbers using SET D.P. EXP follow:

Do This	See This	Comments
CLEAR SET D.P. EXP	0.0000000000 00	Display zero in scientific notation
3 0 0	3.0000000000 02	Enter and display 300
CLEAR 4 EXP 3	4.0000000000 03	Enter and display 4000
CLEAR CHG SIGN . 5	-5.0000000000 -01	Enter and display -0.5





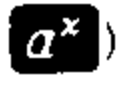
Basic Arithmetic and the Number/Operator Register

The basic arithmetic operations are as follows:





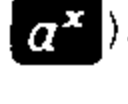

Key	Operation
	Add
	Subtract
	Multiply
	Divide
	Raise a to the power x

These keys and several others related to arithmetic operations are highlighted in Figure 4.




To use the 326 Scientist most effectively, you should be familiar with how the 326 performs these arithmetic operations for you.

In addition to the Entry register there is another register called the Number/Operator register (or simply Num/Op register). When you enter a number it is placed in the Entry register. When an arithmetic operation key (    ) is pressed, the number in the Entry register is duplicated and placed in the Num/Op register along with the arithmetic operator.

The number and operator in the *Num/Op register* changes only when you do one of the following:

1. Press another arithmetic key (, , ,  or .
2. Press .

The contents of the *Entry register* changes when you do one of the following:

1. Enter a new number.
2. Press .
3. Press the next arithmetic key in a sequence of operations (see the section on Chain Arithmetic).
4. Press a function key (refer to the section on Functions).
5. Press  or .

You may enter numbers in a form convenient to you and the Display will automatically show the number in scientific notation. For example, if you want to convert a number

220 Megohms (1 Megohm = 10^6 Ohms)

to its equivalent in Ohms, you simply enter it as follows:

Do This

See This



The answer is 2.2×10^8 Ohms.

Regardless of decimal point setting the 326 will automatically express all entries and answers of absolute value greater than or equal to


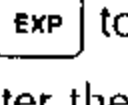







$$1 \times 10^{12}$$

or less than

$$1 \times 10^{-9}$$

in scientific notation.

Summary of Rules for Entering Numbers

- Enter numbers by pressing one digit key at a time.
- Thirteen digits are retained in the Entry register; a maximum of 12 digits is displayed.
- Use  to enter the decimal point. Only one decimal point may be entered.
- Use  to enter numbers in scientific notation:
 1. Enter the base number.
 2. Press .
 3. Enter the power of ten (the exponent) using either one or two digits.
- Use  to change the sign of a number in the Entry register from positive to negative or negative to positive.
 1.  may be used at any time during numeral entry.
 2. When entering numbers in scientific notation,  affects the sign of the *base number* before  has been pressed;  affects the sign of the *exponent* after  has been pressed.

1.234567890-29

MONROE 326 Scientist

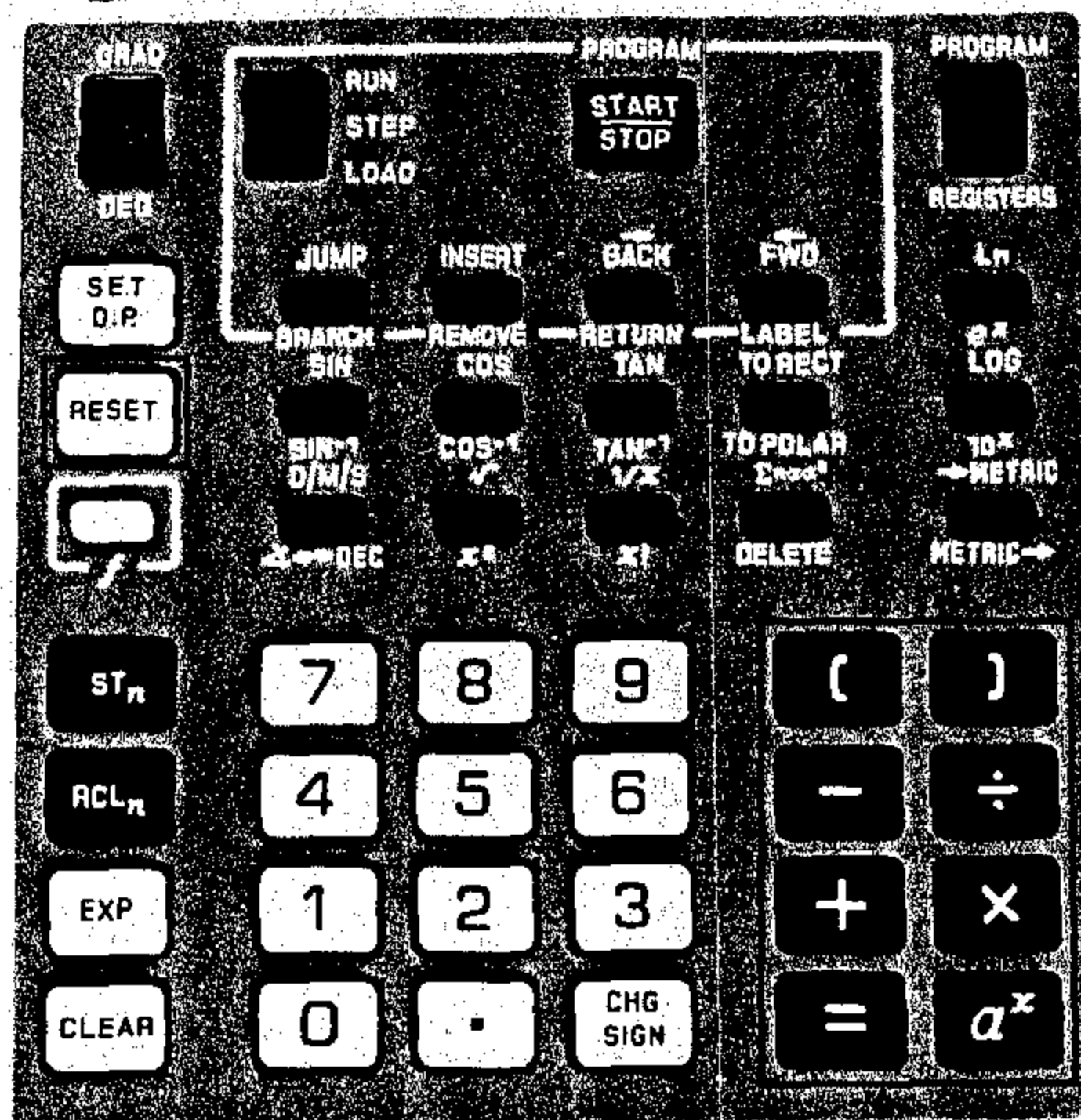


Figure 4

Reset **RESET**

RESET places zeros in the Entry register and the display. It also erases the contents of the Number/Operator register. In effect it removes both entered numbers and operators.

RESET may be used after an error condition occurs.

Differences Between Reset and Clear

RESET removes an error condition, places zeros in the Entry register, and erases the contents of the Number/Operator register.

CLEAR when used after an error condition, **CLEAR** removes the error condition and displays the number that was in the Entry register when the illegal operation was attempted. Pressing **CLEAR** again places zeros in the Entry register and display.

When there is no error condition, **CLEAR** simply places zeros in the Entry register.

Equal Sign **=**

= is used to complete a sequence of arithmetic operations. The answer is placed in the Entry register and the Display. **=** does not affect the contents of the Number/Operator register.

Basic Arithmetic Procedure

The procedure for doing simple calculations is straightforward and conforms to the way you would normally calculate numbers.

The steps are:

1. Enter the first number.
2. Enter the arithmetic operation.
3. Enter the second number.
4. Press **=** to obtain the answer.

Instead of step 4, another arithmetic operation may be entered, followed by another number and then **=**; you may build an extensive sequence of arithmetic operations, terminated finally by **=**. This will be discussed in the section on Chain Arithmetic.

Addition

The procedure for adding two numbers:

1. Enter the first number.
2. Press **+**.
3. Enter the second number.
4. Press **=** (to obtain the sum).

Example:

To find $2 + 3$

Do This

2 **+** **3** **=**

See This

5.0000

Let's look at what is happening in the Entry and Number/Operator registers during this addition.

Calculation	Do This	Entry Register Contents	Num/Op Register Contents	Comments
	2	2	Blank	The Num/Op is blank when the 326 is first turned on.
	+	2	2+	The number 2 is duplicated in the Num/Op register with the + operator.

Calculation	Do This	Entry Register Contents	Num/Op Register Contents	Comments
	3	3	2+	Num/Op doesn't change.
$2 + 3$	=	5	2+	2+ in the Num/Op register is combined with the 3 in the Entry register to give 5 which is placed in the Entry register. Num/Op doesn't change.

Subtraction

The procedure for subtracting two numbers:

1. Enter the number from which a subtraction is to be made (minuend).
2. Press **-**.
3. Enter the number to be subtracted (subtrahend).
4. Press **=** (to obtain the difference).

Example:

To find $10 - 3$

Do This

1 **0** **-** **3** **=**

See This

7.0000

Multiplication

The procedure for multiplying two numbers:

1. Enter the number to be multiplied (multiplicand).
2. Press **x**.
3. Enter the multiplier.
4. Press **=** (to obtain the product).

Example:

To find 7×6.3

Do This

7 **x** **6** **.** **3** **=**

See This

44.1000

Division

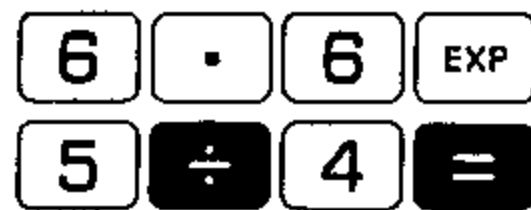
The procedure for dividing one number by another:

1. Enter the number to be divided (dividend).
2. Press \div
3. Enter the number by which the first number is to be divided (divisor).
4. Press $=$ (to obtain the quotient).

Examples:

To calculate $6.6 \times 10^5 \div 4$

Do This



See This

165,000.0000

To calculate $-3 \div 2$

Do This



See This

-1.5000

Chain Arithmetic

Chain arithmetic means a sequence of arithmetic operations, which is terminated by an $=$. Each time a new arithmetic operator is entered, the intermediate answer is placed in the Entry register and displayed because of an implied equals in each arithmetic operator. The new contents of the Entry register are also duplicated in the Number/Operator register and held there with the new operator.

The following example shows what happens in the Entry register and the Number/Operator register during a chain arithmetic sequence. The display will show the contents of the Entry register at each step of the sequence.

Example:

Calculate $2 \times 3 - 4 \div 5 + 6$

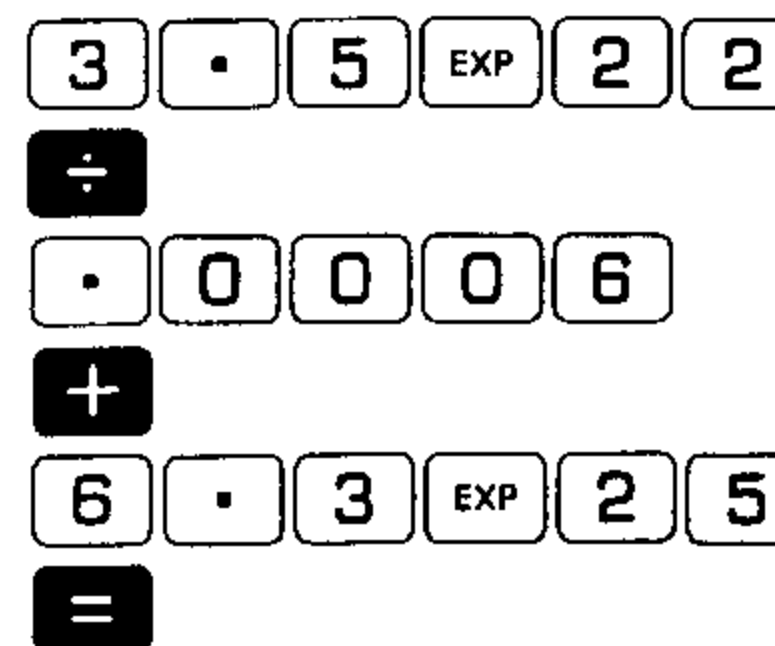
Calculation	Do This	Entry Register	Num/Op Register
	2	2	Blank
	\times	2	2 \times
	3	3	2 \times
$2 \times 3 =$	$-$	6	6 $-$
	4	4	6 $-$
$6 - 4 =$	\div	2	2 \div
	5	5	2 \div
$2 \div 5 =$	$+$	0.4	0.4 $+$
	6	6	0.4 $+$
$0.4 + 6 =$	$=$	6.4	0.4 $+$

In the previous example, each time an arithmetic operator is entered, an $=$ is executed before the arithmetic operation is performed.

Example:

Calculate $3.5 \times 10^{22} \div 0.0006 + 6.3 \times 10^{25}$

Do This



See This

3.5000000000 22
3.5000000000 22
0.0006
5.8333333333 25
6.3000000000 25
1.2133333333 26

Notice how numbers in scientific notation may be easily combined with other numbers in chain arithmetic.

Parentheses (and)

The parentheses keys allow you to specify a particular grouping of operations when you are performing calculations. You use them exactly as you would when writing equations. For example, in the calculation

$$3 + 5 \div 4 = 2$$

the implied = inherent in the \div (as well as other arithmetic operations), causes the $3 + 5$ operation to be executed. The result, 8, and \div are placed in the Num/Op register, giving the final answer $8 \div 4 = 2$. The insertion of parentheses as follows, changes the result.

$$3 + (5 \div 4) = 4.25$$

The parentheses prevent operators outside the parentheses from being executed by the implied = of operators inside the parentheses. The right parenthesis executes an implied = on the last operator inside the parentheses. Hence, $5 \div 4$ is executed, giving 1.25 in the Entry register. Pressing = causes 3+ to operate on 1.25 for the answer of 4.25.

Pressing the keys as follows demonstrates these points.

Do This	See This
3 + 5 ÷	8.0000
4 =	2.0000
RESET	0.0000
3 +	3.0000
(5 ÷ 4)	1.2500
=	4.2500

The appendix on "Parentheses and the Num/Op Register" fully describes how operators inside and outside of parentheses affect intermediate and final results.

The 326 Scientist allows you to "nest" parentheses, as in the example;

$$4 \times (1 + (2 - (3 \times (4 \div 5))))$$

Up to four levels of parentheses are permitted.

Every time you enter a left parenthesis **(**, the parenthesis level is increased by one. Each time you use a right parenthesis **)**, the parenthesis level decreases by one. The parenthesis level is shown at the left side of the display.

If you enter a fifth level of parentheses, an error condition will result, with

E----- in the display. Use **CLEAR** to remove the error condition and return the 326 to the state just prior to the entry of the fifth-level parenthesis.

A right parenthesis will terminate the arithmetic operations for that parenthesis level, and the value resulting from the calculations within those parentheses will be placed in the Entry register and the display.

Example:

Calculate $5 \times ((2 \div 3 - 1) + (3/4))$

Do This

See This

RESET	1 5.0000
5 × (2 1.0000
(2 ÷ 3 - 1	1 -0.3333
)	2 4.0000
+ (3 ÷ 4	1 0.7500
)	0.4166
)	2.0833
=	

In this calculation, the displays where the parenthesis level changes are shown. After each right parenthesis, which forms a closed pair of parentheses, the result of the arithmetic performed inside those parentheses is displayed.

Summary of Rules for Using Parentheses

- For every left parenthesis, there must be a corresponding right parenthesis.
- If you nest parentheses (use one pair within another pair), don't use more than four left parentheses before using a right parenthesis. The current parenthesis level (1, 2, 3, or 4) is indicated on the left side of the display. Press **CLEAR** to remove the error condition resulting from using a fifth-level parenthesis.
- A right parenthesis operates as an = for the arithmetic inside the parentheses.
- Don't use **RESET** inside parentheses. It destroys the entire sequence of arithmetic operations. Use **CLEAR** to correct an incorrect entry.

A more detailed description of how the 326 works internally when parentheses are used is given in Appendix G, "Parentheses and the Number/Operator Register."

Raising a Number to a Power, a^x

a^x is used to raise the number a to the power X .

This key is extremely powerful and convenient to use because of the following features.

- The number a may be any value, integer or decimal fraction, positive or negative.
- The power x may be any value positive or negative, integer or fraction, (with the exception that, when a is negative, x must be an integer).
- Both a and x may be entered or calculated numbers. This allows you to use parenthetical expressions with a^x .

The procedure for a^x is as follows:

1. Enter or calculate a .
2. Press a^x .
3. Enter x or calculate x within parentheses.
4. Press $=$ (to obtain the answer).

The following facts are true for all values of a and x :

$$a^1 = a$$

$$a^0 = 1$$

$$1^x = 1$$

$$0^x = 0$$

In addition, an attempt to raise a negative a to a non-integral (fractional) power will give an error condition. Also, the attempt to raise zero to a negative power will yield an error condition, since that is equivalent to trying to divide by zero.

Example:

To calculate 4^3

Do This

$$4 \ a^x \ 3 \ =$$

See This

$$64.0000$$

To calculate $(3 + 4)^3$

$$3 \ + \ 4 \ a^x$$

$$3 \ =$$

$$7.0000$$

$$343.0000$$

Notice that you do not need to place parentheses around $3 + 4$ in the preceding example because a^x causes an implied $=$ to be executed. This causes the sum of $3 + 4$ to be stored in the Num/Op register with the operator a^x .

The following example shows the use of parentheses with a^x

Example:

To calculate $(3 + (4 \div 2))^{(4 \times 2)}$

Do This

See This

$3 \ +$	
$(\ 4 \ \div \ 2 \)$	2.0000 <small>value of parenthetical expression</small>
a^x	5.0000 <small>value of a</small>
$(\ 4 \ \times \ 2 \)$	8.0000 <small>value of parenthetical expression</small>
$=$	$390,625.0000$

When a^x is pressed, it executes an implied $=$ on $3+$ and the value inside the parentheses (2), placing the answer (5) in the E-register and duplicating, it in the Num/Op register with the operator a^x . Thus, the final $=$ causes the calculation of 5 raised to the power 8.

$$5^8 = 390,625$$

In the next example a number is raised to a non-integral power.

Examples:

To calculate $4^{3.2}$

Do This

See This

$$4 \ a^x \ 3 \ . \ 2 \ = \ 84.4485$$

The 326 Scientist permits the easy calculation of uncommon roots of numbers using the a^x in conjunction with the reciprocal key $1/x$. You don't have to use parenthetical expressions for the exponent to calculate high order roots.

Example:

To find $\sqrt[12]{10}$

Do This

See This

$1 \ 0 \ a^x$	10.0000
$1 \ 2 \ 1/x$	0.0833
$=$	1.2115

The $1/x$ function key may be used in this way because it doesn't contain an implied $=$ and affects only the number in the Entry register. Thus, it doesn't interfere

with a calculation in progress. A full description of \sqrt{x} and other function keys is given in the section on Functions.

As a final example of the power of a^x , we show the combined use of chain arithmetic and nested-parentheses.

Example:

To calculate $(3 \div (4 + 5))^{(6 - (7 \times (8 \div 9))}$

Do This

3 \div
 (4 + 5)
 a^x
 (6 -
 (7 \times
 (8 \div 9
)))
 =

See This

3.0000
 9.0000
 0.3333 value of a
 2 7.0000
 -0.2222 value of x
 1.2765

Constant Arithmetic

The 326 Scientist allows you to repeatedly use a constant number and operator without having to re-enter them for each new calculation. The procedure is as follows:

1. Enter or calculate the constant.
2. Press the operator to be used $+$ $-$ \times \div or a^x .
3. Enter a value of the variable.
4. Press $=$ (to obtain the answer).
5. Enter a new value of the variable.
6. Press $=$ (to obtain the new answer).

This feature results from the use of the Number/Operator register described previously. The contents of the Number/Operator register change only when you enter a new operator ($+$ $-$ \times \div or a^x) or when you press RESET .

Addition Using a Constant

To Add Do This

5 + 2 5 + 2 =
 5 + 4 4 =
 5 + 3 3 =

See This

7.0000
 9.0000
 8.0000

The number 5 and operator + are retained in the Num/Op register.

Subtraction Using a Constant

To Subtract Do This

100-20 1 0 0 -
 2 0 =
 100 - 5.31 5 . 3 1 =
 100 - 90 9 0 =

See This

80.0000
 94.6900
 10.0000

The number 100 and operator - are retained in the Num/Op register.

Multiplication Using a Constant

To Multiply Do This

3 \times 6 3 \times 6 =
 3 \times 5.2 5 . 2 =
 3 \times 7 7 =

See This

18.0000
 15.6000
 21.0000

The number 3 and operator \times are retained in the Num/Op register.

Division Using a Constant Dividend

To Divide Do This

20 \div 2 2 0 \div 2 =
 20 \div 4 4 =
 20 \div 30 3 0 =

See This

10.0000
 5.0000
 0.6666

The number 20 and the operator \div are retained in the Num/Op register.

Division Using a Constant Divisor

You may divide by a constant *divisor* using the $\frac{1}{x}$ (reciprocal) key, since dividing by a number is equivalent to multiplying by the reciprocal of that number. Thus, in the following examples, you divide a series of numbers by the constant 20 by taking the reciprocal of 20 and entering that with the operator \times in the Num/Op register.

To Divide	Do This	See This
$2 \div 20$	$20 \frac{1}{x} \times 2 =$	
$62 \div 20$	$62 \times =$	
$40 \div 20$	$40 \times =$	

The number $1/20$ and the operator \times are retained in the Num/Op register.

Raise a Constant to a Power

To Calculate	Do This	See This
3^2	$3 a^x 2 =$	
3^4	$4 =$	
3^6	$6 =$	

The number 3 and the operator a^x are retained in the Num/Op register.

Repeated use of $+$ \times or $=$

If $+$ is pressed repeatedly, an implied $=$ is executed after each $+$, causing the answer to be placed in the Entry register. The contents of the Entry register are then placed in the Num/Op register with the operator $+$. Thus repeatedly pressing $+$ causes a repeated doubling of the number in the E-register, as shown in the following example.

Calculation	Do This	Entry Register	Num/Op Register	See This
	3	3	Blank	
	$+$	3	3+	
$3 + 3 =$	$+$	6	6+	
$6 + 6 =$	$+$	12	12+	
$12 + 12 =$	$=$	24	12+	

The repeated use of \times causes an implied $=$ to be executed with each \times . The answer is placed in the E-register and duplicated in the Num/Op register with the operator \times . Thus, repeatedly pressing \times causes a repeated squaring of the number in the E-register, as shown in the following example.

Calculation	Do This	Entry Register	Num/Op Register	See This
	2	2	Blank	
	\times	2	2 \times	
$2 \times 2 =$	\times	4	4 \times	
$4 \times 4 =$	\times	16	16 \times	
$16 \times 16 =$	$=$	256	16 \times	

If you press $=$ repeatedly, you will cause the contents of the Num/Op register to operate repeatedly on the successive contents of the Entry register. For example,

Do This	See This
$5 + 1 =$	
$=$	
$=$	
$=$	

The 5+ is retained and operates on each successive value in the Entry register. In many calculations, you can take advantage of these features to save yourself unnecessary keystrokes and time.


Angles

Entering Angles

The 326 Scientist permits you great flexibility in dealing with angles. You may enter angles in any of the following modes:

- Degrees (360 degrees per full circle)
- Grads (400 grads per full circle)
- Radians (2π radians per full circle)

Degrees


Set the  switch to DEG (Degrees). You may enter angles in degrees in one of three ways:

1. Degrees and decimal fraction of a degree (e.g. 30.5 degrees or 30.5°).
2. Degrees and minutes with decimal fraction of a minute (e.g. 54 degrees 7.65 minutes or 54° 7.65').

Note: One degree = 60 minutes

3. Degrees, minutes and seconds with decimal fraction of a second (e.g. 121 degrees 17 minutes 2.3 seconds or 121° 17' 2.3").

Note: One minute = 60 seconds

Entering degrees with minutes and seconds requires the use of the  (Degrees/Minutes/Seconds) key. The procedures for angle entry are as follows:

For degrees and decimal fraction of a degree

1. Enter the number of degrees with decimal fraction as you would any number.

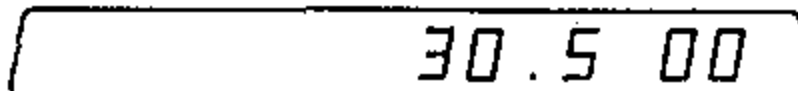
Example:

To enter 30.5°


Do This

See This




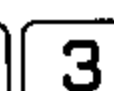
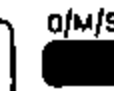
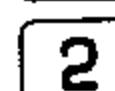



For degrees and minutes with decimal fraction of a minute.

1. Enter the number of degrees.
2. Press .
3. Enter the number of minutes and decimal fraction of a minute.

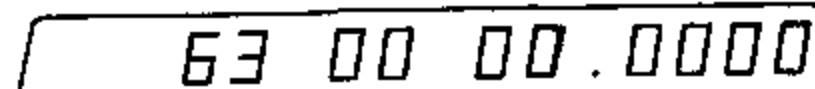
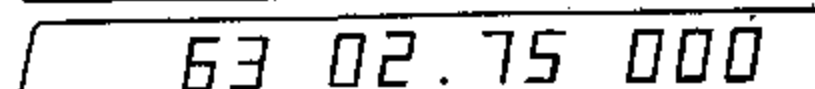
Example:

To enter the value 63° 2.75'



Do This

See This

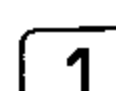



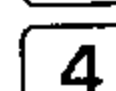
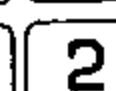

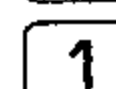



For degrees, minutes and seconds with decimal fraction of a second.

1. Enter the number of degrees.
2. Press .
3. Enter the number of minutes.
4. Press .
5. Enter the number of seconds including decimal fraction.

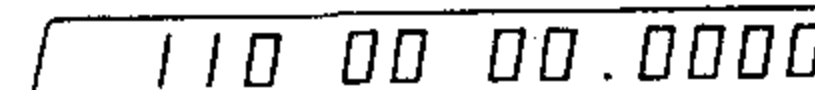
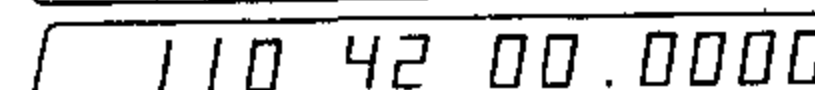
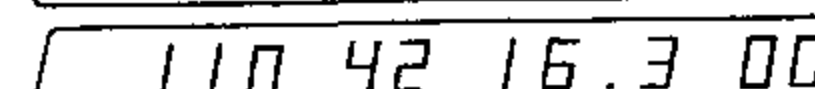
Example:


To enter the value 110° 42' 16.3"




Do This

See This

You may enter positive or negative numbers for angles. The  key is used to change the sign of an angle.

Use of the  key identifies the entry as an angle in the D/M/S format. Whenever that entry is used in further operations, it is treated as a D/M/S angle. The  key is inoperative after  is pressed.

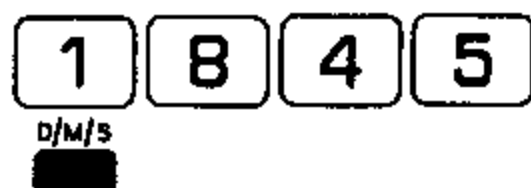
Angles entered in degrees with decimal fraction of a degree are not restricted in their value any more than ordinary numbers.

Angles in D/M/S format may have a maximum of three digits in the degrees column. Entries of 1000° or greater, or -1000° or less are automatically converted to an equivalent angle less than 360° by subtraction of the greatest possible multiple of 360° (or -360° for negative angles).

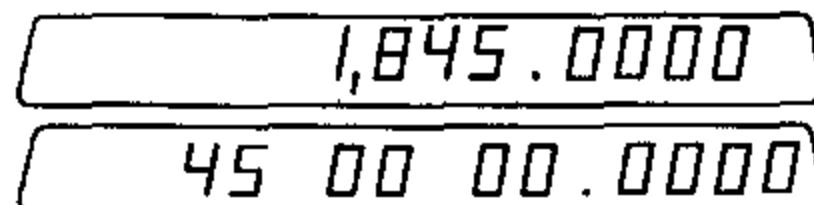
Example:

To enter 1845° in D/M/S format.

Do This



See This



This entry results because 1800° ($5 \times 360^\circ$) is the greatest multiple of 360° contained in 1845° . The remainder after subtraction is 45° .

If the angle to be expressed in D/M/S format is the result of a calculation, values greater than 360° or less than -360° will be expressed as an equivalent angle in the range of 0 to 360° .

Example:

If the value $378^\circ 40'$ is the result of a calculation, the 326 Scientist will reduce it to the equivalent angle $18^\circ 40'$. These two angles are shown in the accompanying figure. The angles shown are measured counterclockwise from the X-axis.

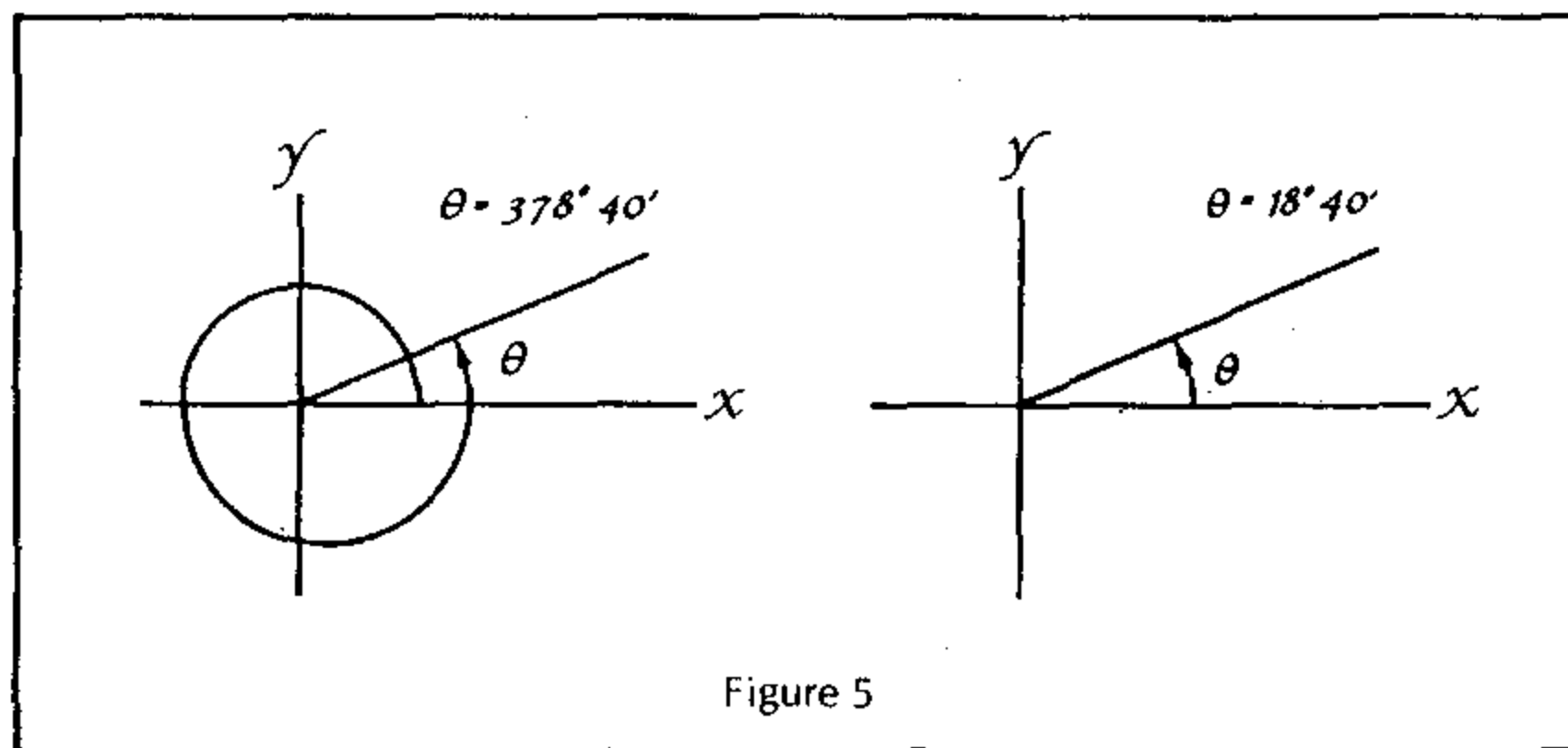



Figure 5

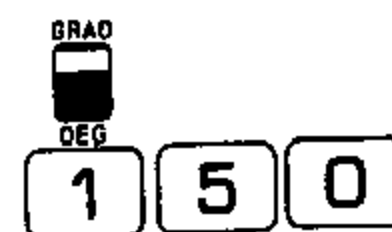
Grads

Set the  switch to GRAD. There are 400 grads to a full circle and 100 grads to a quadrant ($90^\circ = 100$ grads). Enter angles in grads as you would any number including decimal fraction.

Example:



To enter the value 150 grads

Do This



See This

**Radians**

The GRAD/DEG switch may be set to  or . There are 2π radians in a full circle (2π radians = 360°). Enter angles in radians as you would enter any number including decimal fraction.

Angles entered in radians may be converted to grads or degrees, using the function key sequences described in the section on Functions.

Arithmetic with Angles

Arithmetic operations can be done with angles expressed in degrees and decimal fractions, D/M/S format, grads or radians. Angles may be entered at any time during arithmetic operations. Arithmetic with angles is governed by the following rules.

Operation	Result
Angle + Angle	Angle
Angle - Angle	Angle
Number \times Angle	Angle
Angle \div Angle	Number
Angle \div Number	Angle

When the 326 Scientist performs arithmetic with angles in D/M/S format, it will automatically express the answers in standard form: with degrees in the range of 0- 360° , and minutes and seconds both in the range of 0-60. For example, an answer like

$$372^\circ 120' 63''$$

Would be converted to the equivalent answer in simpler form

$$14^\circ 1' 3''$$

since

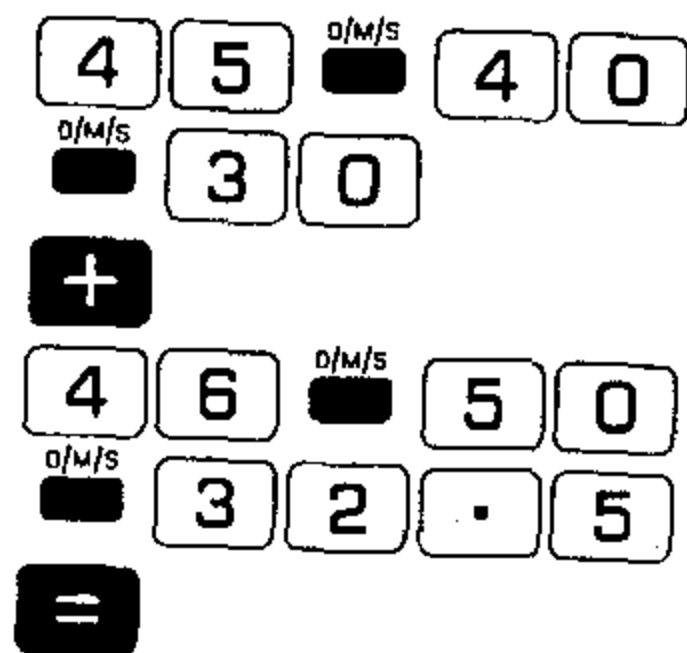
$$\begin{aligned} 372^\circ &= 360^\circ + 12^\circ \\ 120' &= \quad \quad 2^\circ + 0' + 0'' \\ 63'' &= \quad \quad \quad 1' + 3'' \\ \hline &= 360^\circ + 14^\circ + 1' + 3'' \end{aligned}$$

The following example shows addition of two angles in D/M/S format with reduction of the answer to its simplest D/M/S form.

Example:

To calculate $45^\circ 40' 30'' + 46^\circ 50' 32.5''$

Do This



See This

45 40 30.0000

46 50 32.5 00

92 31 02.5000

Converting Angles

The 326 Scientist permits you to convert angles from degrees with decimal fractions to degrees, minutes and seconds, and vice versa. You may also convert between degrees and radians, and between grads and radians. These conversions are accomplished easily by the use of *function keys* and are fully described in the section on "Functions".

Data Storage Registers

The 326 Scientist provides 12 data storage registers which can be used to hold numbers for later review or use in calculations. Each of these 12 data storage registers is identified and addressed by one of the numeral or symbol keys: **0** - **9**, **.** or **CHG SIGN**.

There are several basic ways in which you can use these data storage registers, as follows:

1. You can store a number in a data storage register (designated **0** - **9**, **.** or **CHG SIGN**). By store, we mean to duplicate the contents of the Entry register in the specified data storage register. The contents of the E-register are not changed

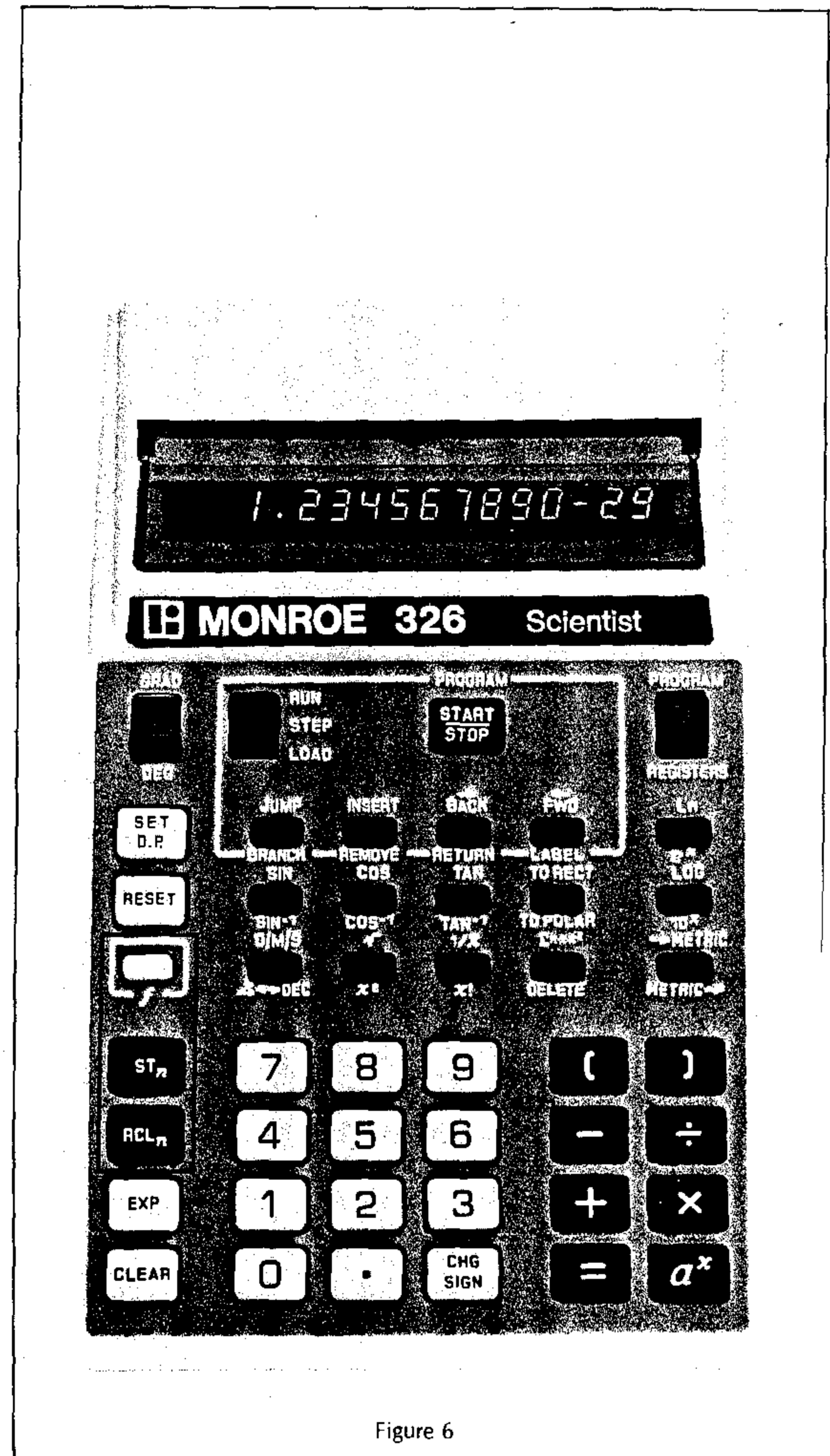


Figure 6

by the act of storing data into a data storage register. However the previous contents of the data storage register are replaced by the number that is being stored.

The **CLEAR** and **RESET** keys have no effect on the data storage registers. The numbers stored in data storage registers are protected for your future use.

You may store numbers during the course of an arithmetic calculation and the calculation will not be affected.

2. You can *recall* a number from a specified data storage register into the Entry register. By *recall*, we mean to duplicate the contents of the specified data storage register in the E-register. The Recall operation does not change the contents of the data storage register. The previous contents of the E-register are replaced by the recalled number. You may recall numbers during the course of an arithmetic calculation without affecting the calculation any more than a number entered from the keyboard would.
3. You can *exchange* the number in the E-register with the number in a specified data register. By *exchange*, we mean that the contents of the E-register become the new contents of the specified data register, and the previous contents of the specified data register are placed in the E-register.

Store **ST_n**

The procedure for storing a number from the Entry register into a specified data storage register is as follows:

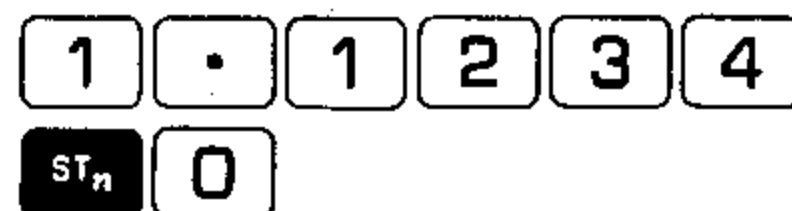
1. Enter the number to be stored if it is not already present in the Entry register as the result of a calculation.
2. Press **ST_n**.
3. Press one of the ten numeral keys **0** - **9**, **.** or **CHG SIGN**.

If a number is already in the specified data storage register, it is automatically replaced by the newly stored number.

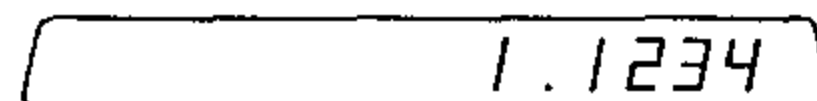
Example:

To store 1.1234 into data storage register 0

Do This



See This



Recall **RCL_n**

The procedure for recalling a number from a specified data register to the E-register is as follows:

1. Press **RCL_n**.
2. Press one of the ten numeral keys **0** - **9**, **.** or **CHG SIGN**.

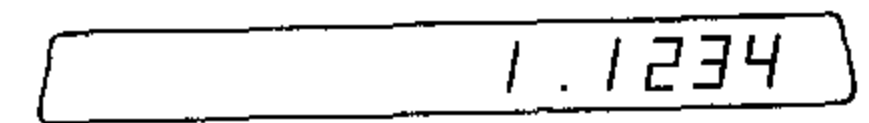
Example:

To recall the previously stored contents of register 0

Do This



See This



The recalled number in the Entry register can be used just as if it had been entered from the keyboard.

Exchange **f** **ST_n**

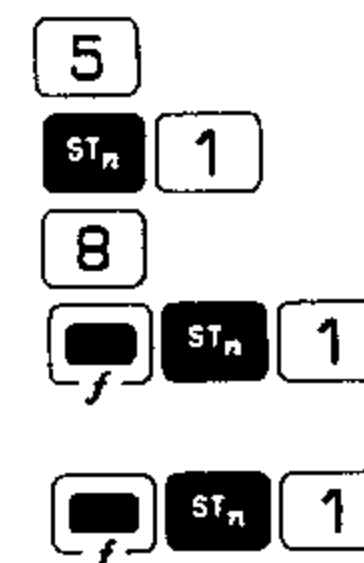
This operation exchanges the number in the Entry register with the number in a specified data storage register. The procedure is as follows:

1. Press **f**. (The function key **f** will be fully discussed in the section on Functions.)
2. Press **ST_n**.
3. Press one of the ten numeral keys **0** - **9**, **.** or **CHG SIGN**.

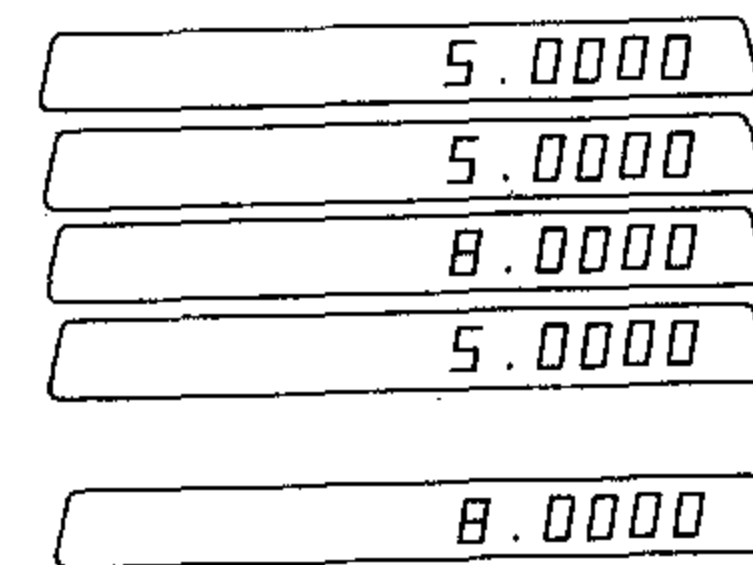
Example:

Store the number 5 in register 1. Then enter 8.0 in the E-register and exchange the contents of register 1 with the E-register.

Do This



See This



Comments

- Enter 5
- Store 5 in register 1
- Enter 8
- Exchange contents of E-register and data storage register 1
- Exchange again

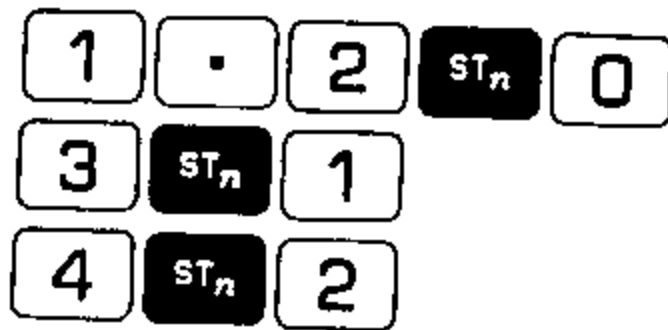
Using the Contents of Registers in a Calculation Sequence

You may use the contents of registers in a calculation sequence. The following example demonstrates this capability.

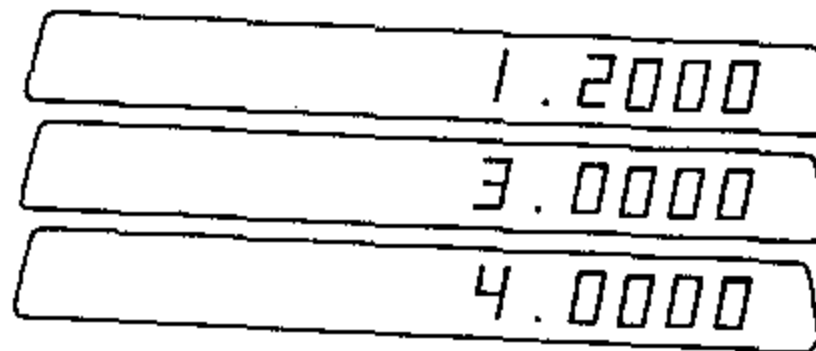
Example:

Store the numbers 1.2, 3 and 4 into data storage registers 0, 1 and 2, respectively. Then recall these stored numbers to use in various calculations.

Do This



See This

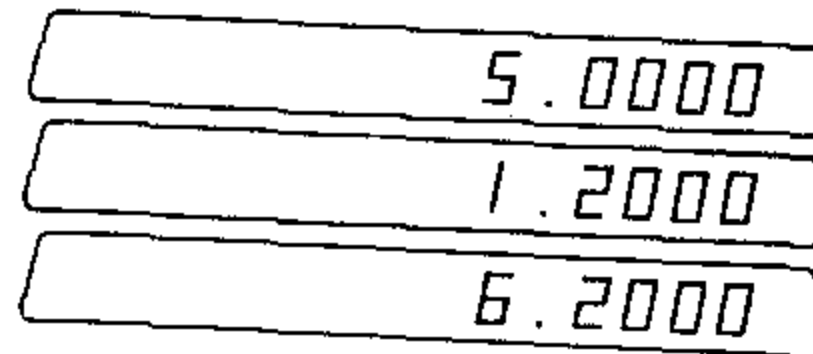


To calculate $5 + 1.2$

Do This

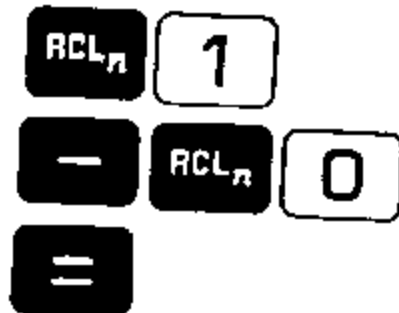


See This

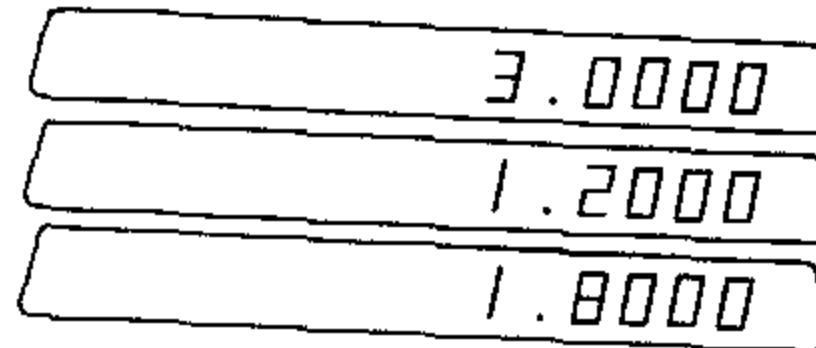


To calculate $3 - 1.2$

Do This

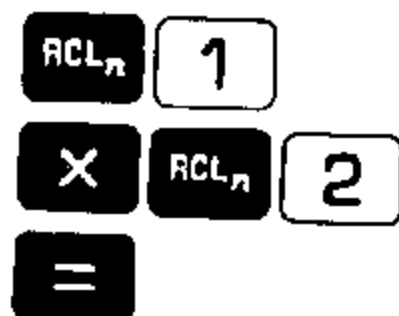


See This

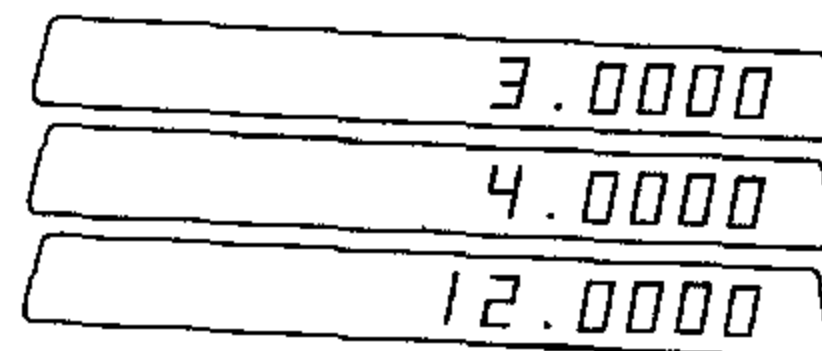


To calculate 3×4

Do This

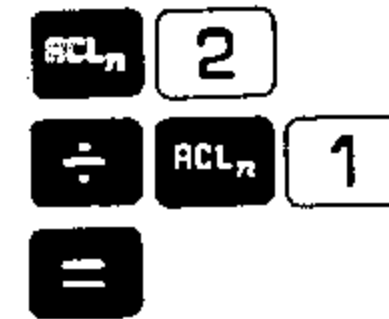


See This

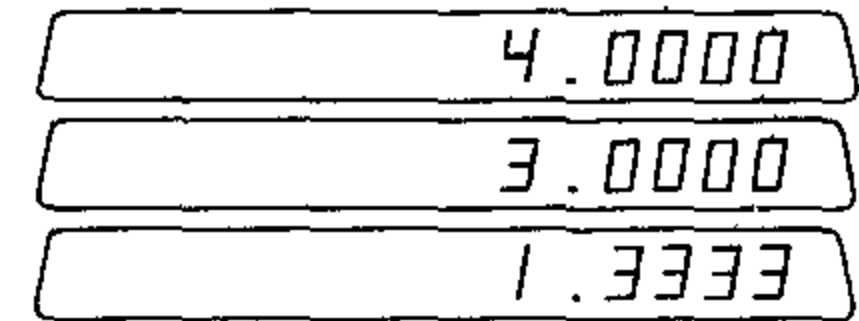


To calculate $4 \div 3$

Do This

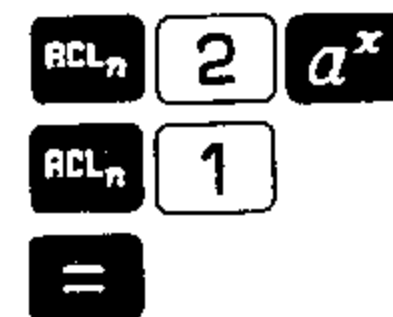


See This

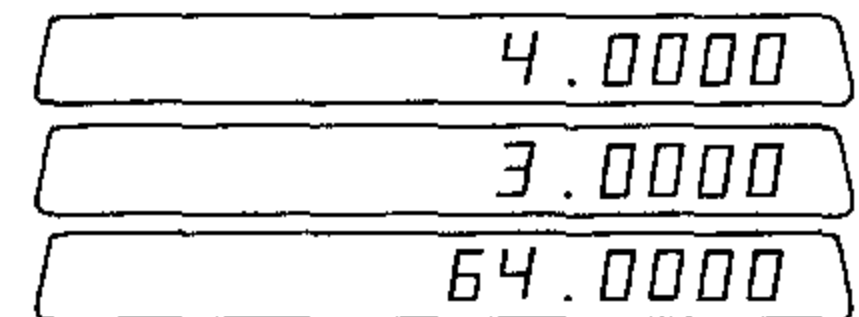


To calculate 4^3

Do This

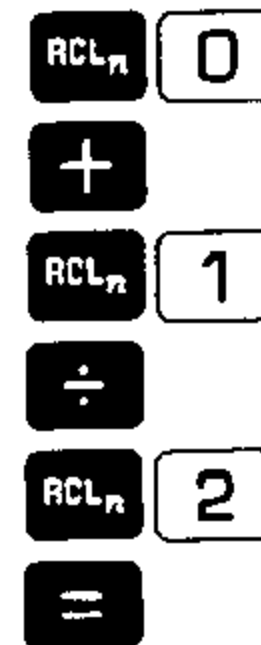


See This

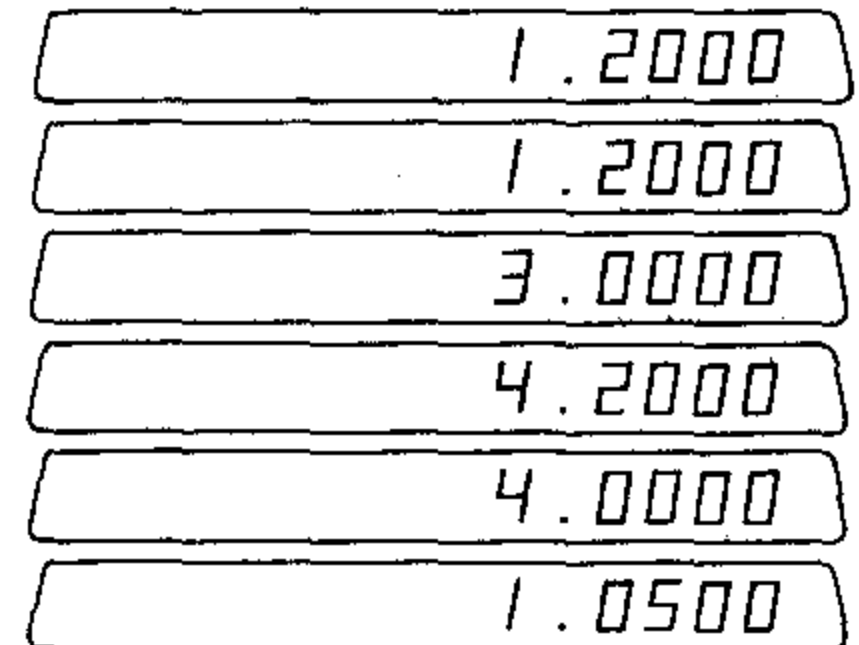


To calculate $1.2 + 3 \div 4$

Do This



See This



Direct Register Arithmetic

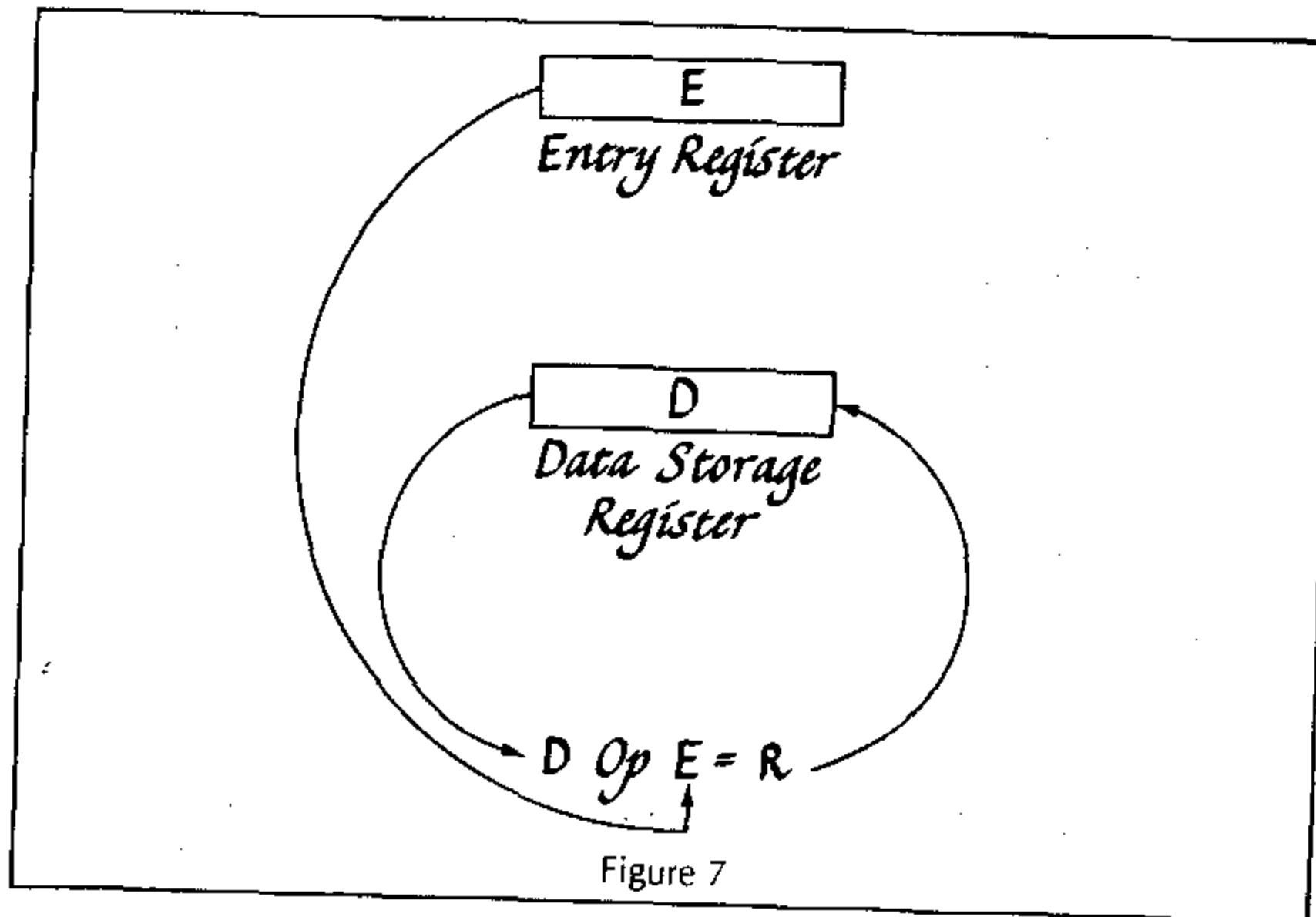
The 326 Scientist allows you to perform arithmetic directly in the data storage registers. You may combine a number in a specified data storage register with the number in the Entry register using any of the five arithmetic operators ($+$, $-$, \times , \div , and a^x). The result of the arithmetic operation can be left in the data storage register, the E-register or both places. This type of register arithmetic does not require that you recall a number into the E-register to perform arithmetic with it.

There are three basic types of direct register arithmetic:

- ST_n with Arithmetic Operator
- f ST_n with Arithmetic Operator
- RCL_n with Arithmetic Operator

ST_n with Arithmetic Operator

This type of register arithmetic uses the ST_n key in conjunction with a specified arithmetic operator and data storage register. The number in the indicated data storage register is operated on by the specified operator and the number in the E-register. The result is left in the data storage register and the number in the E-register is unchanged.



This is shown in Figure 7, where the number D in the data register is affected by the operator Op and the number E from the E-register. The result R is placed in the data register.

The procedure is as follows:

1. Press ST_n .
2. Press an arithmetic operator key ($+$, $-$, \times , \div or a^x).
3. Press the numeral or symbol key for the data storage register in which the arithmetic will be performed ($0-9$, \cdot or $CHG SIGN$).

The ST_n arithmetic operations have the following meanings:


- ST_n $+$ Adds the number in the Entry register to the number in the specified data storage register. Leaves the sum in the data storage register.
- ST_n $-$ Subtracts the number in the Entry register from the number in the specified data storage register. Leaves the difference in the data storage register.
- ST_n \times Multiplies the number in the Entry register by the number in the specified data storage register. Leaves the product in the data storage register.
- ST_n \div Divides the number in the Entry register into the number in the specified data storage register. Leaves the quotient in the data storage register.
- ST_n a^x Raises the number in the specified data register to the power indicated by the number in the E-register. Leaves the result in the data storage register.

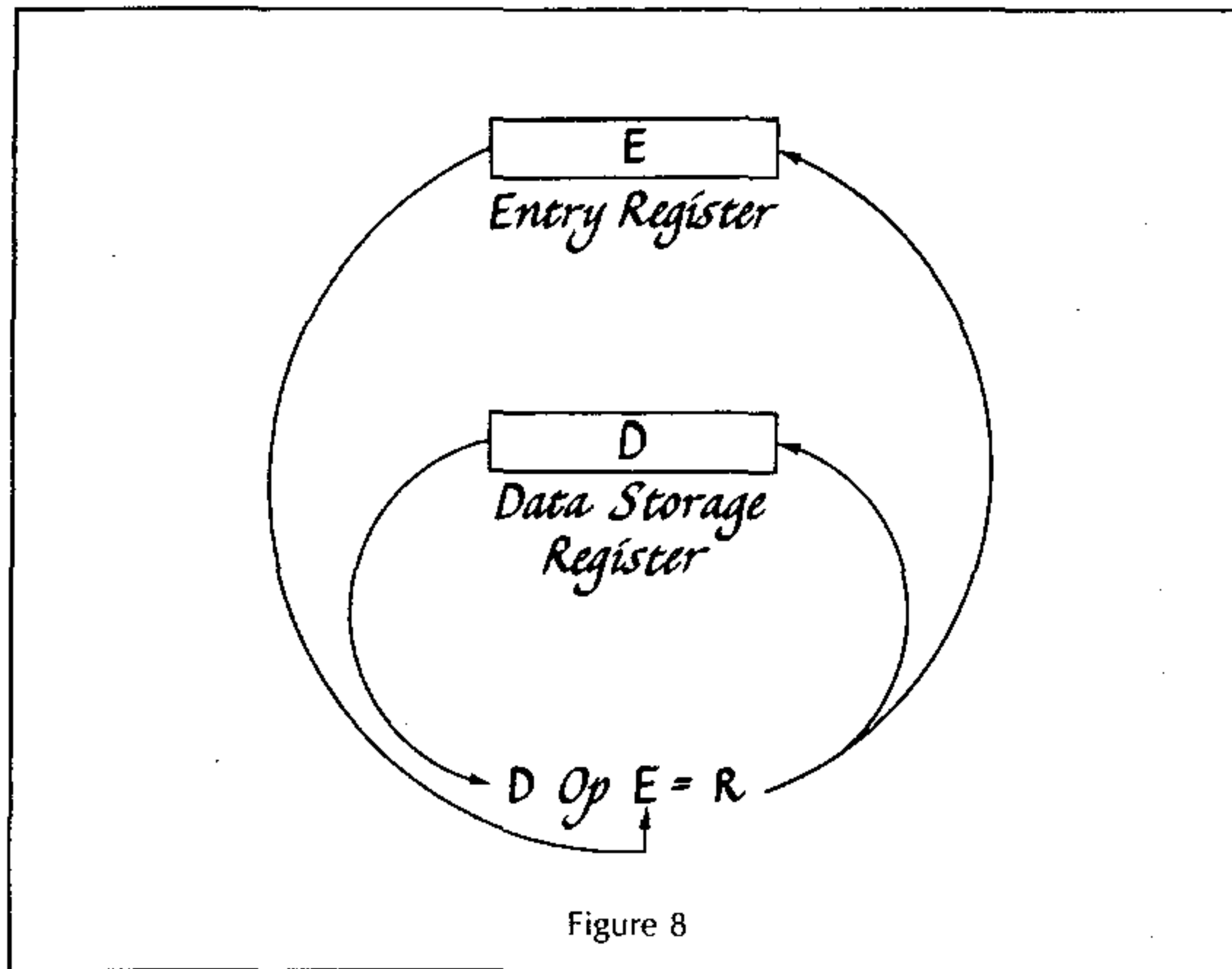
Example:

The following example demonstrates the use of ST_n arithmetic.


Do This	See This	Comments
2 ST_n 0	2.0000	Store 2 in data register 0
3 ST_n $+$ 0	3.0000	Add 3 to number in register 0
RCL_n 0	5.0000	Recall sum from register 0
1 ST_n $-$ 0	1.0000	Subtract 1 from the number in register 0
RCL_n 0	4.0000	Recall difference from register 0
3 ST_n \times 0	3.0000	Multiply 3 by the number in register 0
RCL_n 0	12.0000	Recall product from register 0
6 ST_n \div 0	6.0000	Divide 6 into the contents of register 0
RCL_n 0	2.0000	Recall quotient from register 0
3 ST_n a^x 0	3.0000	Raise the contents of register 0 to the third power
RCL_n 0	8.0000	Recall answer from register 0


ST_n with Arithmetic Operator






This type of register arithmetic is identical to " ST_n With Arithmetic Operator" except that the result is left not only in the specified data register but the E-register as well. Figure 8 shows the operation of  ST_n with register arithmetic. The number D in the indicated data register is affected by the arithmetic operator Op and the number E from the Entry register. The result R is placed both in the data storage register and the Entry register.




The procedure is as follows:


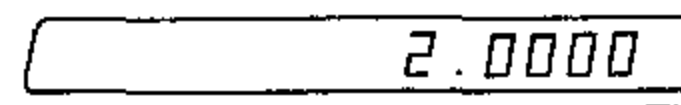

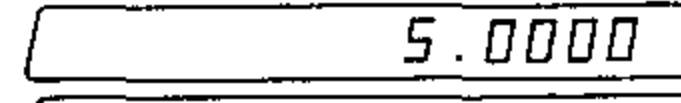
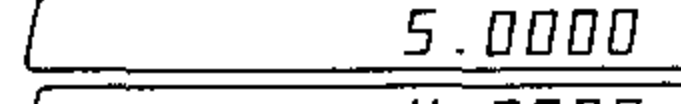

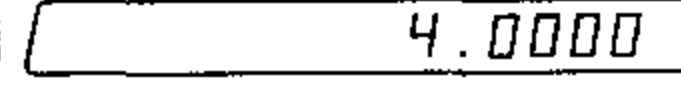


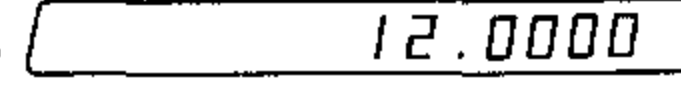
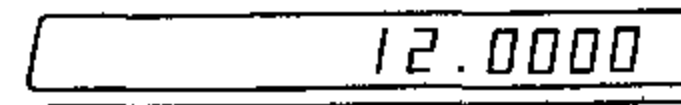

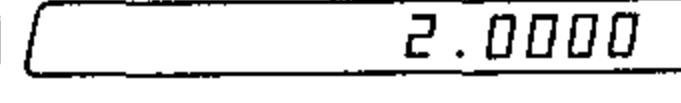
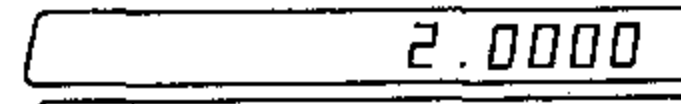

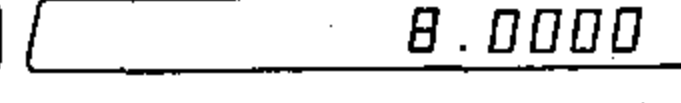
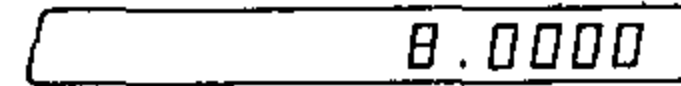
1. Press  ST_n .
2. Press an arithmetic operator key ($+$, $-$, \times , \div or a^x).
3. Press the numeral or symbol key for the data storage register in which the arithmetic will be performed. ($0-9$, \cdot or $CHG SIGN$).

The  ST_n arithmetic operations have the following meanings:

-  ST_n $+$ Adds the number in the Entry register to the number in the specified data storage register. Leaves the sum in the data storage register and the Entry register.
-  ST_n $-$ Subtracts the number in the Entry register from the number in the specified data storage register. Leaves the difference in the data storage register and the Entry register.
-  ST_n \times Multiplies the number in the Entry register by the number in the specified data storage register. Leaves the product in the data storage register and the Entry register.
-  ST_n \div Divides the number in the Entry register into the number in the specified data storage register. Leaves the quotient in the data storage register and the Entry register.
-  ST_n a^x Raises the number in the specified data register to the power indicated by the number in the E-register. Leaves the result in the data storage register and the Entry register.

Example:

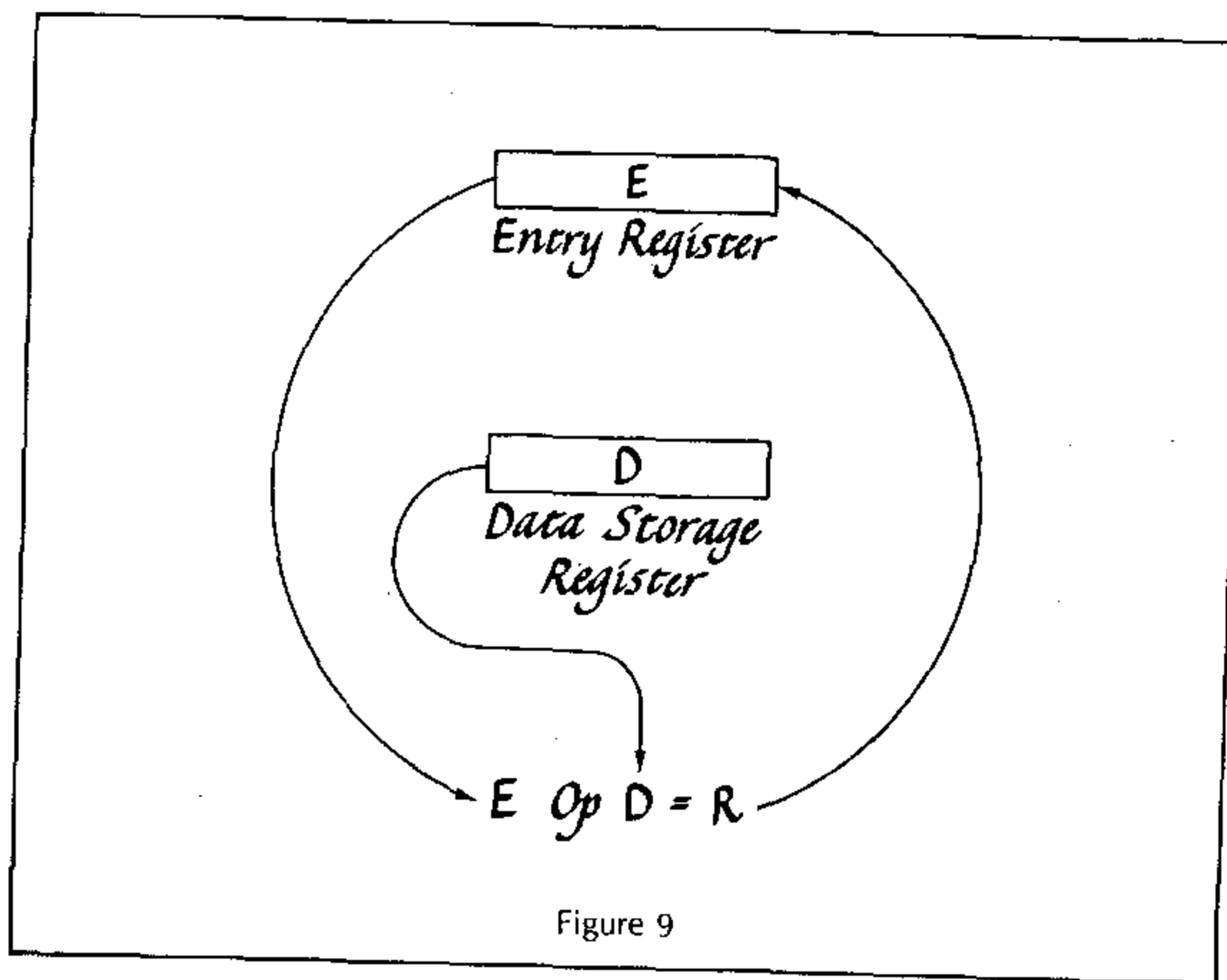
The following calculations show the use of  ST_n arithmetic.

Do This	See This	Comments
2  ST_n 0		Store 2 in data register 0
3  ST_n $+$ 0		Add 3 to number in register 0
RCL_n 0		Recall number from register 0
1  ST_n $-$ 0		Subtract 1 from number in register 0
RCL_n 0		
3  ST_n \times 0		Multiply 3 by the number in register 0
RCL_n 0		
6  ST_n \div 0		Divide 6 into the number in register 0
RCL_n 0		
3  ST_n a^x 0		Raise the number in register 0 to the power 3
RCL_n 0		

RCL_n with Arithmetic Operator

This type of register arithmetic uses the **RCL_n** key in conjunction with a specified arithmetic operator and data storage register. The number in the E-register is operated on by the indicated operator and the number in the specified data storage register. The result is left in the E-register.

This is shown in Figure 9, where the number E from the Entry register is affected by the operator *Op* and the number D from the specified data storage register. The result R is placed in the Entry Register.



The procedure is as follows:

1. Press **RCL_n**.
2. Press an arithmetic operator key (**+**, **-**, **×**, **÷** or **a^x**).
3. Press the numeral or symbol key for the data storage register that contains the number that will operate on the contents of the E-register with the specified operation. (**0** - **9**, **.** or **CHG SIGN**).

The **RCL_n** arithmetic operations have the following meanings:

- RCL_n +** Adds the number in the specified data storage register to the number in the E-register. Leaves the sum in the E-register.
- RCL_n -** Subtracts the number in the specified data storage register from the number in the E-register. Leaves the difference in the E-register.
- RCL_n ×** Multiplies the number in the specified data storage register by the number in the E-register. Leaves the product in the E-register.
- RCL_n ÷** Divides the number in the specified data storage register into the number in the E-register. Leaves the quotient in the E-register.
- RCL_n a^x** Raises the number in the E-register to the power indicated by the number in the specified data storage register. Leaves the result in the E-register.

Example:

Perform the following **RCL_n** arithmetic calculations.

Do This	See This	Comments
3 ST_n 0	<input type="text" value="3.0000"/>	Store 3 in register 0
4	<input type="text" value="4.0000"/>	Enter 4 in E-register
RCL_n + 0	<input type="text" value="7.0000"/>	Add the number in register 0 to the number in the E-register
RCL_n × 0	<input type="text" value="21.0000"/>	Multiply the number in register 0 by the number in the E-register
RCL_n - 0	<input type="text" value="18.0000"/>	Subtract the number in register 0 from the number in the E-register
RCL_n ÷ 0	<input type="text" value="6.0000"/>	Divide the number in register 0 into the number in the E-register
RCL_n a^x 0	<input type="text" value="216.0000"/>	Raise the number in the E-register to the power indicated by the number in register 0

Throughout the calculation, the results are placed in the E-register while the number in register 0 remains unchanged.

Sample Application of Register Arithmetic

A practical example of the use of register arithmetic is in performing vector addition. (Vectors are quantities having both magnitude and direction and a vector in a plane can be described in terms of two rectangular components: the X and Y components.)

Suppose we wish to find the vector sum of three vectors A, B, and C, which lie in the same plane. Figure 10 shows the geometric method of adding these three vectors to form a resultant vector D. The vector D may also be calculated as follows:

1. The X-component of D is the sum of the X-components of A, B and C.
2. The Y-component of D is the sum of the Y-components of A, B and C.

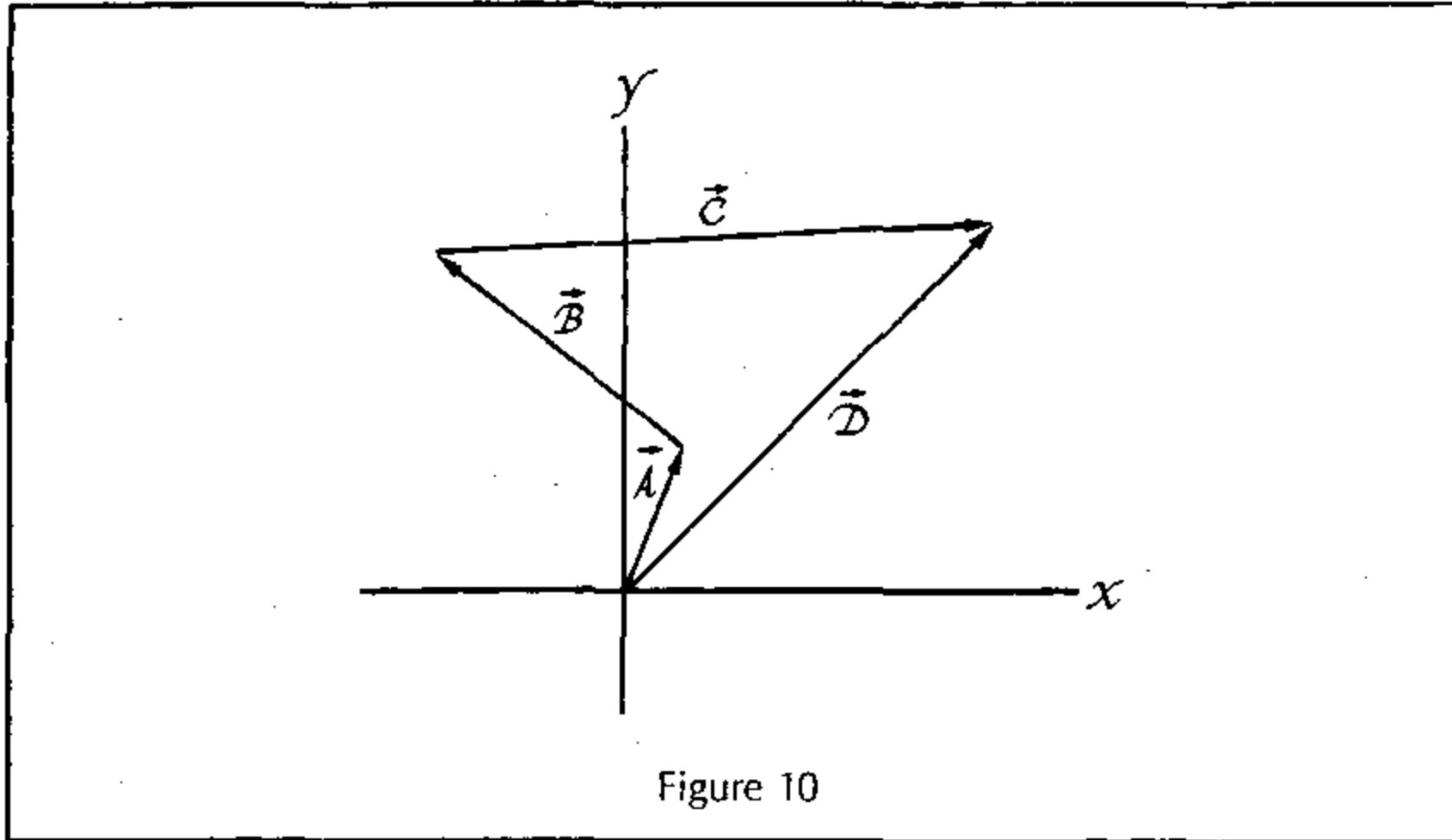


Figure 10

The vectors have the following components.

Vector	X-Component	Y-Component
A	2	5
B	-10	6
C	21	2

Use register 1 to calculate the sum of the X-components and register 2 to calculate the sum of the Y-components.

Do This	See This	Comments
2 ST_n 1	2.0000	X-component of Vector A
5 ST_n 2	5.0000	Y-component of Vector A
CHG SIGN 1 0		
ST_n + 1	-10.0000	X-component of Vector B
6 ST_n + 2	6.0000	Y-component of Vector B
2 1 ST_n		
+ 1	21.0000	X-component of Vector C
2 ST_n + 2	2.0000	Y-component of Vector C
RCL _n 1	13.0000	Recall sum of X-components
RCL _n 2	13.0000	Recall sum of Y-components

The X-component and Y-component of the resultant Vector D are both 13 (see Figure 10).

Note: Direct register arithmetic has no effect on the contents of the Num/Op register. Thus direct register arithmetic may be done while other types of calculations are in progress, without affecting those calculations.

Section 3 Functions

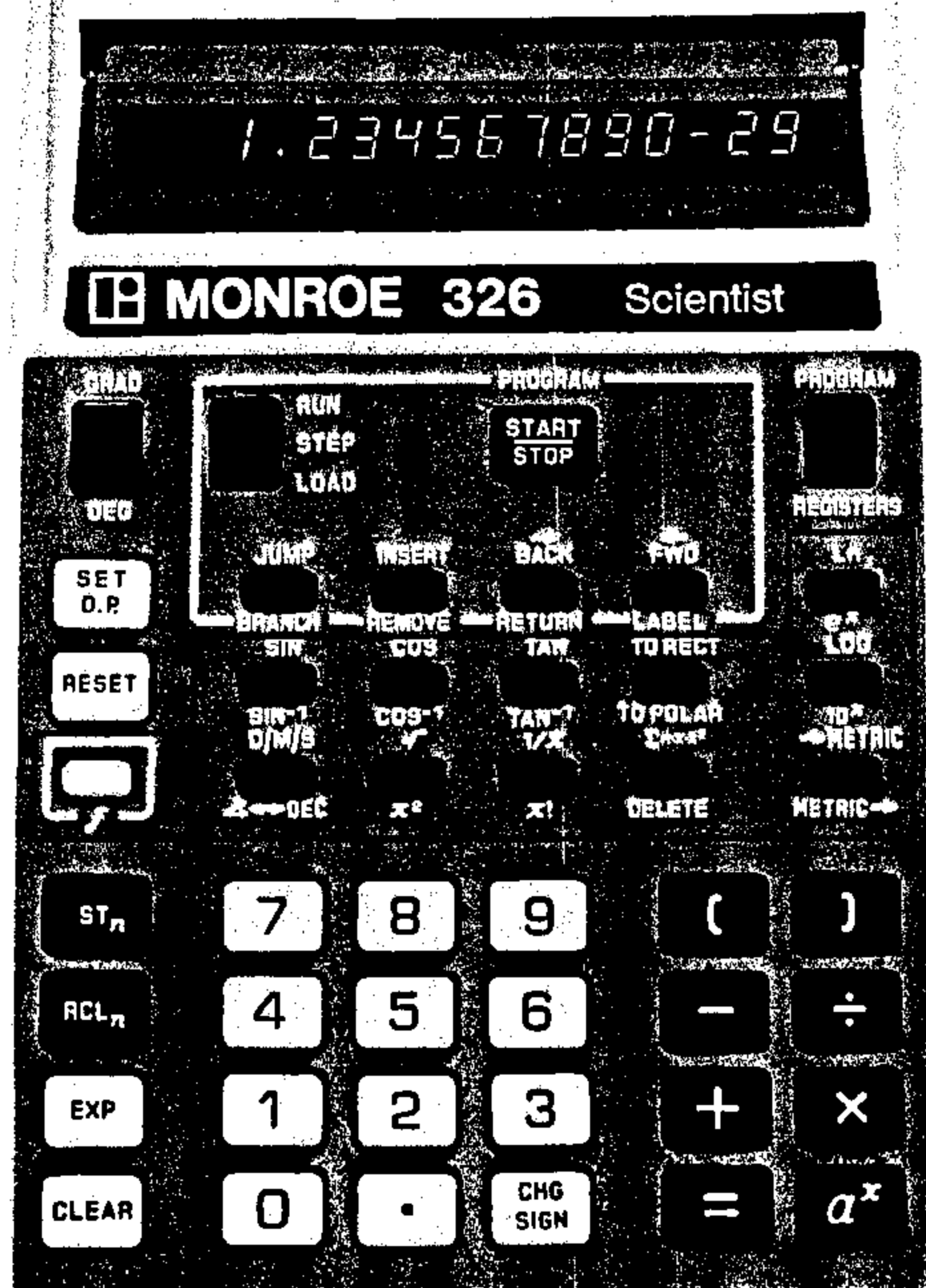




Figure 11

The 326 Scientist provides a number of function keys. (See the indicated area in Figure 11.) Each function key has two legends, one above the key and one below the key. To use the function indicated by the legend above a key, simply press the key. To use the function indicated by the legend below a key, press the  key first and then press the function key. A description of these functions follows.

Reciprocal Function

Pressing  calculates the reciprocal of the number in the Entry register.

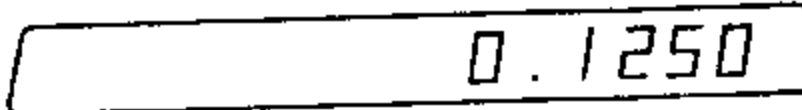
Example:

To calculate the reciprocal of 8

Do This


 

See This



The reciprocal function may be used to simplify many calculations. For example, to evaluate the expression

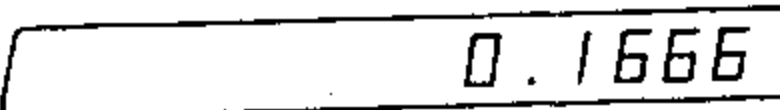
$$\frac{1}{3 \times (4 - 2)}$$


you may calculate the denominator first and then use the  key.


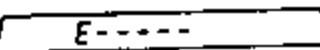
Do This



See This



The  key may be used with angles in D/M/S format. The 326 Scientist automatically converts the angle to degrees and decimal fraction of a degree and takes the reciprocal of that value.

Pressing  when zero is in the Entry register causes an error condition with  in the display.

Factorial

The key sequence   calculates the factorial of the number in entry.

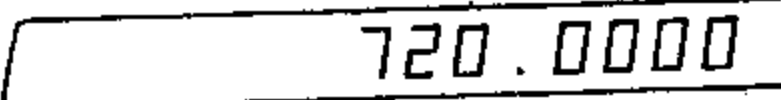
Example:

To calculate 6!

Do This


See This



The following points should be noted:

- The largest factorial that can be calculated is 69!
- The value of 0! is 1
- Attempting to compute the factorial of a non-integer or a negative number gives $E-----$. Press **CLEAR** to display the number previously in the E-register.

Square Root

Pressing  calculates the square root of the number in the Entry register.

Example:

To calculate $\sqrt{9}$

Do This


9 

See This

3.0000

To calculate $\sqrt{(3 \times 4)}$

Do This



3 **×** **4** **=** 

See This

3.4641

Attempting to calculate the square root of a negative number gives an error condition with a display of $E-----$. Press **CLEAR** to display the negative number.

Square

Pressing   calculates the square of the number in the Entry register.

To find 2.3^2


Do This

2 **.** **3**  

See This

5.2900

Logarithm (Base 10)

Pressing  calculates the logarithm to the base 10 (\log_{10}) of the number in the Entry register.

Example:

To calculate \log_{10} of 5

Do This



5 

See This

0.6989

Attempting to calculate the base 10 logarithm of a number less than or equal to zero is illegal (since the logarithm function is not mathematically defined for these values) and gives $E-----$. Press **CLEAR** to display the number previously in the E-register.

Antilogarithm

Press   calculates the antilogarithm to the base 10 of the number in the Entry register.

Example:

To calculate antilog_{10} of 0.5


Do This

. **5**  

See This

3.1622

Logarithm (Base e)

Pressing  calculates the logarithm to the base e (Ln_e) of the number in the Entry register.

Example:

To calculate Ln_e of 5

Do This



5 

See This

1.6094

Attempting to calculate the base e logarithm of a number less than or equal to zero is illegal and gives $E-----$. Press **CLEAR** to display the number previously in the E-register.

Antilogarithm (Base e)

Pressing   calculates the antilogarithm to the base e (or exponential function) of the number in the Entry register.

Example:

To calculate the antilogarithm to the base e of 2

Do This

2  

See This

7.3890

Hyperbolic Functions

You may use the logarithm and antilogarithm to the base e capabilities of the 326 Scientist to calculate all the hyperbolic and inverse hyperbolic functions. The defining equations and key sequences that are used to evaluate these functions are presented in the following tables. In all the key sequences given, it is assumed that the number, x , is present in the Entry register, either having been entered from the keyboard or being the result of a calculation.

Hyperbolic Function	Equation	Key Sequence
Sinh x	$\frac{e^x - e^{-x}}{2}$	f e^x $-$ $1/x$ \div 2 $=$
Cosh x	$\frac{e^x + e^{-x}}{2}$	f e^x $+$ $1/x$ \div 2 $=$
Tanh x	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	f e^x ST_n 0 $-$ $1/x$ \div $($ RCL_n 0 $+$ $1/x$ $)$ $=$
Coth x	$\frac{e^x + e^{-x}}{e^x - e^{-x}}$	f e^x ST_n 0 $+$ $1/x$ \div $($ RCL_n 0 $-$ $1/x$ $)$ $=$
Sech x	$\frac{2}{e^x + e^{-x}}$	f e^x $+$ $1/x$ \div 2 $=$
Csch x	$\frac{2}{e^x - e^{-x}}$	f e^x $-$ $1/x$ \div 2 $=$



Inverse Hyperbolic Function

Inverse Hyperbolic Function	Equation	Key Sequence
Sinh ⁻¹ x	$\text{Ln}(x + \sqrt{x^2 + 1})$ where $-\infty < x < \infty$	$+$ $($ f e^x $+$ 1 $)$ \sqrt $=$ Ln
Cosh ⁻¹ x	$\text{Ln}(x + \sqrt{x^2 - 1})$ where $x \geq 1$	$+$ $($ f e^x $-$ 1 $)$ \sqrt $=$ Ln
Tanh ⁻¹ x	$\frac{1}{2} \text{Ln} \left(\frac{1+x}{1-x} \right)$ where $-1 < x < 1$	ST_n 0 $+$ 1 \div $($ RCL_n 0 CHG $SIGN$ $+$ 1 $)$ $=$ Ln \div 2 $=$
Coth ⁻¹ x	$\frac{1}{2} \text{Ln} \left(\frac{x+1}{x-1} \right)$ where $x < -1$ or $x > 1$	ST_n 0 $+$ 1 \div $($ RCL_n 0 $-$ 1 $)$ $=$ Ln \div 2 $=$
Sech ⁻¹ x	$\text{Ln} \left(\frac{1}{x} + \sqrt{\frac{1}{x^2} - 1} \right)$ where $0 < x \leq 1$	$1/x$ $+$ $($ f e^x $-$ 1 $)$ \sqrt $=$ Ln
Csch ⁻¹ x	$\text{Ln} \left(\frac{1}{x} + \sqrt{\frac{1}{x^2} + 1} \right)$ where $x \neq 0$	$1/x$ $+$ $($ f e^x $+$ 1 $)$ \sqrt $=$ Ln

Functions Involving Angles




When you calculate functions involving angles, such as trigonometric functions, conversions between rectangular and polar coordinates and conversions between radians and degrees or grads, you must specify whether angles will be expressed in degrees or grads. This is controlled by the GRAD/DEG switch.

The GRAD/DEG Switch

The GRAD/DEG switch specifies how angle entries and calculations are made. When the switch is in the  position, all angle entries and calculations are based on the metric angular unit of the grad, with 400 grads in a circle. In the  position, angles are entered and calculated in degrees, with 360 degrees in a circle.

Converting Degrees or Grads to Radians

To convert from degrees to radians, do the following:

1. Set GRAD/DEG switch to .
2. Enter the angle in degrees and decimal fraction or D/M/S format. Alternatively, the angle may be in the E-register as the result of a calculation.
3. Press .
4. Press .

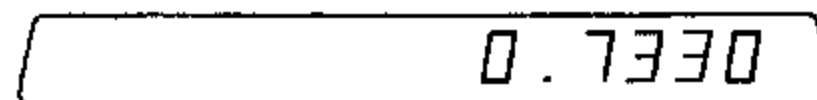
Example:

To convert 42° to radians




Do This



See This



To convert from grads to radians, do the following:

1. Set GRAD/DEG switch to .
2. Enter the angle in grads. Alternatively, the angle in grads may be present in the E-register as the result of a calculation.
3. Press .
4. Press .

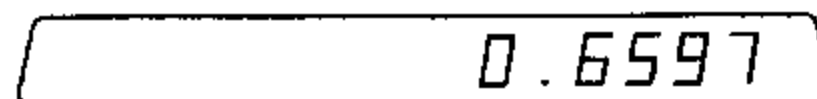
Example:

To convert 42 grads to radians

Do This







See This



Converting Radians to Degrees or Grads

To convert from radians to degrees, proceed as follows:

1. Set GRAD/DEG switch to .
2. Enter the angle in radians. Alternatively, the angle in radians may be in the E-register as the result of a calculation.
3. Press .
4. Press .
5. Press .

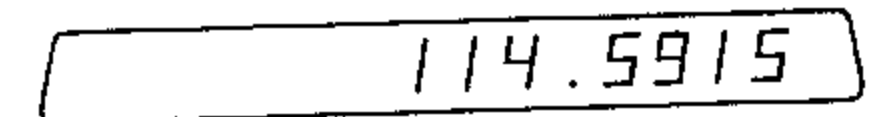
Example:

To convert 2 radians to degrees





Do This



See This

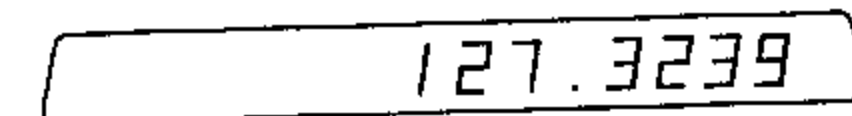


To convert from radians to grads, proceed as follows:

1. Set GRAD/DEG switch to .
2. Enter the angle in radians. Alternatively, the angle in radians may be present in the E-register as the result of a calculation.
3. Press .
4. Press .
5. Press .


Example:

To convert 2 radians to grads, proceed as follows:



Converting D/M/S Angles to Degrees with Decimal Fraction



The key sequence  converts an angle in the Entry register in D/M/S format to degrees with a decimal fraction.

Example:

To convert 31° 44' 59" to degrees with decimal fraction

Do This

See This

The answer is 31.7497°.

Sine

Pressing calculates the sine of the angle in the Entry register. The angle must be in degrees or grads. The angles may be positive or negative and may be entered in D/M/S format.

Example:

To calculate the sine of 35.5°

Do This

See This

To calculate the sine of 35° 30'

The answers in both examples are the same since the angles are equal.

Arc Sine

Pressing calculates the arc sine (in degrees or grads according to the GRAD/DEG switch setting) of the number in the E-register.

Example:

To calculate the arc sine in degrees of 0.7

Do This

See This

Attempting to calculate the arc sine of numbers greater than +1 or less than -1 gives . Press to display the number previously in the E-register.

Cosine

Pressing calculates the cosine of the angle (in degrees or grads) in the Entry register.

Example:

To calculate the cosine of 35 grads

Do This

See This

The angles may be positive or negative and may be entered in D/M/S format.

Arc Cosine

Pressing calculates the arc cosine (in degrees or grads) of the number in the Entry register.

Example:

To calculate the arc cosine in degrees of 0.7

Do This

See This

Attempting to calculate the arc cosine of numbers greater than +1 or less than -1 gives since the arc cosine function is not defined for those values. Press to display the number previously in the E-register.

Tangent


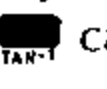
Pressing calculates the tangent of the angle (in degrees or grads) in the Entry register. To find the tangent of 35°.

Do This

See This

Angles may be positive or negative and may be entered in D/M/S format. The tangent of 90 degrees (100 grads) or odd multiples of 90 degrees (100 grads) is calculated as 1×10^{98} or -1×10^{98} .

Arc Tangent

Pressing   calculates the arc tangent (in degrees or grads according to the setting of the GRAD/DEG switch) of the number in the Entry register.

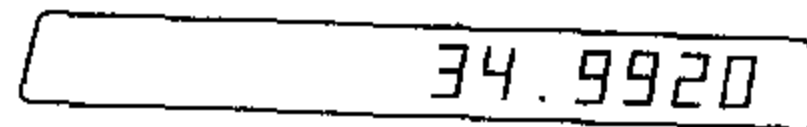
Example:

To calculate arc tangent (in degrees) of 0.7

Do This


















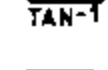








See This

























All trigonometric functions are easily accessed from the keyboard. The following table shows how to calculate them. It is assumed that the angle θ or the number n is already in the Entry register.


Trigonometric Function Table

Function	Key Sequence
Sine θ	
Cosine θ	
Tangent θ	
Cotangent θ	 
Secant θ	 
Cosecant θ	 
Arc sine n	 
Arc cosine n	 
Arc tangent n	 
Arc cotangent n	  
Arc secant n	  
Arc cosecant n	  




Some less common trigonometric functions and their key sequences follow:



Function	Equation	Key Sequence
Exsecant θ	$\text{Sec } \theta - 1$	    
Versine θ	$1 - \cos \theta$	    
Haversine θ	$1/2 (1 - \cos \theta)$	      
Coversine θ	$1 - \text{sine } \theta$	    


Polar to Rectangular Coordinates

The  key converts polar coordinates (defined by an angle, θ , and a radius, r) to rectangular coordinates, X and Y .

The procedure is as follows:





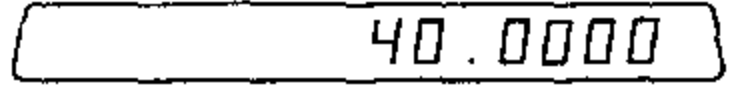

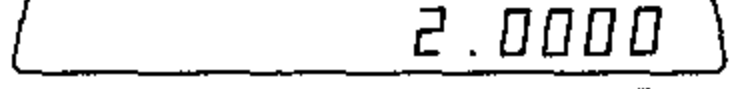

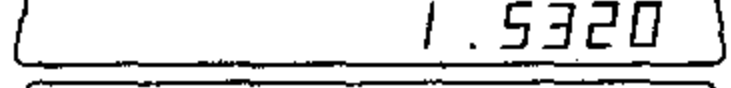



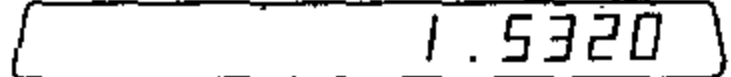
1. Enter θ in degrees or grads (according to the setting of the GRAD/DEG switch).
2. Press .
3. Enter r .
4. Press  (X will be in the E-register and display).
5. Press  again (Y will be in the E-register and display).

You may press  again to display X . If you continue to press , X and Y display alternately.

After obtaining the value of X you may perform any operation except an arithmetic operation at the same parenthesis level without affecting the Y value. For example, you may obtain X , store it in a data storage register, and then press  to obtain the value of Y .

Example:

To convert the polar coordinates ($\theta = 40^\circ$, $r = 2$) to rectangular coordinates:

Do This	See This	Comments
   		Enter θ
		Enter r
		X displays
		Y displays
		X displays again

The conversion follows the equations

$$X = r \cos \theta$$

$$Y = r \sin \theta$$

This may be seen from the diagram in Figure 12.

The angle (θ) may have a positive or negative value and be of any magnitude (the above example gives the same results with $\theta = -320^\circ$).

Rectangular to Polar Coordinates

The key sequence $\boxed{\text{TO POLAR}}$ converts rectangular coordinates X and Y, to polar coordinates θ and r.

The procedure is as follows:

1. Enter X.
2. Press $\boxed{\text{TO POLAR}}$.
3. Enter Y.
4. Press $\boxed{=}$ (θ will be in the E-register and display).
5. Press $\boxed{=}$ again (r will be in the E-register and display).

You may press $\boxed{=}$ again to display θ . If you continue to press $\boxed{=}$, r and θ display alternately. The following points should be noted:

- X and Y may be positive or negative
- The resulting value of r is always positive
- The value of θ ranges from -180 to $+180$ degrees or -200 to $+200$ grads
- The angle θ is measured counterclockwise from the X-axis for positive values of Y and clockwise from the X-axis for negative values of Y.

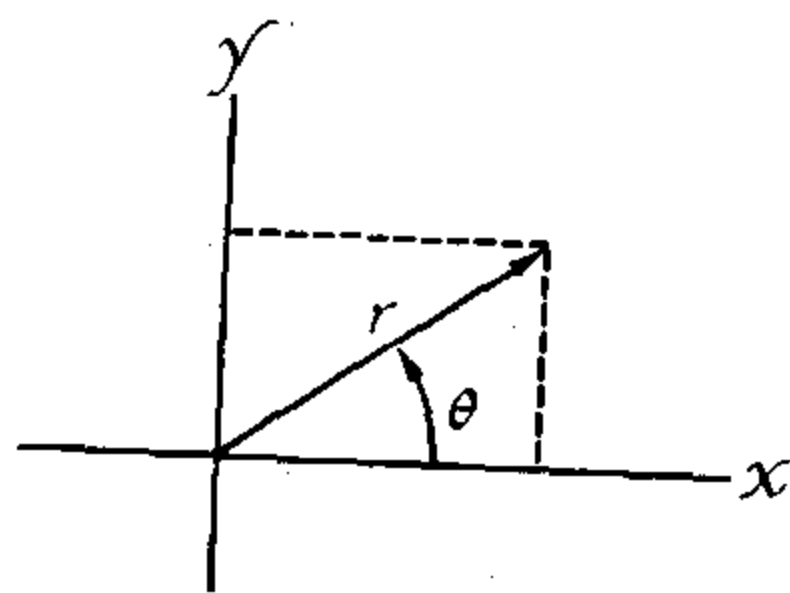


Figure 12

As may be seen from Figure 12, this conversion follows the equations:

$$\theta = \tan^{-1} Y/X$$

$$r = \sqrt{X^2 + Y^2}$$

Example:

To convert the rectangular coordinates $X = 1$, $Y = 2$ to equivalent polar coordinates.

Do This	See This	Comments
$\boxed{\text{TRAD}} \boxed{1} \boxed{\text{TO POLAR}}$	1.0000	Enter X
$\boxed{2}$	2.0000	Enter Y
$\boxed{=}$	63.4349	Value of θ in degrees
$\boxed{=}$	2.2360	Value of r

Statistical Functions

The 326 Scientist has a number of features that assist you in making statistical calculations. In addition to simple two-key sequences for obtaining mean value and standard deviation, there is a $\boxed{\Sigma}$ key that automatically counts n (number of data values entered) in data storage register 1, calculates the sum of the data values, Σx , in register 2, and calculates the sum of the squares of the data values, Σx^2 , into register 3. The symbol Σ , of course, means "sum."

Clear Registers 1, 2 and 3 $\boxed{\text{TO CLR}} \boxed{0}$

The key sequence $\boxed{\text{TO CLR}} \boxed{0}$ clears (places zeros in) data storage registers 1, 2 and 3. It doesn't affect the other data storage registers, the number in the Entry register or the display.

Using $\boxed{\Sigma}$ with Ungrouped Data

"Ungrouped data" means a set of numeric values in which no value is repeated or in which repeated values are handled individually. "Grouped data" means a set of numeric values where one or more numbers appears repeatedly and where repeated numbers may be handled as a group.

The procedure is as follows for ungrouped data:

1. Enter data item (x).
2. Press Σx^{\square} .
 - 1 is added to register 1 n count (number of items entered)
 - x is added to register 2 Σx
 - x^2 is added to register 3 Σx^2
3. Additional data items are accumulated by entering a new value of x and pressing Σx^{\square} .

Accumulated data may be retrieved at any time using:

- RCL_n **1** to display the n count
- RCL_n **2** to display Σx
- RCL_n **3** to display Σx^2

Using Σx^{\square} with Grouped Data

We define the frequency, q, as the number of times a particular data value appears in the set of data values. It is possible to add all these repeated values at once by use of the frequency, q. The procedure is as follows:

1. Enter data item (x).
2. Press Σx^{\square} .
3. Enter the frequency (q).
4. Press $=$.
 - q is added to register 1 (n count)
 - qx is added to register 2 (Σx)
 - qx^2 is added to register 3 (Σx^2)

Additional data values may be added using this method. Accumulated data may be retrieved at any time using

- RCL_n **1** to display the n count
- RCL_n **2** to display Σx
- RCL_n **3** to display Σx^2

Deleting Data From a Data Group

The sequence Δ Σx^{\square} removes either a single data value or a group of data from a statistical summation. For a single data value, the procedure is as follows:

1. Enter data item (x).
2. Press Δ Σx^{\square} .
 - 1 is subtracted from register 1
 - x is subtracted from register 2
 - x^2 is subtracted from register 3

To remove a group of data values, where the value x appears q times, do the following:

1. Enter x.
2. Press Δ Σx^{\square} .
3. Enter the frequency (q).
4. Press $=$.
 - q is subtracted from register 1
 - qx is subtracted from register 2
 - qx^2 is subtracted from register 3

Example:

To calculate n, Σx , and Σx^2 for the data values: 1, 2, 3, 3, 4, 4, 4, 5

Do This	See This	Comments
$CLEAR$ Δ 0		Clear the Entry register and data storage registers 1, 2 and 3.
1 Σx^{\square}		Enter value of 1
2 Σx^{\square}		Enter value of 2
3 Σx^{\square} 2 $=$		Value 3 appears 2 times
4 Σx^{\square} 3 $=$		Value 4 appears 3 times
5 Σx^{\square}		Enter value of 5.

To display results of the summation

Do This	See This	Comments
RCL_n 1	8.0000	n
RCL_n 2	26.0000	Σx
RCL_n 3	96.0000	Σx^2

To delete values 1, 4, 4, from the data group

Do This	See This	Comments
1 f DELETE	1.0000	Delete value of 1
4 f DELETE 2		
=	2.0000	Delete value of 4 twice

To display the new sums after deletion of this data

Do This	See This	Comments
RCL_n 1	5.0000	n
RCL_n 2	17.0000	Σx
RCL_n 3	63.0000	Σx^2

Mean Value f 3

Once a set of data values has been entered using the f key as described in the preceding section, the *mean value*, \bar{x} , may be calculated. The mean or average value is defined by the equation

$$\bar{x} = \frac{\Sigma x}{n}$$

The procedure for obtaining the mean is as follows:

1. Enter all data values using f .
2. Press f 3.

The calculated mean value will be in the Entry register and display.

Alternatively to entering data values you may simply store the values of n and Σx in registers 1 and 2, respectively, and then press f 3 to obtain the mean. If n = 0, attempting to calculate a mean value gives an error condition (E----- in display). Press CLEAR to display the number previously in the E-register.

Example:

To calculate the mean of the set of data values 1, 2, 3, 3, 4

Do This	See This	Comments
CLEAR f 0	0.0000	Clear registers 1, 2 and 3, the Entry register and display.
1 f	1.0000	
2 f	2.0000	
3 f 2 =	2.0000	
4 f	4.0000	
f 3	2.6000	Mean Value

Standard Deviation f 4

Once a set of data values has been entered using the f key (or if the values of n, Σx , Σx^2 have been directly stored into registers 1, 2 and 3, respectively), you may calculate the *standard deviation*, SD. The defining equation for the standard deviation is

$$SD = \sqrt{\frac{\Sigma x^2 - \frac{(\Sigma x)^2}{n}}{n - 1}}$$

The procedure is as follows:

1. Enter the set of data values using f (or enter n, Σx and Σx^2 in registers 1, 2 and 3).
2. Press f 4.

If n = 0 or 1, attempting to calculate a standard deviation results in an error condition (E----- in display). Press CLEAR to display the number previously in the E-register.

Example:

To calculate the standard deviation of the set of data values (1, 2, 3, 3, 4) used in the previous calculation of the mean value.

Do This	See This
f 4	1.1401

It was not necessary to enter the data values again since n, Σx and Σx^2 were already stored in registers 1, 2 and 3.

Summary of Rules for Statistics Calculations

Function	Procedure
Clear registers 1, 2 and 3	(Enter x)
Enter single data value	(Enter x)
Enter group of q identical values	(Enter x) (Enter q)
Delete single data value	(Enter x)
Delete group of q identical values	(Enter x) (Enter q)
Recall n count	
Recall Σx	
Recall Σx ²	
Mean	
Standard Deviation	

Metric/English Conversions

The 326 Scientist meets the continued need for handy conversions from English to metric units and metric to English units. To convert English units to metric units,

1. Enter the value in English units.
2. Press .
3. Press the appropriate numeral or symbol key for the desired conversion. These are listed in the following table.

The value in metric units will be placed in the Entry register and the display.

To convert metric units to English units,

1. Enter the value in metric units.
 2. Press .
 3. Press the appropriate numeral or symbol key for the desired conversion
- The value in English units will be placed in the Entry register and display.

The following table shows the basic English/Metric conversions. Other conversions using multiple key sequences are also available. Note that the last two conversions on the list (degrees to/from radians and grads to/from radians) are not actually in the category of English/Metric conversions. Since the key sequences for these two are identical, the result depends on whether the GRAD/DEG switch is set to GRAD or DEG.

Numeral or Symbol Key	English	Metric
	Degrees Fahrenheit	Degrees centigrade
	Inches, inches/second, inches/second ²	Centimeters, centimeters/second, centimeters/second ²
	Feet, feet/second, feet/second ²	Meters, meters/second, meters/second ²
	Miles, miles/hour	Kilometers, kilometers/hour
	Cubic inches	Cubic centimeters
	U.S. gallons	Liters
	Imperial gallons	Liters
	Pounds	Kilograms
	Ounces	Grams
	Pounds/cubic feet	Grams/cubic centimeter
	Pounds/square inch	Kilograms/square centimeter
	Degrees, degrees/second	Radians, radians/second
	Grads, grads/second	Radians, radians/second

Other Conversions

Additional conversions can be made using the basic conversions repetitively. For example, the following conversions for units of surface and volume measure can be performed.

From	To	Key Sequence
Square Inches	Square Centimeters	(Enter No. of in. ²)
Square Centimeters	Square Inches	(Enter No. of cm ²)
Square Feet	Square Meters	(Enter No. of ft ²)
Square Meters	Square Feet	(Enter No. of m ²)
Cubic Feet	Cubic Meters	(Enter No. of ft ³)
Cubic Meters	Cubic Feet	(Enter No. of m ³)

These are merely samples. Other multiple key sequences can be developed for your convenience from the basic English/Metric conversions.

The Number e

The key sequence places e , the base of natural logarithms, in the Entry register. The value of e is accurate to 13 digits.

Do This

See This

Absolute Value

The key sequence takes the absolute value of the number in the Entry register. The resulting positive number is left in the E-register.

Example:

To find the absolute value of -5

Do This

See This

Comments

Change -5 to 5
 No change, because number in E-register is already positive.

Fraction

The key sequence causes the fractional portion of a calculated or entered number to be retained in the Entry register and the display.

Example:

To display the fractional portion of $\sqrt{2}$

Do This

See This

Integer

The key sequence causes the integer portion of a calculated or entered number to be retained in the Entry register and the display.

Example:

To display the integer portion of $\sqrt{2}$

Do This

See This


The Number π

The key sequence places the constant π in the Entry register. The value of π is accurate to 13 digits.

Do This


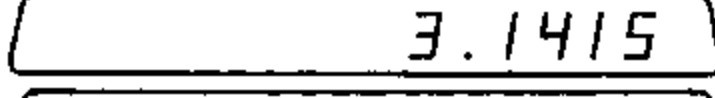



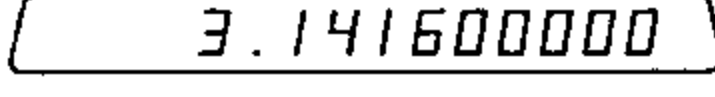
See This

Round 8




The key sequence  8 rounds the number in the Entry register to the current decimal point setting.

Example:


To round π (3.141592---) to the current decimal setting of four places.

Do This	See This	Comments
 7		Place π in E-register
 8		Round to 4 places
 9		Display subsequent decimal places

When rounding to the decimal point setting, the 326 Scientist tests the number that is one place to the right of the last digit displayed. If that digit is less than 5, all the undisplayed digits are replaced with zeros. If that digit is greater than or equal to 5, one is added to the last digit displayed and all the undisplayed digits are replaced with zeros.

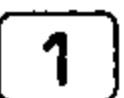

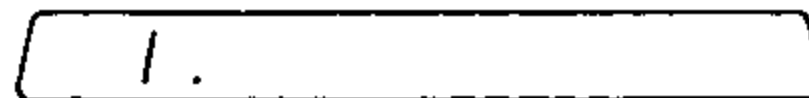
Pressing  8 has no effect on numbers in scientific notation (when   has been specified), nor does it affect angles entered in D/M/S notation.

Identifier 9


The key sequence  9 blanks out the zeros after the last significant digit of the number in the Entry register and leaves the resulting number at the extreme left of the display. Such a number is referred to as an *identifier*. This function is used in programming and will be discussed fully in the section on Programming.

Example:





To display the identifier 1

Do This	See This
 1  9	

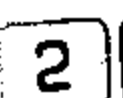






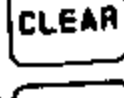



Pause EXP

The key sequence  EXP is used to momentarily display a number during the execution of a program. The Pause function will be fully explained in the Programming section.

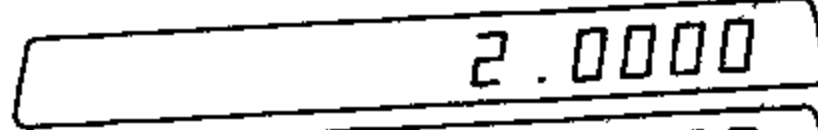
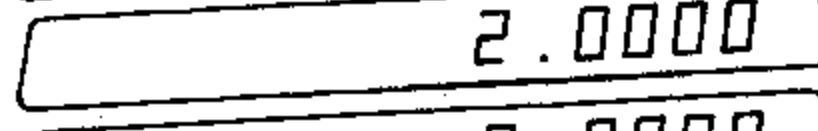
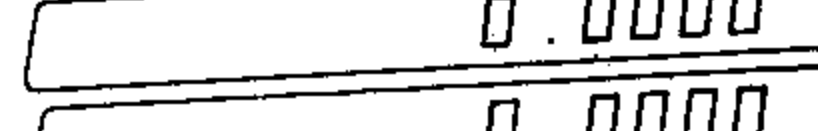
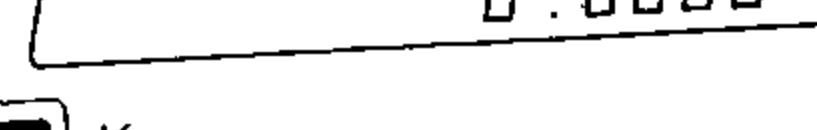
Clear All Registers CLEAR

The key sequence  CLEAR places zeros in all 12 data storage registers ( 0-9,  . and ). It also clears the Entry register and display.

Do This











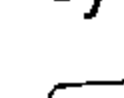


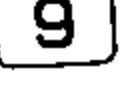






 2		 0
		 0
		
		 0

See This

Summary of Functions Using Key

Key Sequence Function

 0	Clear data storage registers 1, 2 and 3
 1	Places the number e in the Entry register
 2	Takes absolute value of number in the Entry register
 3	Calculates mean value using registers 1 and 2
 4	Calculates standard deviation using registers 1, 2 and 3
 5	Retains the fractional portion of the number in the Entry register
 6	Retains the integer portion of the number in the Entry register
 7	Places the constant π in entry
 8	Rounds the number in the Entry register to the current decimal point setting
 9	Identifier (see Programming section)
 	Used to delete data from a statistical data group
 	Fills all 12 data storage registers and the Entry register with zeros
 	Converts D/MS angle in the Entry register to degrees with decimal fraction or degrees with decimal fraction to D/M/S format.
 	Metric-to-English conversions
 	Pause (see Programming section)

Sample Applications of Functions

At this point you have been introduced to all the keys and functions that are required for the efficient use of the 326 Scientist from the keyboard. Therefore you are ready to solve a number of problems representing a variety of scientific applications. The following examples make use of most of the functions described in this section and the techniques described in the Basic Operations section.

Distance Traveled by Projectile

Suppose we wish to calculate the horizontal distance, X , traveled by a projectile fired from a cannon with a muzzle velocity of 1200 ft/sec with the cannon barrel inclined at an angle of 10° (see Figure 13)

The distance, X , is given by the formula

$$X = \frac{v^2 \sin 2\theta}{g}$$

where

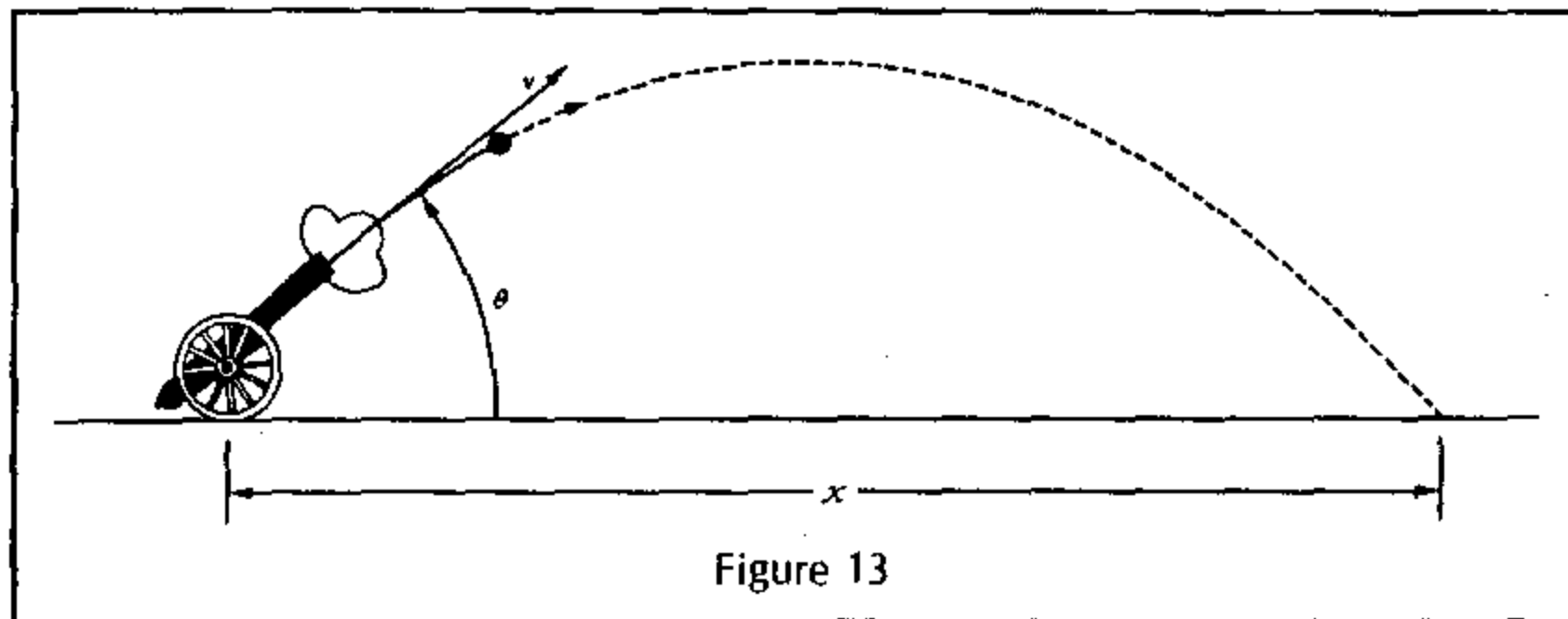
v = muzzle velocity = 1200 ft/sec

θ = angle of inclination = 10°

g = acceleration of gravity = 32.17 ft/sec²

Do This	See This	Comments
		Enter v
		Calculate v^2
		Multiply by $\sin 2\theta$
		Divide by g

The answer $X = 15,309.5743$ feet



To convert X from feet to kilometers

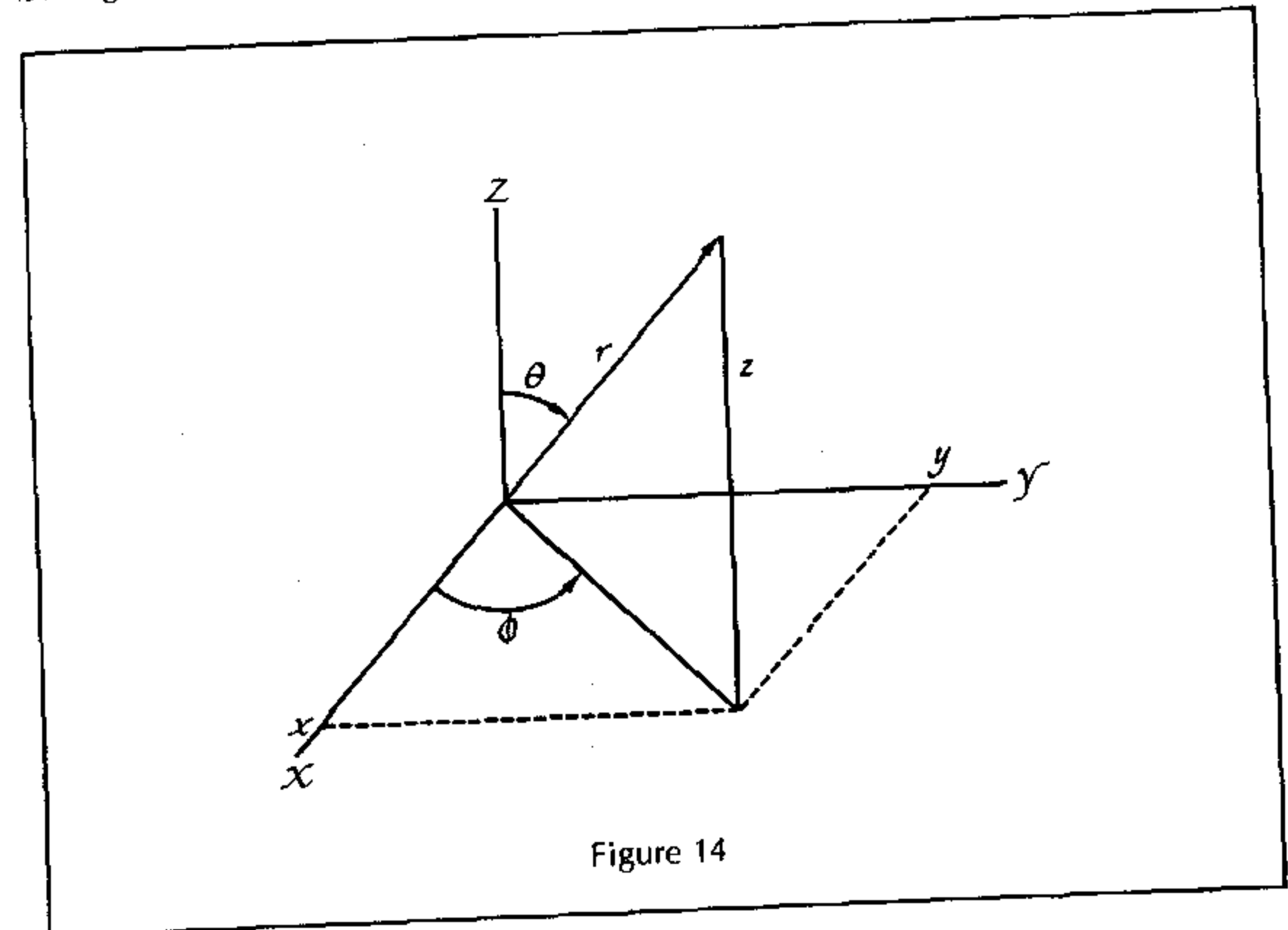
Do This	See This	Comments
		Displays X in meters
		Divide by 1000
		Round to 4 decimal places

The answer to four decimal places is

$$X = 4.6664 \text{ kilometers}$$

Converting Three-Dimensional Rectangular Coordinates to Spherical Coordinates

Suppose we wish to convert the rectangular coordinates of a point in three-dimensional space ($X = 2, Y = 3, Z = 4$) to the equivalent spherical polar coordinates (r, θ, ϕ) where r is the radius from the origin of the coordinate system, θ is the angle that r makes with the Z -axis and ϕ is the angle that r makes with the X -axis (see Figure 14).



The conversions are given by the equations

$$r = \sqrt{X^2 + Y^2 + Z^2}$$

$$\theta = \text{Tan}^{-1} \frac{\sqrt{X^2 + Y^2}}{Z}$$

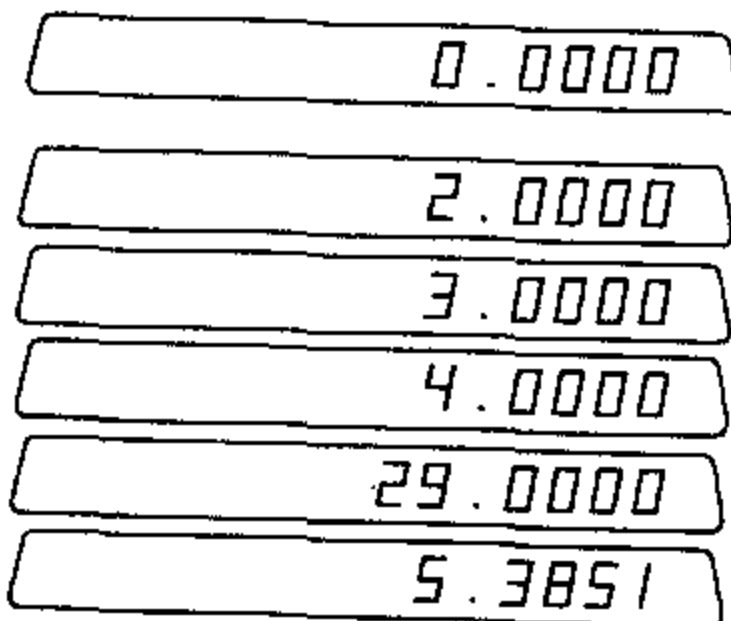
$$\phi = \text{Tan}^{-1} \frac{Y}{X}$$

To calculate r

Do This



See This

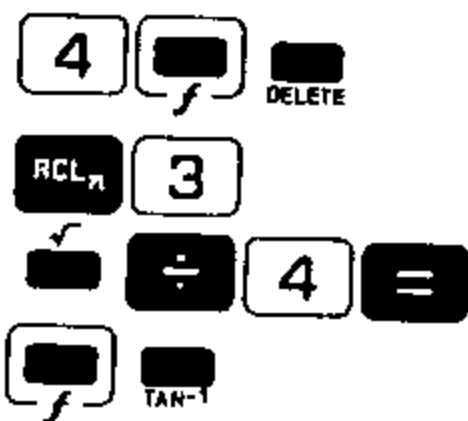


Comments

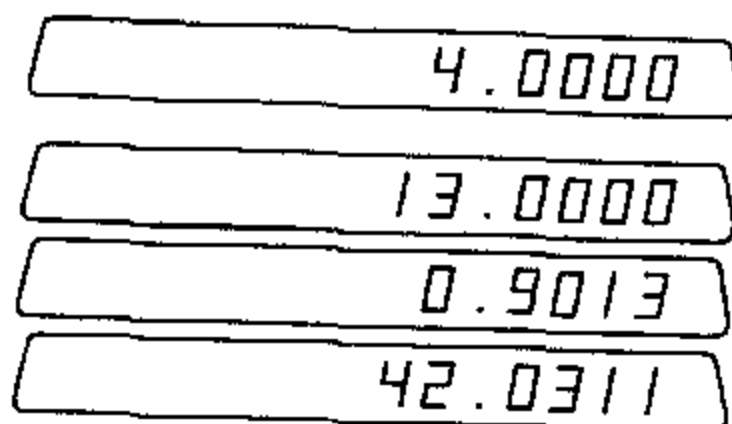
Clear all data storage registers and E-register
 Add X^2 to register 3
 Add Y^2 to register 3
 Add Z^2 to register 3
 Recall register 3
 Value of r

To calculate θ

Do This



See This



Comments

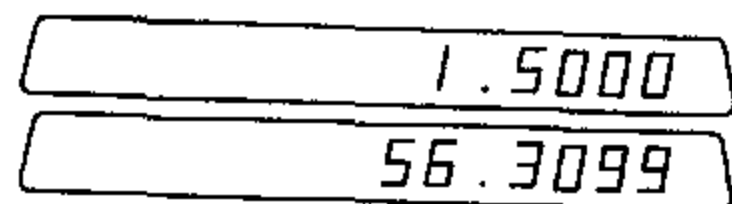
Delete value of Z^2 from number in register 3
 Recall register 3
 Value of θ

To calculate ϕ

Do This



See This



Comments

Value of ϕ

The results are

$$r = 5.3851$$

$$\theta = 42.0311^\circ$$

$$\phi = 56.3099^\circ$$

Mass of a Moving Electron

A prediction of Einstein's special theory of relativity is that the mass of a moving body is greater than its mass when at rest. The equation that describes this is

$$M = \frac{M_0}{\sqrt{1 - \frac{v^2}{c^2}}}$$

where

M = the mass of the moving body

M_0 = the mass of the body when it is at rest (rest mass)

v = the speed of the body relative to an observer

c = the speed of light (2.9979×10^8 meters/second)

Find the mass of an electron traveling at 99% of the speed of light.

$$M_0 = 9.1086 \times 10^{-31} \text{ kilogram}$$

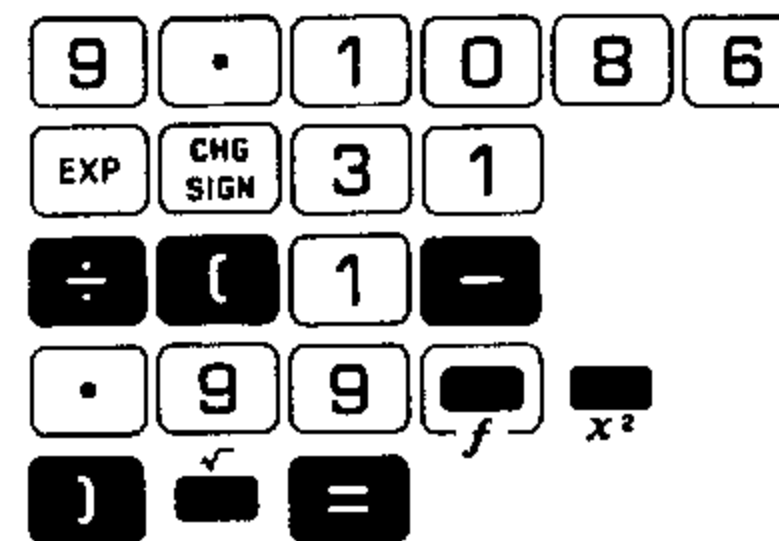
$$v = 0.99 c$$

Substituting the numeric values

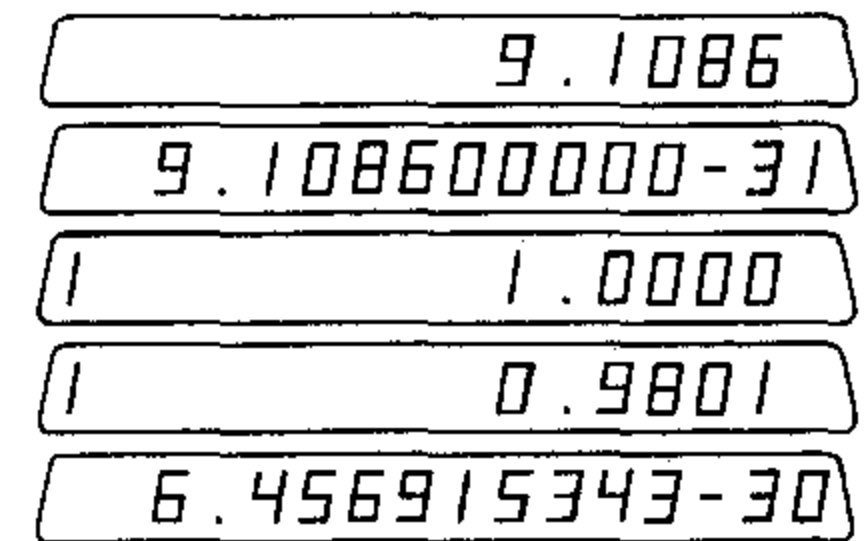
$$M = \frac{9.1086 \times 10^{-31}}{\sqrt{1 - \frac{(0.99 c)^2}{c^2}}} = \frac{9.1086 \times 10^{-31}}{\sqrt{1 - (0.99)^2}}$$

To calculate the mass, M

Do This



See This



The mass of an electron traveling at 99% of the speed of light is therefore 6.4569×10^{-30} kilogram.

Great Circle Distance

A segment of a great circle is the shortest distance between two points on the surface of the earth. The great circle distance (in kilometers), D , between points a and b is given by

$$D = K \cos^{-1} (\sin L_a \sin L_b + \cos L_a \cos L_b \cos (\lambda_a - \lambda_b))$$

where

L_a = Latitude of Point a

λ_b = Longitude of Point b

λ_a = Longitude of Point a

$K = 111.12$ kilometers per degree

L_b = Latitude of Point b

To find the distance between Los Angeles and New York (see Figure 15).

where

$L_a = 34^\circ 01'$

$L_b = 40^\circ 57'$

$\lambda_a = 118^\circ 40'$ } Los Angeles

$\lambda_b = 73^\circ 58'$ } New York

Do This

Calculator sequence for the distance calculation:

```

3 4 0 1 SIN
x 4 0
5 7 SIN
+ ( 3 4
0 1 COS
x 4 0
5 7 COS
x ( 1 1 8
4 0 -
7 3 5 8 )
COS
)
=
( 1 1 1 . 1 2
=

```

See This

Calculator display results:

```

0.5594
0.6554
0.8288
0.7552
2 118 40 00.0000
0.4449
0.8116
3,971.8435

```

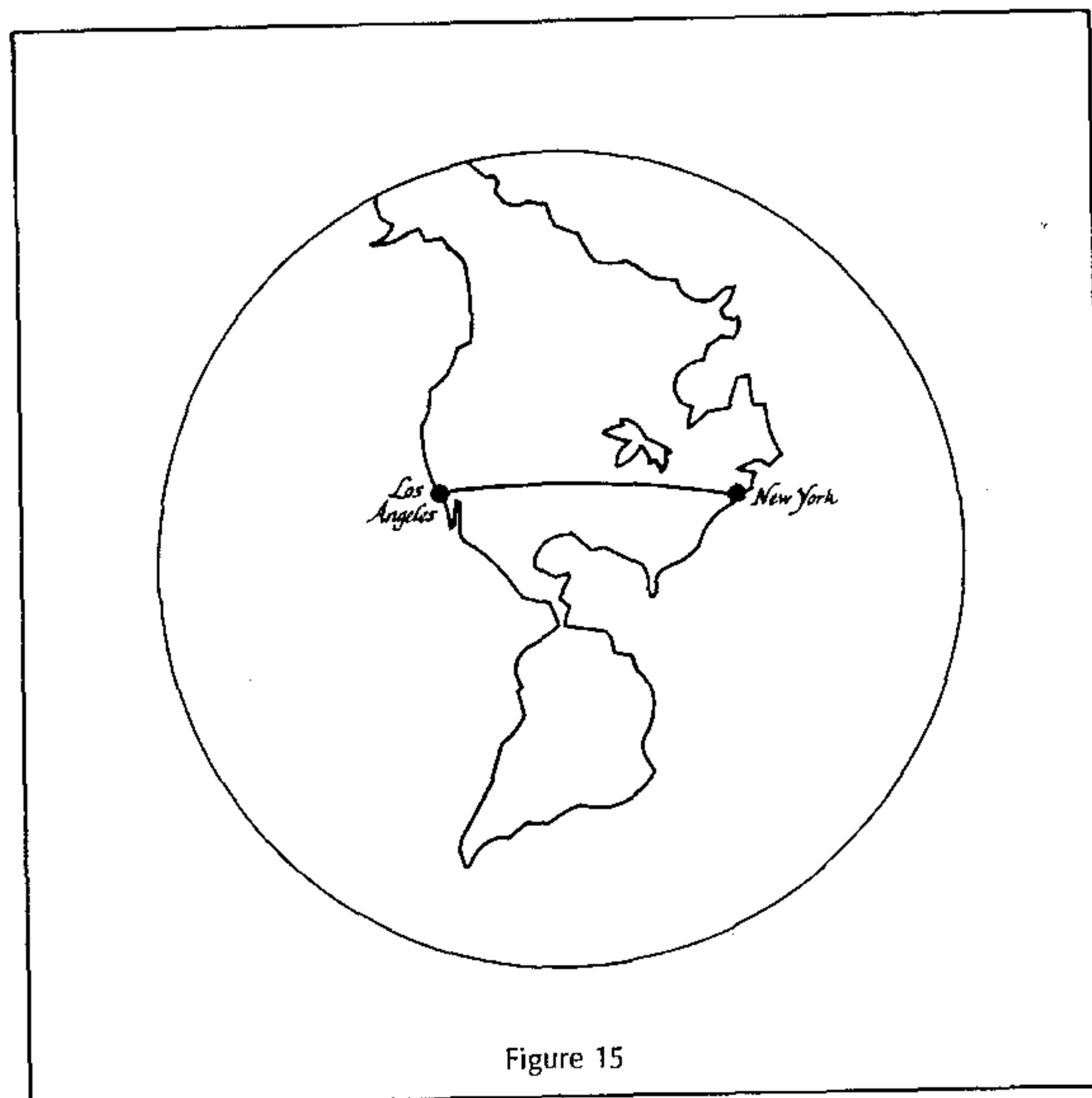


Figure 15

The great circle distance between Los Angeles and New York is 3,971.8435 kilometers. To convert this distance to miles, make the conversion

Do This

Calculator sequence for conversion:

```

3
METRIC ->

```

See This

Calculator display result:

```

2,467.9891

```

The result is 2,467.9891 miles.

Section 4 Programming

Statistical Application

Suppose we wish to find Σx and Σx^2 for a set of data values if the mean value, \bar{x} , standard deviation, SD, and number of data items, n , are known.

The equations that we use are as follows (refer to the discussions of Mean and Standard Deviation).

$$\Sigma x = n \bar{x}$$

$$\Sigma x^2 = (SD)^2 (n - 1) + \frac{(\Sigma x)^2}{n}$$

For example:

$$n = 5$$

$$\bar{x} = 3$$

$$SD = 1.5$$

To calculate Σx

Do This

5 × 3 = Σx 0

See This

15.0000

To calculate Σx^2

1 . 5 Σx^2
× (5 - 1)
+ (Σx 0 Σx^2
÷ 5) =

54.0000

Thus, the value of $\Sigma x = 15$; the value of $\Sigma x^2 = 54$.

What is a Program?

A *program* is a sequence of actions designed to bring about a desired result. While we customarily speak of programs or programming in association with computers, the concepts are valid in everyday life. For example, a person going to work in the morning might follow a "program" that consisted of the following steps:

1. Open the door of the house.
2. Look outside.
3. If the sky is cloudy, take the umbrella.
4. Step outside.
5. Close the door.
6. Walk to the garage.
7. Open the garage door.
8. Enter the car.
9. Start the engine.
10. Back the car out of the garage, down the driveway and into the street.
11. Drive 25 blocks west.
12. Arrive at work.

Normally a person thinks of such a procedure as a unit: *going to work*. However, he could consider it a sequence of steps. And if he finds it necessary to communicate this procedure to someone else, he might break the procedure into steps as we have. It might even be necessary to describe the procedure in greater detail to someone unfamiliar with automobiles. For example, "start the engine" actually consists of several discrete actions: inserting the ignition key, turning the key, etc.

From this viewpoint, whether you are a programmer or not, you have been *programming* all your life. Whenever you establish a new routine for yourself or change an existing routine, you are programming an activity. Once you have learned a new procedure you no longer need to devote attention to the steps that make up the procedure, and your mind is free for more creative work.

Similarly, when you create a program or sequence of actions for the 326 Scientist to carry out, the computer executes the program automatically, freeing you for the more creative aspects of problem solving.

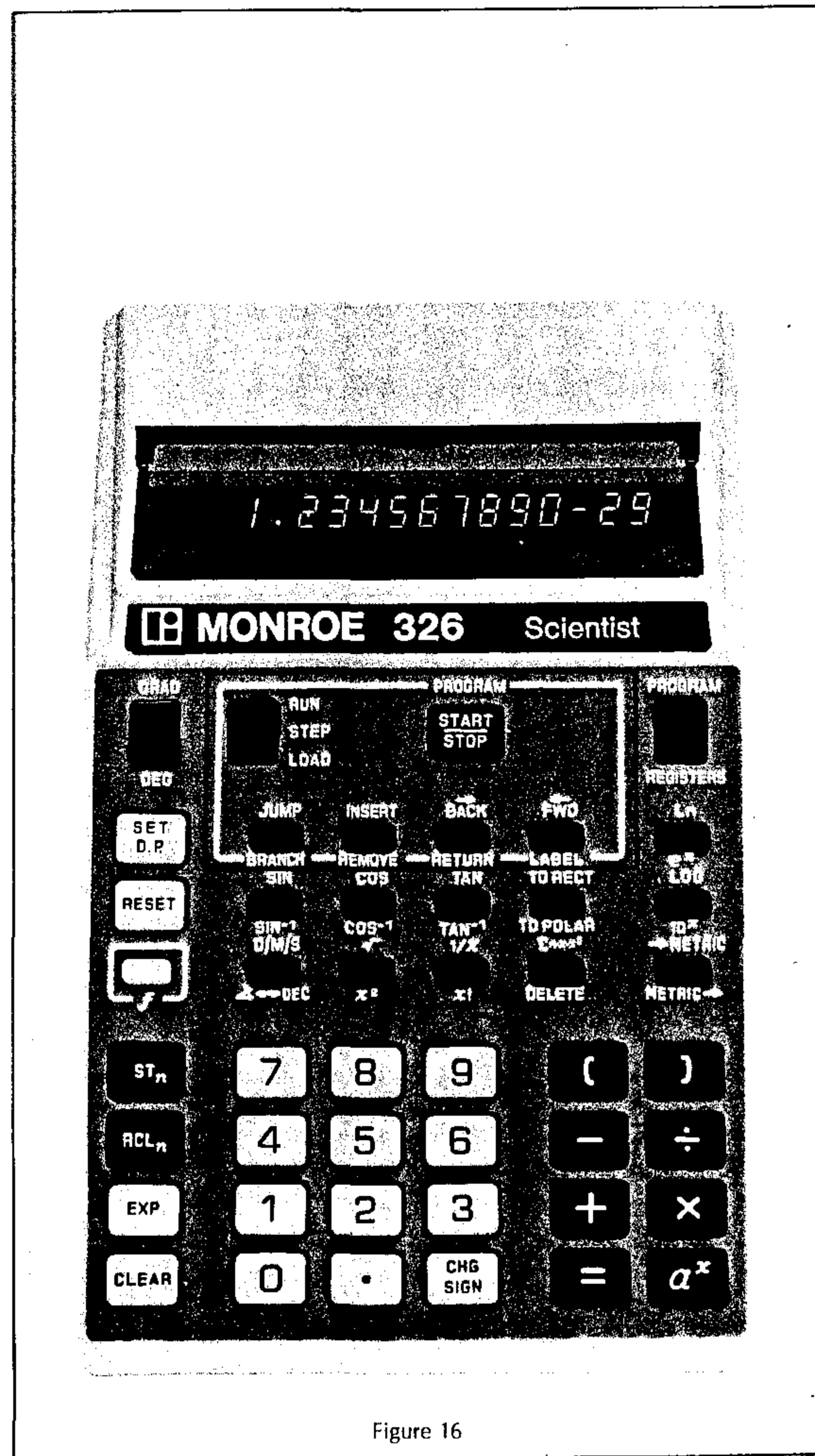


Figure 16

Programming and the 326 Scientist

If the 326 Scientist is to compute answers for you, you must tell it what to do. You can do this by giving it *instructions* from the keyboard. When you press the key sequence **2** **+** **3** **=** you are "telling" the 326 Scientist to add the numbers 2 and 3 and display the sum. These instructions can be executed immediately as you enter them or, if you are creating a program, the instructions are stored in the computer's *memory*. When instructions have been stored in the memory, the 326 carries out the program automatically at your request. In other words, you can command the 326 to carry out a sequence of instructions rather than a single instruction.

The internal memory of the 326 Scientist contains 160 *memory steps*. Each step is capable of holding an instruction, usually representing a single key. Each key and, in some cases, multiple key sequences have corresponding instruction codes of three digits. Each three-digit instruction code is stored in a memory step. For example, the three-digit code for the number 2 is 002; the code for the arithmetic operator + is 024. The instruction codes for all of the 326 Scientist's keys are given in Appendix D.

When programming, you first *load* the program into memory, that is, place the instructions that make up the program in the computer's memory. After you have loaded the program in memory, you cause it to be executed. Depending on how you design it, the program may run entirely without your intervention or it may require the entry of numeric data from the keyboard. You may also program the computer to make decisions, such as choosing one of two calculation procedures depending on the value of a calculated number. Lastly, the calculated results of the program are displayed, stored in data storage registers or recorded on magnetic tape...entirely at your discretion.

The use of the 392 Tape Cassette Drive greatly increases the scope of the computer, since both programs and data may be written on magnetic tape cassettes for later entry and use. These expanded capabilities will be discussed in the section on the 392 Tape Cassette Drive.

Program Controls

The keys and switches specifically used in programming are indicated in Figure 16. These controls are discussed at appropriate points in this Programming section.

The RUN/STEP/LOAD switch governs the mode of operation of the 326 Scientist, as described in the following paragraphs.

The RUN/STEP/LOAD Switch

During operation from the keyboard, the RUN/STEP/LOAD switch is in the RUN position. The RUN position is also used when the 326 Scientist is automatically carrying out a program stored in its memory.

The LOAD position is used to load a program into the computer's memory. A program is loaded by pressing the desired sequence of keys. Each key on the 326 has a corresponding three-digit instruction code. As the keys are pressed, the three-digit instruction codes are stored into successive memory steps. In some cases a single instruction code is used to represent a sequence of several keys. After the program has been loaded, the RUN/STEP/LOAD switch is placed on RUN and the program can be started and executed with little or no intervention required on your part.

The Step position of the RUN/STEP/LOAD switch allows you to execute a program one step at a time at your command rather than have the computer execute all the steps automatically. This is useful when you wish to closely analyze the operation of a program.

Display During Program Loading

During loading the display is divided into four segments. Each segment displays three digits. Immediately to the left of center is a three digit number preceded by a decimal point. This is the present step number. When an instruction is loaded, it will be loaded at this step. The segment to the left of the present step number is the three-digit code for the instruction at the previous step number.

If a program has already been loaded, the segment just to the right of the step number shows the three-digit code at the present step number. The segment on the far right shows the three-digit code for the instruction at the next step number.

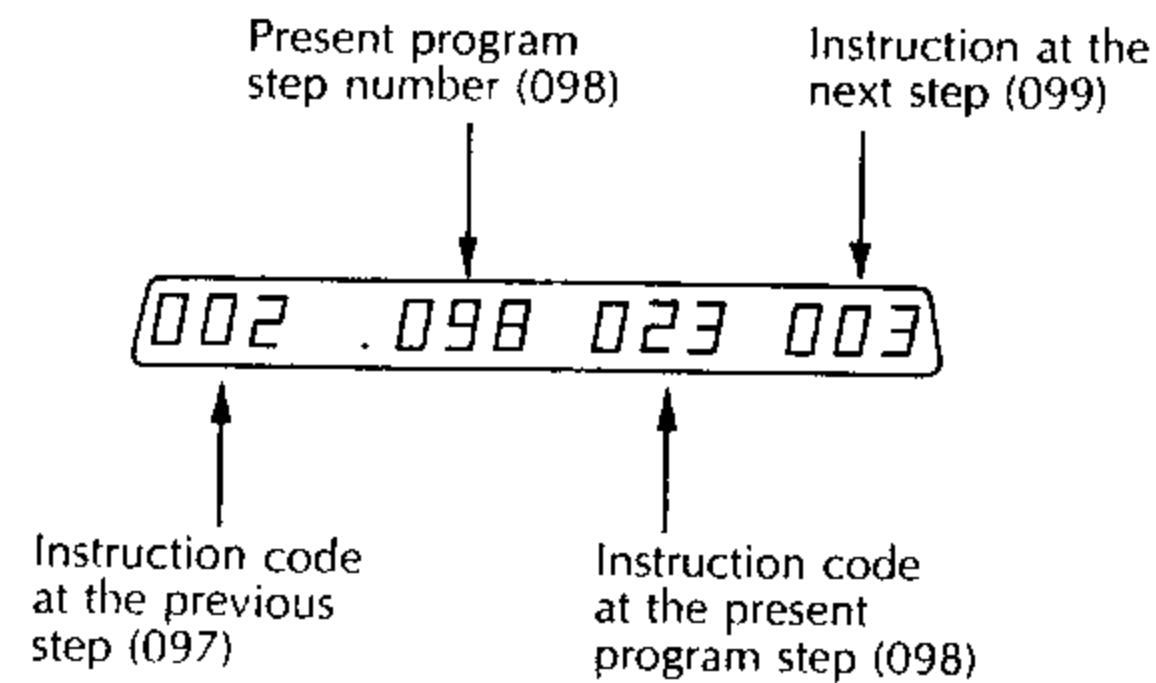
If the 326 has just turned on, the display looks like this when you switch to Load.



This means you are at program step number 001 and there are no instructions in memory.

The display below shows the display divided into four segments as just explained. The step number and codes have been arbitrarily chosen but are typical of what would appear in a program.

Display Format



The following table shows the keys to which the instruction codes in steps 097-099 correspond.

Program Step Number	Instruction Code	Key
097	002	
098	023	
099	003	

Thus, this portion of the program represents the calculation of 2×3 .

The START/STOP Key

This key, used exclusively with programming, has two functions:

1. To *start* execution of a program that is in memory when the computer is in Run mode.
2. To *stop* execution of a program, either for entry of data from the keyboard or to display a result.

Additionally, when the key is used immediately following a key as in the sequence , the program jumps to step 001 of program memory.

The Jump Key

This key is used to go directly from one point in a program to another point. is used in conjunction with a *label*, which is a key used to indicate the point in the program to which the jump is to be made.

Labels and their use with will be discussed in greater detail later in this section.

Loading a Program

The general procedure for loading a program is:

1. With the RUN/STEP/LOAD switch in position, press . This provides access to the first program memory location (step 001).
2. Put the RUN/STEP/LOAD switch in position. The display changes to the four-part Load format previously described.
3. Load the program, pressing the appropriate keys in the desired sequence.

When you press a key, its corresponding three-digit instruction code is loaded at the step number shown in the display. The step number is increased by one and you are immediately ready to load the next instruction. The code just loaded will be at left side of the display. Step 160 is the last available location in memory. After loading step 160, the next step number is 001 again.

Some multiple key sequences load into one step rather than several steps. These key sequences will be discussed later in this section.

A complete list of instruction codes is given in Appendix D. The pullout card under the computer also has a complete list.

Executing a Program



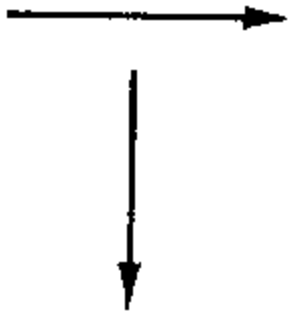
After a program has been loaded, you execute it as follows:

1. Switch the RUN/STEP/LOAD switch to .
2. Press .

The program executes automatically until it encounters a instruction. The program then stops either to display a calculated result or wait for the entry of a value from the keyboard.

Flowcharting

A useful preliminary step in writing a program is to make a *flowchart* of the procedure for solving the problem. A flowchart is a diagram representing the sequence of steps and actions that make up the program. The following symbols and conventions will be used in flowcharts of programs appearing in this book.

Symbol	Meaning
	Function. The words or symbols within the rectangle represent the action to be accomplished in this portion of the program.
	Terminal. This symbol represents the starting or stopping point in a program.
	Lines with arrows. These represent the direction of flow in the flowchart. If the arrow is omitted, a horizontal line is assumed to flow to the right and a vertical line is assumed to flow downwards.

There are many other symbols that can be used in flowcharting, but these will serve our immediate purpose.

Introductory Program

As an introduction to programming the 326 Scientist, we will write a program that multiplies 6×7 , displays the answer, and returns to the start of the program to execute it again. The flowchart (see Figure 17) shows the different parts of the program.

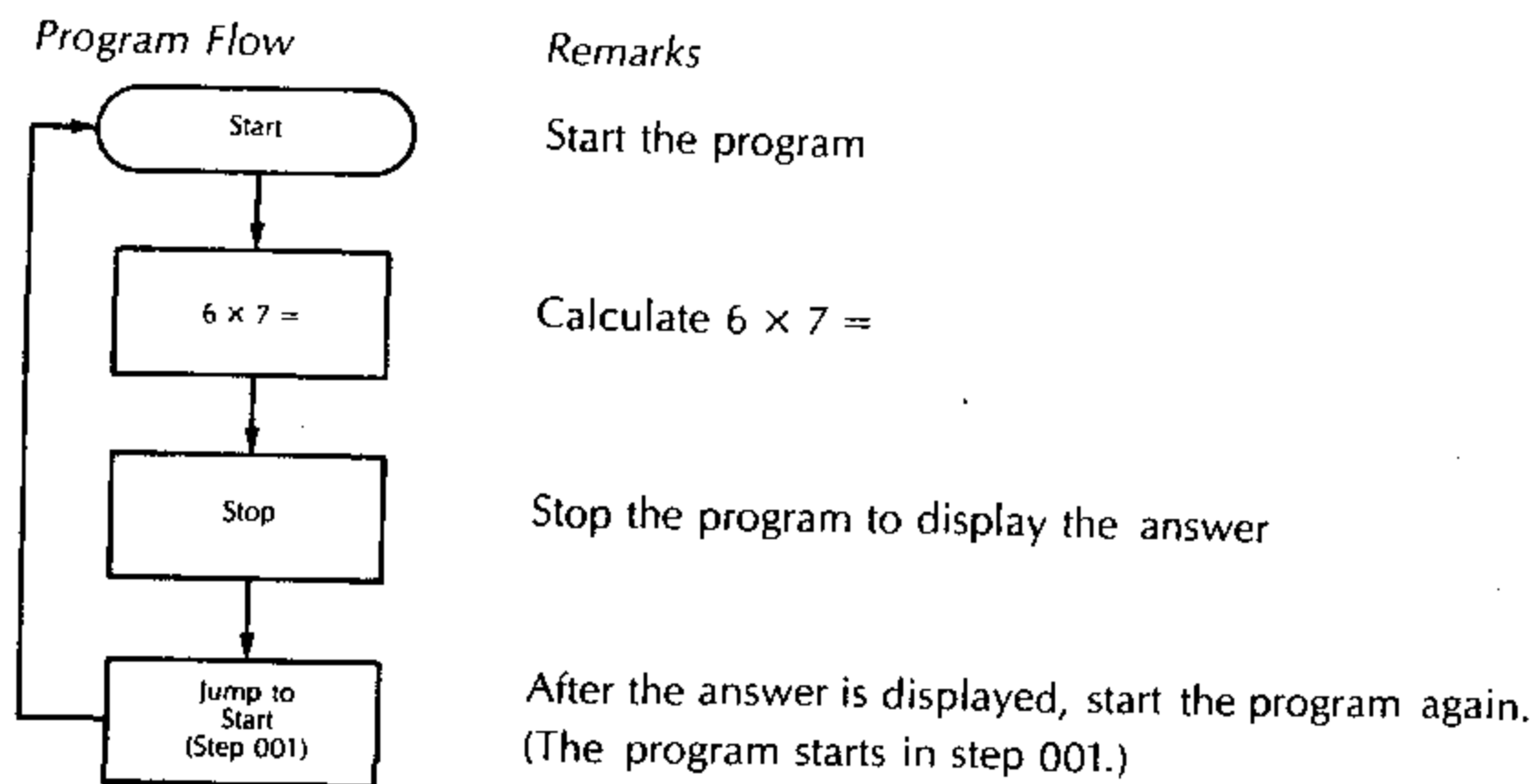




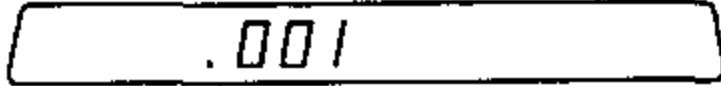


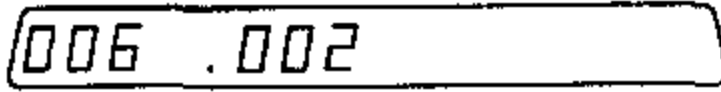

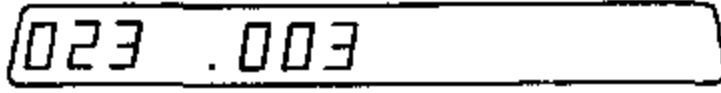
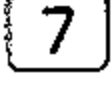
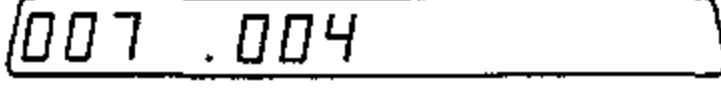

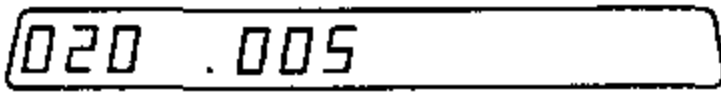



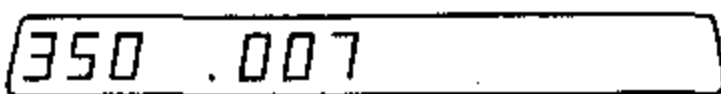

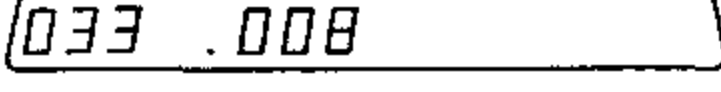

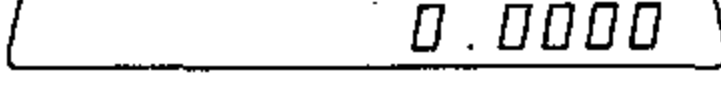


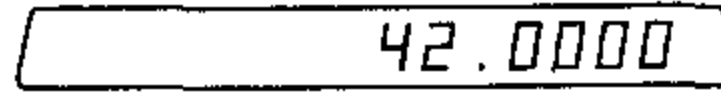

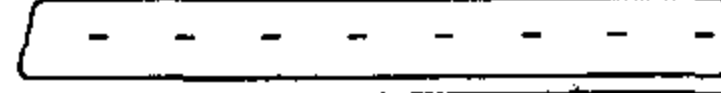
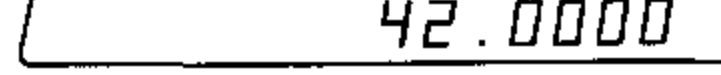


Figure 17

To Load the Program

Do This	See This	Remarks
   		Proceed to step 001 in program memory. Begin loading instructions there.
		
	<p>These positions may have instruction codes from a previous program. If no program has been loaded since turning on the 326, these positions will be blank.</p>	
		Enter 6 (Code 006) at step 001. Step 002 is ready to be loaded.
		Enter multiplication operator X (Code 023) at step 002.
		Enter 7 (Code 007) at step 003.
		Enter = (Code 020) at step 004.
		The Stop instruction (Code 033) causes the program to stop to display the answer.
		These two instructions cause the program to return to step 001.
		
		Prepare to execute (run) the program. The display shows the number that was in the display before switching to Load.

You execute the program, as follows:

Do This	See This	Remarks
		The display blinks during program execution.
		The program stops after the calculation to display the answer.
		The program executes the Jump and Start instructions and then calculates 6×7 again.
		

Sample Program: Volume of a Hollow Sphere

Suppose we want to write a program that calculates the material volume of a hollow sphere (see Figure 18). The Volume V of the solid part is given by the equation

$$V = \frac{4}{3} \pi (r_0^3 - r_1^3)$$

where

r_0 = The outside radius of the sphere

r_1 = The inside radius of the sphere

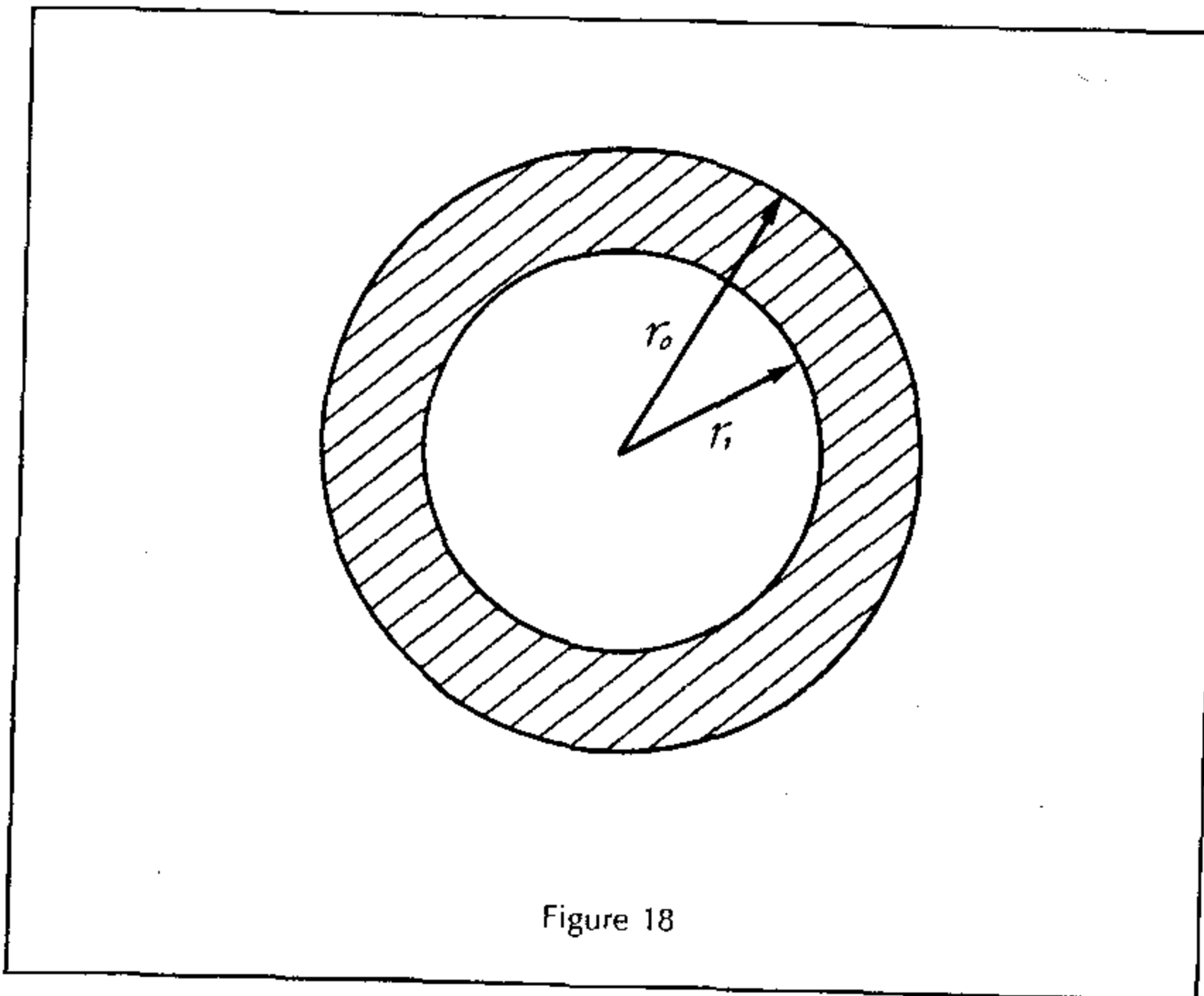


Figure 18

The general procedure to solve this problem is: enter values for r_0 and r_1 , calculate V , display V , then return to the beginning of the program to execute it with new values of r_0 and r_1 . The first time the program is run, r_0 is assumed to be in the Entry register. The flowchart for this program is shown in Figure 19.

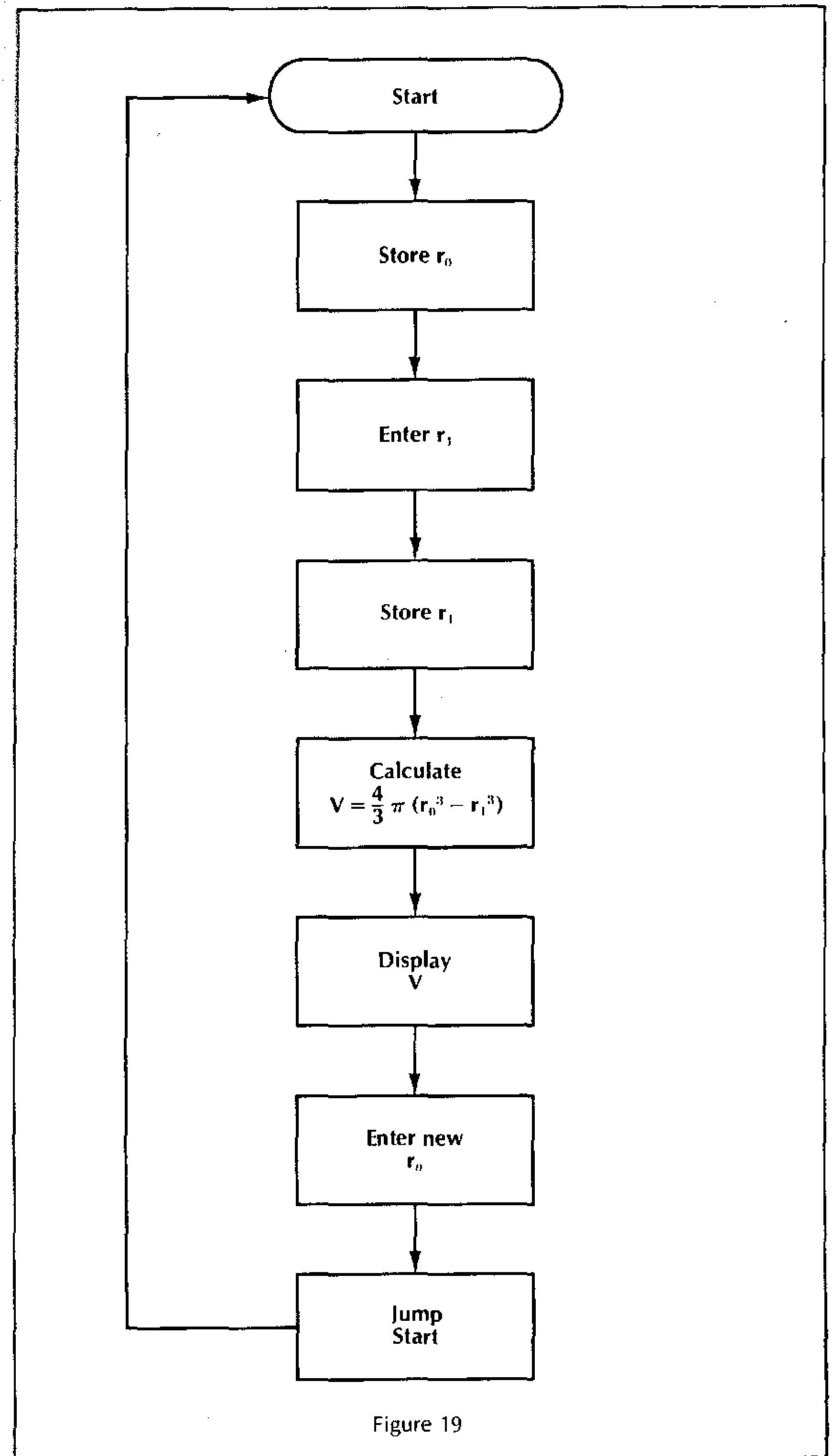


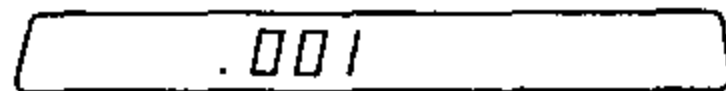
Figure 19

To load the program

Do This

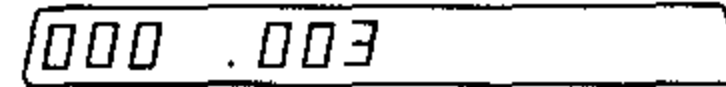


See This

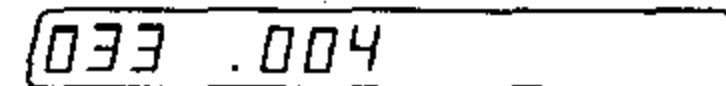


Old instruction codes appear in these positions if a program is already in memory. They are replaced as the new program is loaded.

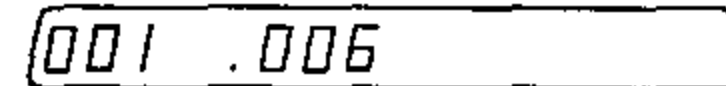
Comments



r_0 is assumed to be in the Entry register when the program starts. It is stored in register 0



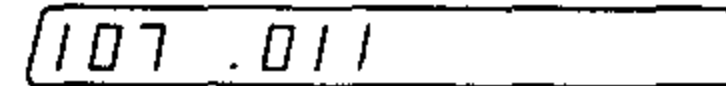
The program stops to allow entry of r_1



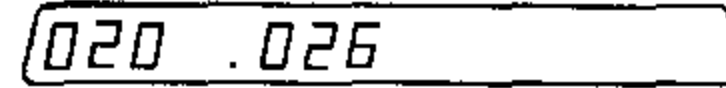
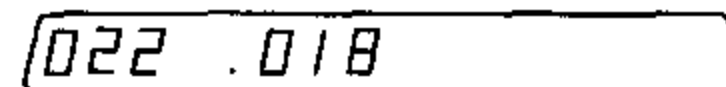
Store r_1 in register 1



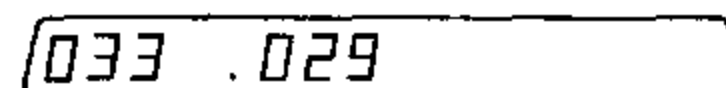
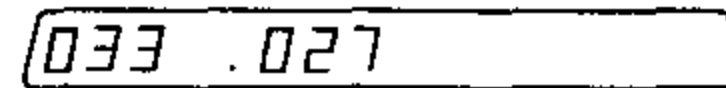
Calculate $4/3 \pi$



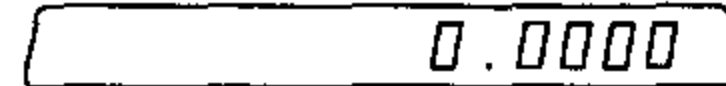
$V = \frac{4}{3} \pi (r_0^3 - r_1^3)$



The program stops to display V and permit entry of a new r_0



Return to Start (Step 001)



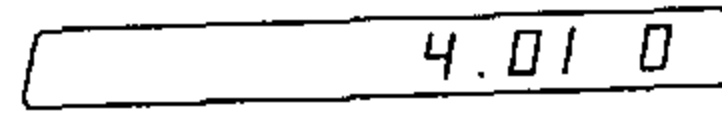
Set switch to RUN in preparation for running program

To run the program with the values $r_0 = 4.01$, $r_1 = 3.8$

Do This



See This

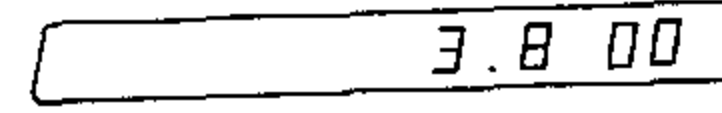


Comments

Enter 4.01



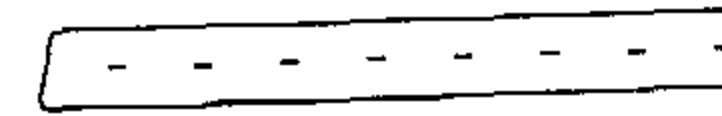
Start execution of program



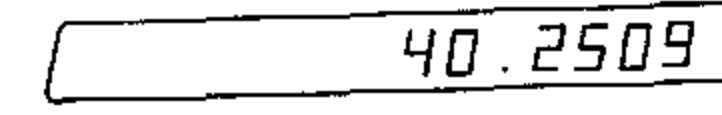
Enter 3.8



Start execution of program again



The display shows blinking dashes as program calculates V



V is displayed

You may now enter a new value of r_0 and press **START STOP** to cause the program to be executed again. Try the program using the following values of r_0 and r_1 (the resulting values of V are presented at the right).










r_0	r_1	V (Calculated)
4.02	3.80	42.2766
4.00	3.85	29.0424
5.00	4.00	255.5162

Labels




































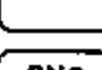
A label is a symbol used to name a program or program segment. The label occupies the step in program memory immediately before the beginning of the program or program segment to which it refers. The symbols available for use as labels are the numeral keys, **0** - **9**, the decimal point key, **.**, and the change sign key, **CHG SIGN**.

When a program or program segment has been labeled, you may jump to it from some other point in program memory using the Jump instruction.

To label a program or program segment, do the following:




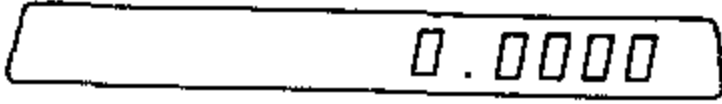


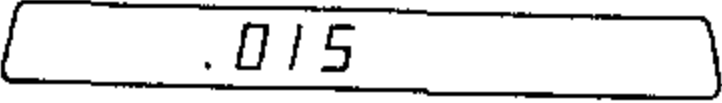

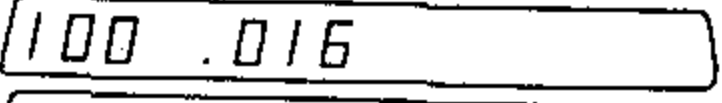

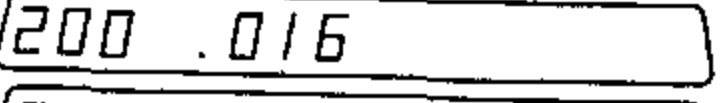
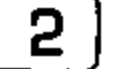
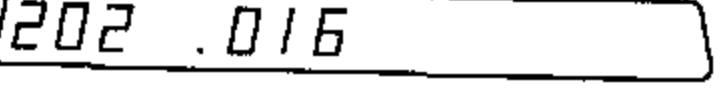




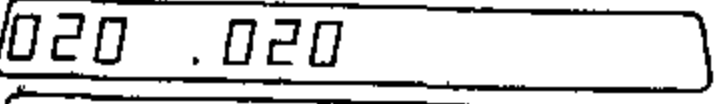

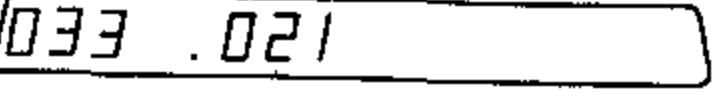

1. Set the RUN/STEP/LOAD switch to .
2. Access the desired memory step using  or .
3. Press  .
4. Press one of the following keys: ( - ,  or .

Each label has its own three-digit code. The label codes are as follows:







Key Sequence	Instruction Code	Key Sequence	Instruction Code
  	200	  	206
  	201	  	207
  	202	  	210
  	203	  	211
  	204	  	212
  	205	  	213

Example:

To load a program that calculates 6×7 starting at memory step 15 and assign it a label 2.



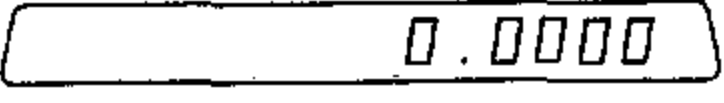

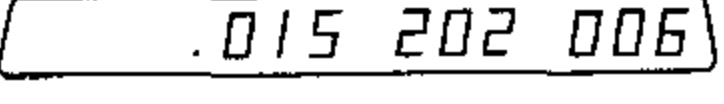


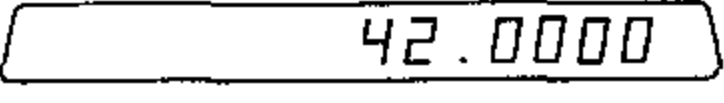



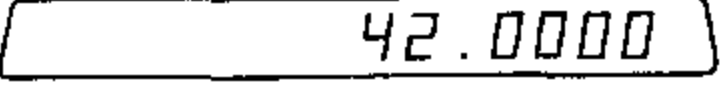
Do This	See This	Comments
  		Jump to step 001
		Switch to Load
(Press  14 times)		Go forward in memory to step 15
		This sequence of three keys loads one instruction code into step 015. The step counter does not change until all three keys have been pressed.
		
		
   		Begin loading the program in step 016.
		Calculate $6 \times 7 =$
		Stop to display answer.
		Place in Run mode in preparation for execution.

To jump to a labeled program or program segment, do the following:

1. Set the RUN/STEP/LOAD switch to .
2. Press .
3. Press the key for the desired label ( - ,  or .

Example:

To jump to the program labeled 2, which was previously loaded at step 15

Do This	See This	Comments
 		Jump to label 2
		Switch to Load mode to verify that the computer is indeed at step 015.
 		Switch to Run mode and execute the program that calculates 6×7 .
  		Jump to step 015 and execute the program again.

These important points should be noted with respect to labels:

- The use of labels allows you to jump to a program or program segment regardless of its exact numeric step number. A program may be relocated in memory through the use of INSERT and REMOVE instructions (these instructions are discussed in the section on Modifying a Program). However, you may jump directly to the program by using a JUMP and the appropriate label.
- If you wish to jump to a label, you must define it first. If you attempt to jump to a label that has not been defined, the 326 will automatically jump to step 001 in memory.
- If a label appears more than once in memory, only the label at the lowest step number is recognized.

Sample Program Using Labels

This example demonstrates the use of labels when loading more than one program. Three different programs are written to calculate the hyperbolic functions:

Program A $\sinh(x) = \frac{e^x - e^{-x}}{2}$

Program B $\cosh(x) = \frac{e^x + e^{-x}}{2}$

Program C $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

These programs may be loaded anywhere in program memory. For demonstration we will load them at starting address 027.

PROGRAM FLOW

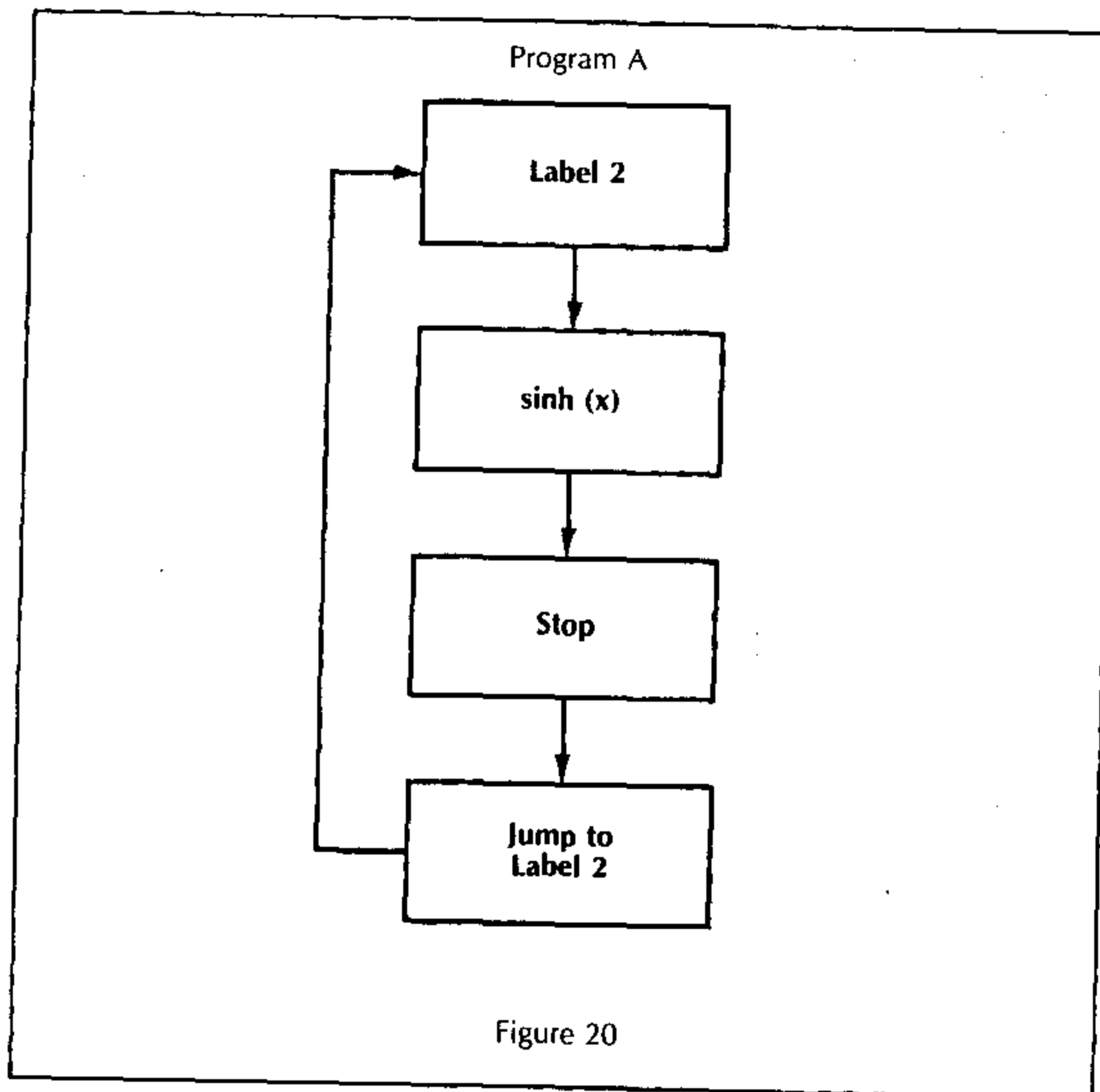


Figure 20

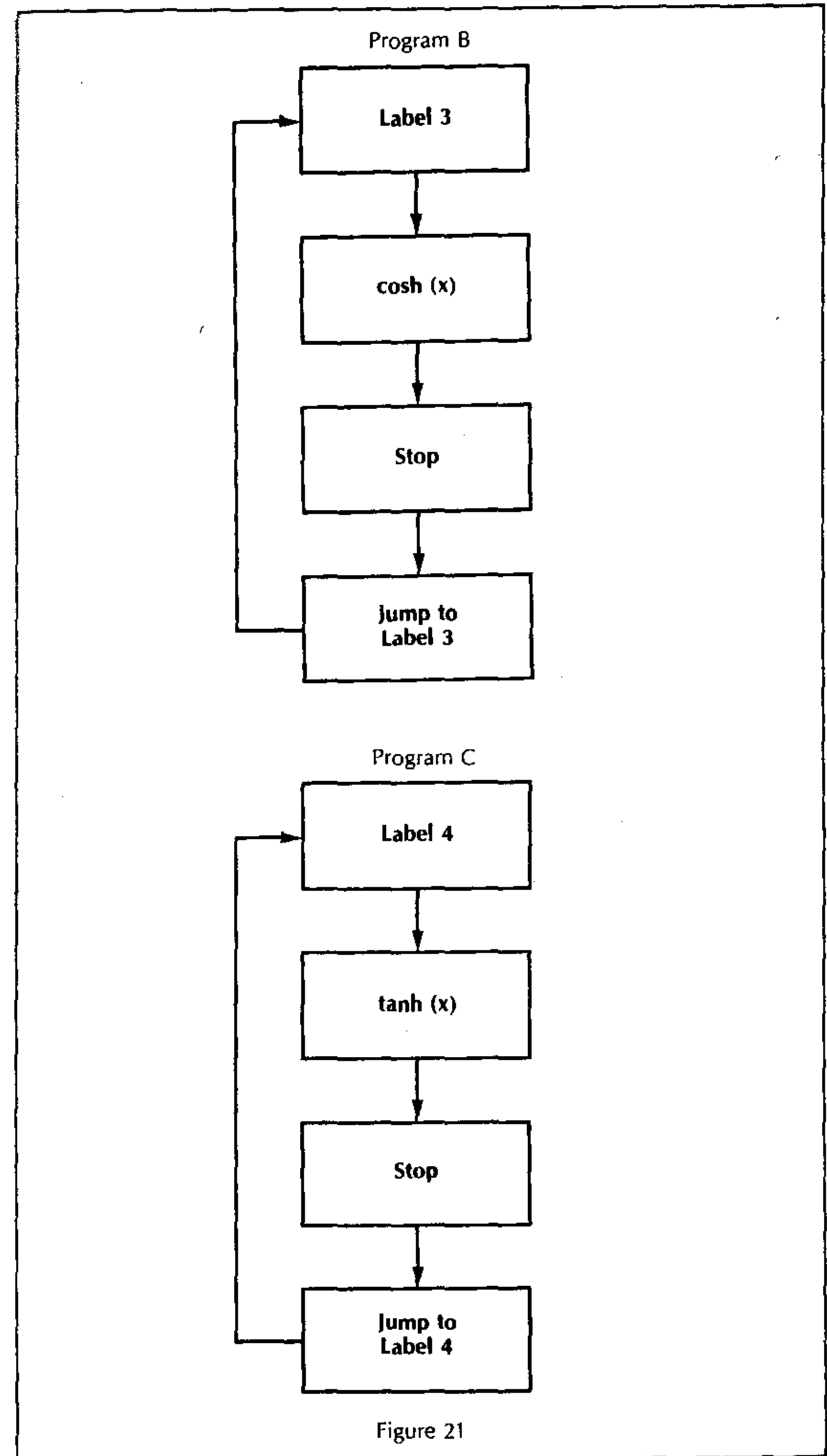


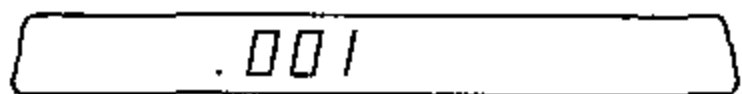
Figure 21

To load the hyperbolic function programs

Do This

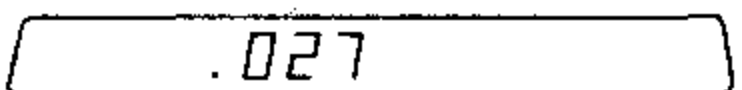
See This

Comments

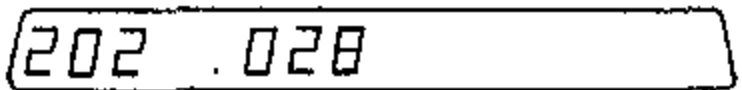


These locations may be blank or have instruction codes from a previous program.

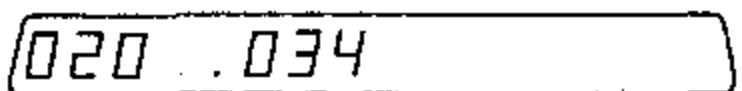
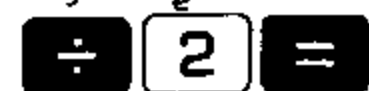
(Press 26 times)



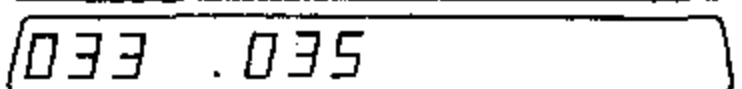
Go forward in memory to step 027



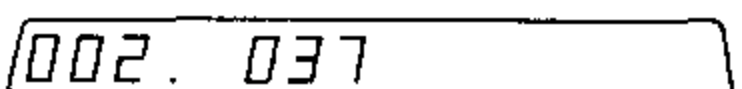
Define label 2



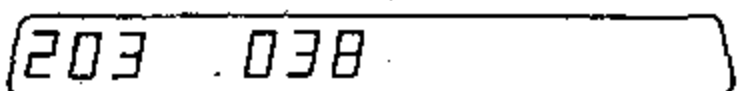
Compute Sinh(x)



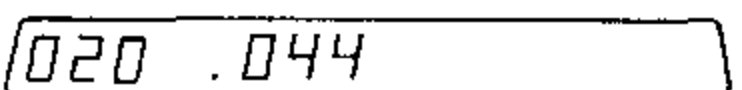
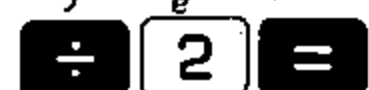
Display Sinh(x) and enter a new x value if another Sinh(x) is to be calculated



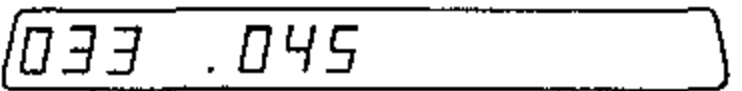
Return to label 2 for a new Sinh(x) calculation



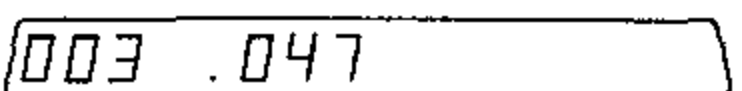
Define label 3



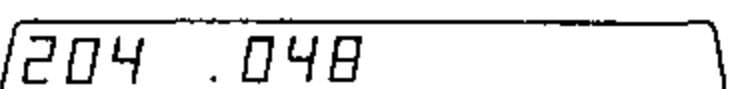
Calculate Cosh(x)



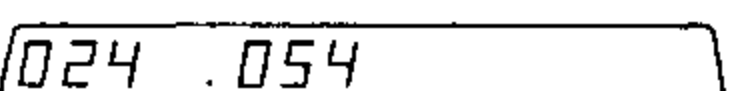
Display Cosh(x), and enter a new x value if another Sinh(x) is to be calculated.



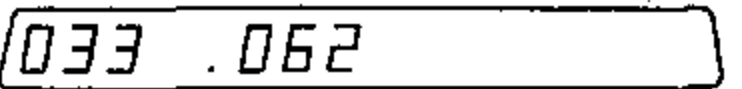
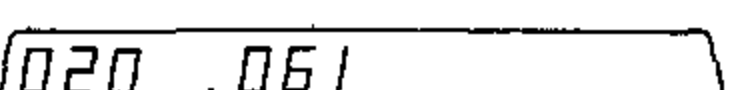
Return to label 2 for a new Sinh(x) calculation.



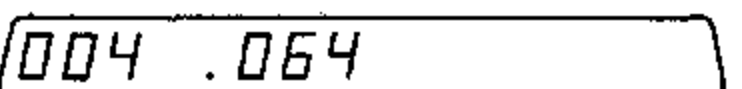
Define label 4



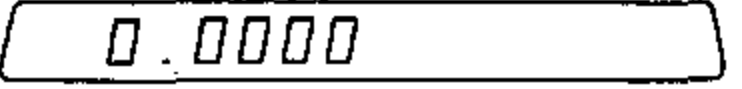
Calculate Tanh(x)



Display Tanh(x), and enter a new x value if another Tanh(x) is to be calculated.



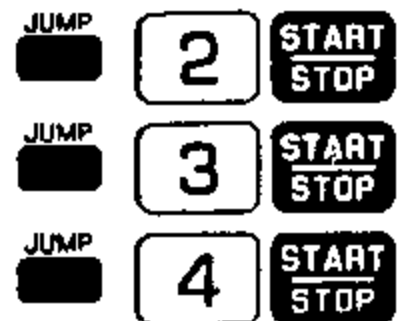
Return to label 4 for a new Tanh(x) calculation.



The programs are loaded and ready to run. The display shows the same value that it held prior to program loading.

To run the hyperbolic function programs using these labels, the procedure is as follows:

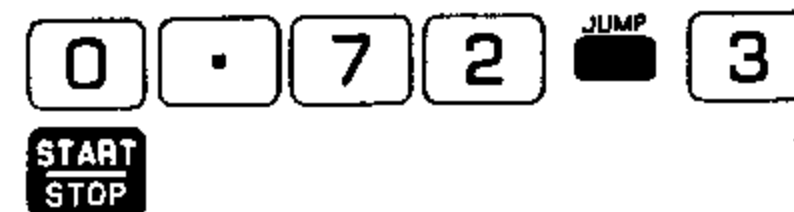
1. Enter x
2. To compute Sinh(x), press
3. To compute Cosh(x), press
4. To compute Tanh(x), press



Examples:

To calculate Cosh (0.72)

Do This

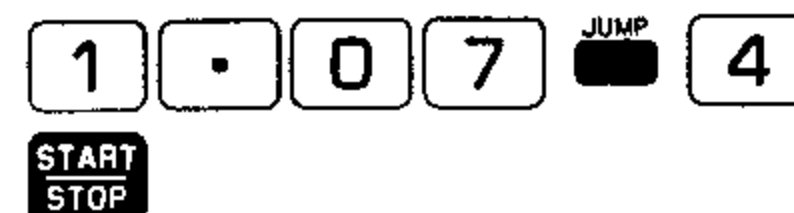


See This



To calculate Tanh (1.07)

Do This

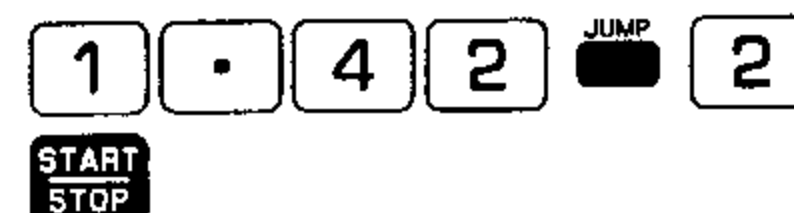


See This

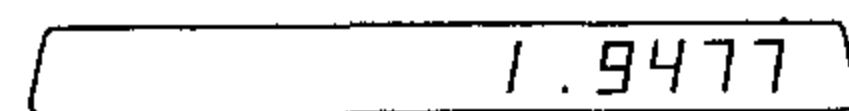


To calculate Sinh (1.42)

Do This










See This





Modifying a Program

After a program is written you may want to modify it by adding, changing or removing instructions. The 326 Scientist is designed to let you do this quickly and easily.



The programming controls ,  and  allow access to any point in the program.  and  allow the insertion and removal of instructions at the location accessed.

Changing an instruction does not require the use of  or . You simply load the new instruction at the desired memory step. The instruction that you load replaces a previously loaded instruction.


The Forward Key

When the 326 Scientist is in Load mode, pressing this key allows you to advance one memory location to the next higher step number. You can move through program memory one step at a time using this key.  does not alter the instructions in the computer's memory and it does not cause program execution. When you are at step 160, pressing  brings you to step 001.



The Backspace Key

When the 326 Scientist is in Load mode, pressing this key allows you to move back one memory location to the next lower step number. You can move backwards through program memory one step at a time using this key.  does not alter the instructions in the computer's memory and it does not cause program execution. When you are at step 001, pressing  brings you to step 160.

The Insert Key







 allows you to add an instruction within a program. The step numbers of instructions after the insertion are increased by one and no instructions are lost.

The Remove Key



  allows you to remove an instruction from a program. The instruction is removed and the step numbers of all program steps after the removed instruction are decreased by one.

Accessing a Step in Memory

To access a particular step in program memory, do the following:

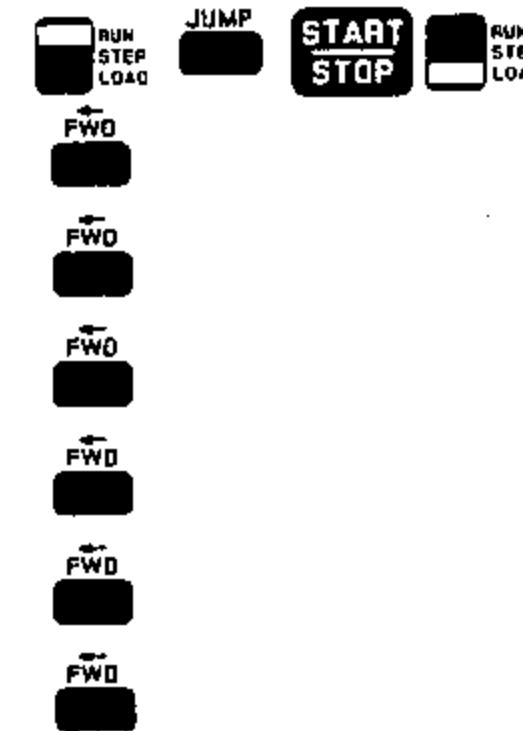
1. Jump to the label nearest the location to be accessed. The sequence is  followed by the appropriate label.
2. Set the RUN/STEP/LOAD switch to . The address of the label, the label code (200 to 213) and the codes immediately before and after the address of the label are displayed.
3. If the memory step to be accessed has a higher step number than the current one, press  to get to the next higher step in memory. To go forward several steps, press  repeatedly.
4. When the memory step to be accessed has a lower step number than the current step, press  to go to the next lower step in memory. Press  repeatedly to go back more than one step.

Example:

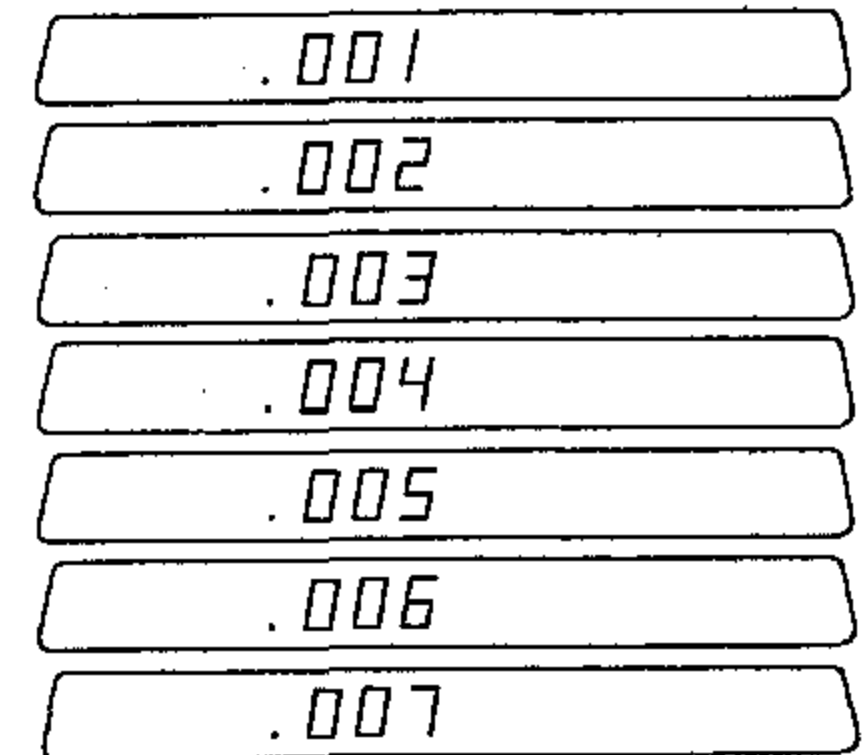
Using  and  to access specific memory locations.

To access Step 007

Do This



See This

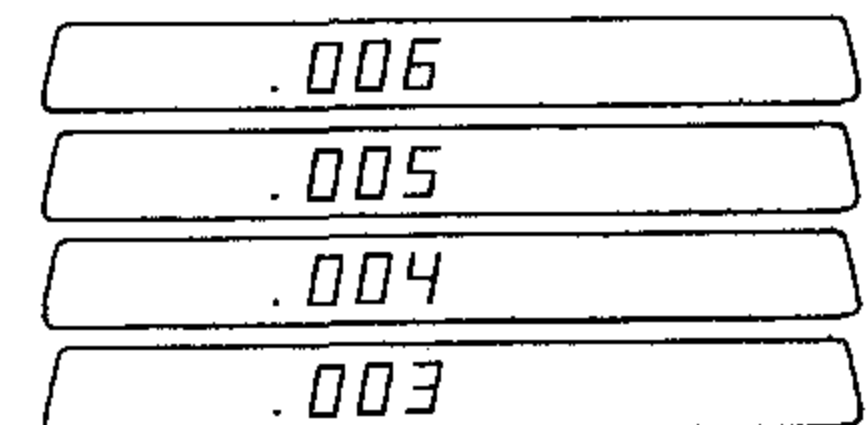


To go back to Step 003

Do This












See This



Inserting an Instruction

The procedure for inserting an instruction in a program is as follows:

1. Access the step number where the instruction is to be inserted (using  to the nearest label,  or  as required).
2. Press . All instructions starting from the current memory step are moved one step higher, leaving a blank at the current memory step. If program memory is full, the last instruction in the program (before  was used) is lost. To insert more than one instruction, press  once for each instruction to be inserted.
3. Press the key of the instruction to be inserted. Its code is put into the blank step created by the . If you have created several blank steps by using  several times, press the appropriate number of keys to fill the blanks with the new instructions. If you don't key in an instruction code after creating a blank with , the program operates as if the blank step were not there.










Changing an Instruction

The procedure for changing an instruction is as follows:

1. Access the memory step or the instruction you wish to change. The instruction code to be changed appears immediately to the right of the step number.
2. Enter the new instruction by pressing the appropriate key. The new instruction replaces the old instruction and appears to the left of the program memory location number. The step number is increased by one.











Removing An Instruction

The procedure for removing an instruction from a program is as follows:




1. Access the memory step of the instruction to be removed, as described above (using  to the nearest label,  or  as required).
2. Press  . The instruction at the current memory step is deleted. All following instructions in the program are moved one step lower in memory. If the program memory was full, the last step becomes a blank.
3. To remove a string of instructions, access the location of the first instruction of the string and press  . Then continue to press  , once for each subsequent instruction to be removed.

Sample Program Modifications

We will load a program in memory that calculates the area of a circle, $A = \pi r^2$, where r is the radius. We will then modify the program three times to show how instructions are inserted, removed, and changed. The program has the following sequence of instructions prior to modification.

Step Number	Code	Key	Comments
001	033		Enter r
002	162	 	Calculate r^2
003	025		
004	107	 	Multiply by π
005	020		
006	033		Display A
007	350		
008	033		Jump to step 001

To load the program, do the following:

1. With the computer in Run mode, press  .
2. Set the RUN/STEP/LOAD switch to .
3. Enter the key sequence as listed.

Inserting Instructions

To insert the Identifier 1 just prior to the first Stop instruction:

Do This



INSERT

INSERT

1



9



See This

.001 033 162

.001 033

.001

001 .002 033

100 .003 033 162

111 .003 033 162

0.0000

Comments

The display shows memory location 001 and the first two codes of the program.

This creates two blank locations at steps 001 and 002. The original program steps are moved forward two steps.

The first blank is replaced by the numeric value 1 which will be used by the next step as the value for the Identifier.

The code (111) for the identifier function is loaded into the blank location at step 002.

Prepare to run the program.

To run the modified program with a radius, $r = 2$:

Do This



START STOP

2

START STOP

See This

1.

2.0000

12.5663

Comments

Jump to step 001.

Start program execution. The identifier displays indicating that the value of r must be entered.

Enter value of r .

Value of A is displayed.

Removing Instructions

Suppose you wish to remove the identifier and the second Stop (where the value of A is displayed).

Do This



(Press 5 times)



See This

.001 001 111

.001 111 033

.001 033 162

020 .006 033 350

020 .006 350 033

Comments

This removes the Identifier instructions at steps 001 and 002.

The second Stop instruction (code 033) is now in the current step position.

Removes the Stop instruction at step 006. Notice that code 350 is now in step 006.

Changing Instructions

Suppose we wish to change the at the end of the program to 3. (This assumes that we have another program in memory labeled 3.) Since we are already at step 006 from the last example, we proceed as follows:

Do This



3

See This

350 .007 033

003 .008

Comments

Advance to step 007 without changing the in step 006.

Step 007 now contains the label 3. Step 008 is blank.

Subroutines and Branching




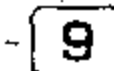
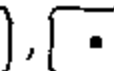
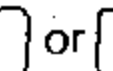
Programs often require the use of the same sequence of instructions in several places. Rather than loading the set of instructions each time you need it, you may define a *subroutine* to which you branch at each point of the program where the sequence is needed.

A *subroutine* is a labeled program or program segment which may be considered a part of another program. The last step of a subroutine is a Return instruction and, after execution of a subroutine is complete, you return to the step immediately following the point from which you branched. This is shown in Figure 22, where the main program branches to the subroutine on two occasions.

Branch

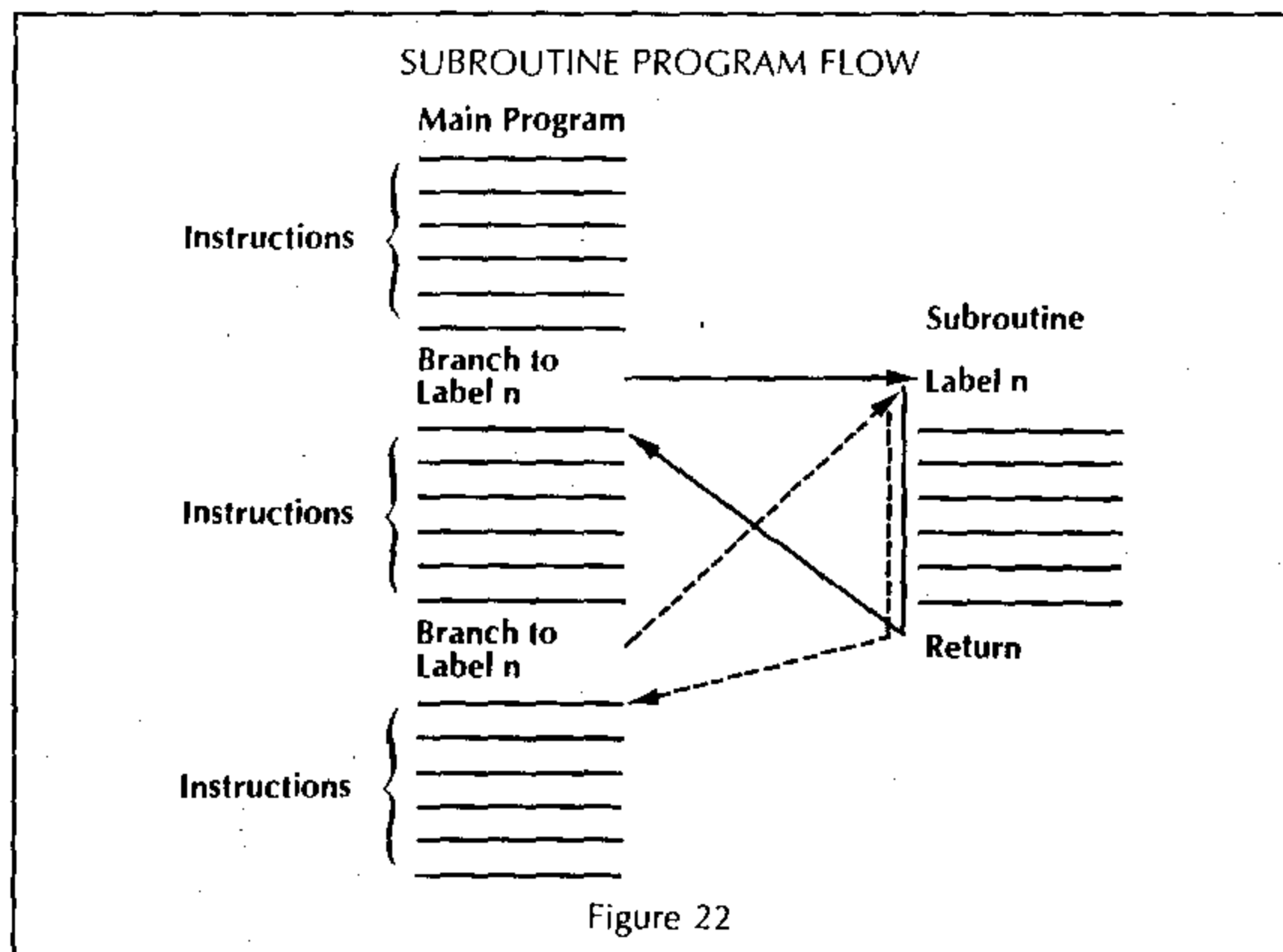
The Branch instruction is similar to the Jump instruction in that it allows you to go from one point to another in program memory. However, it differs from Jump in that Branch causes the step number from which the Branch was made to be saved so that you may later return to the point in the program from which the Branch was made and continue executing from the instruction following the Branch.

The procedure for loading the Branch instruction is as follows:




1. With the computer in Load mode, press  .
2. Enter the label of the step to which the Branch is to be made by pressing one of the keys:  - ,  or .

Return

The Return instruction returns program control to the memory step immediately following the step from which the last Branch occurred. The key sequence used is



Rules for the Use of Subroutines

1. The first instruction of a subroutine must be a Label instruction, .
2. The last instruction executed in a subroutine must be a Return instruction, . If you fail to load a Return or Jump out of a subroutine back to the main program, the program will not operate correctly.
3. Any program instruction can be loaded after the Label and before the Return (except another Return).
4. You can Branch to a subroutine from any place in program memory.
5. Subroutines may be nested. If you are in a subroutine, you can branch to another subroutine, which can branch to another, etc.... Six levels of branching are permitted. Note that this does not limit the number of times you can branch to a subroutine.
6. Executing a Return without a previous Branch may cause a meaningless value to appear in the display and prevent further operation. To recover, press  followed by any label.

Sample Program Using a Subroutine

Suppose we wish to compute values of

$$Y = \frac{\sqrt{1 + \ln A}}{\sqrt{1 + \ln B} + \sqrt{1 + \ln C}}$$

In this equation an expression of the form $\sqrt{1 + \ln X}$ appears three times. Rather than program this expression three times, you may write a labeled subroutine that evaluates the expression; then the main program branches to it as needed.

The main program is used to enter values of A, B and C, to branch to the subroutine and to display the calculated value of Y. The flowchart for the main program and subroutine is shown in Figure 23. It is assumed that the values of A, B and C have previously been stored in registers 1, 2 and 3, respectively.

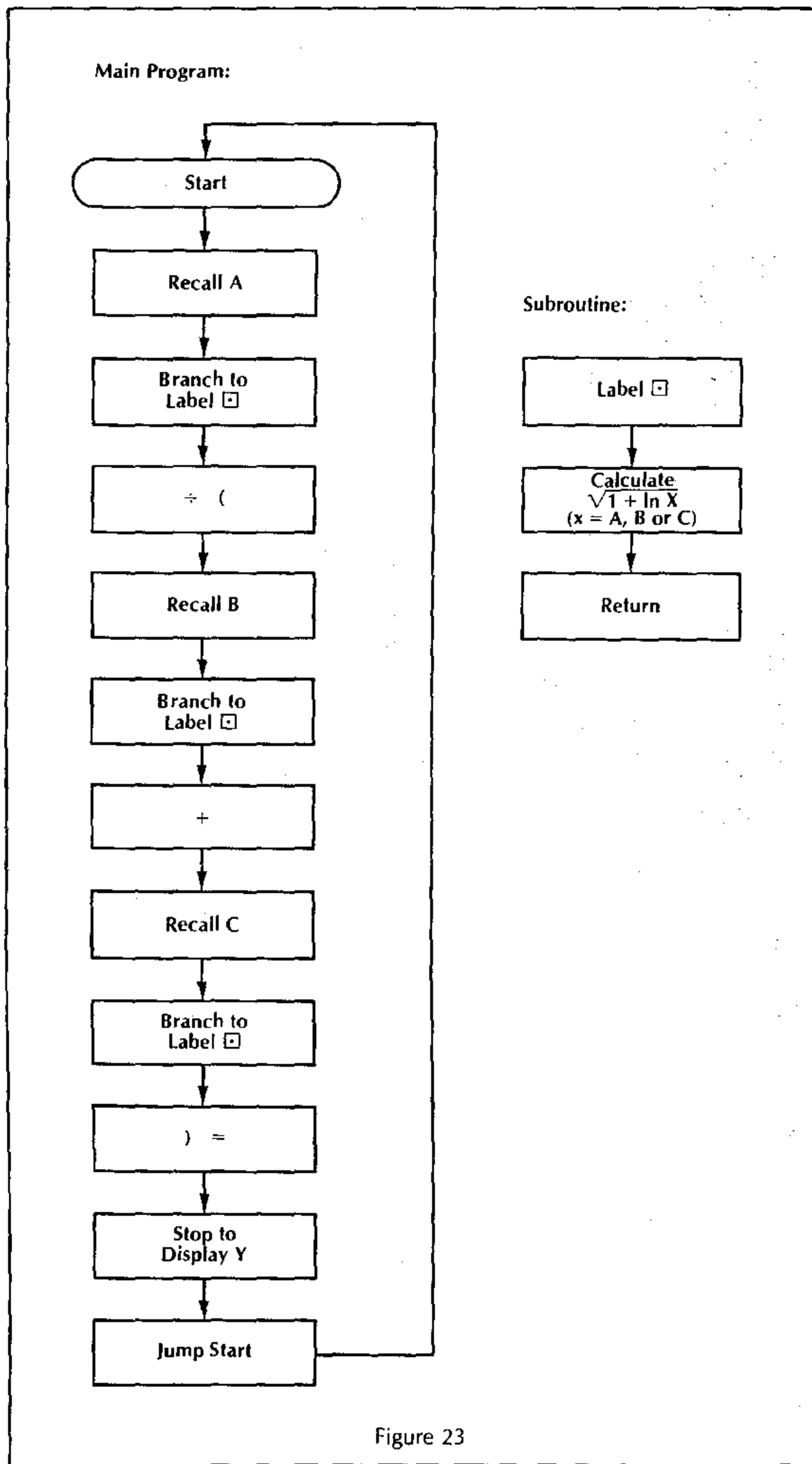




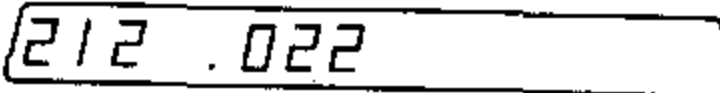



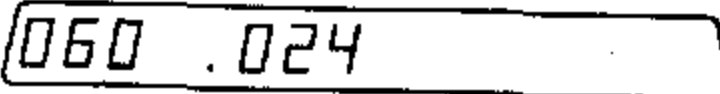



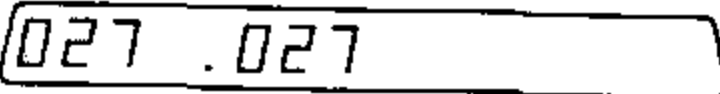

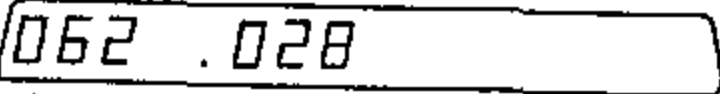


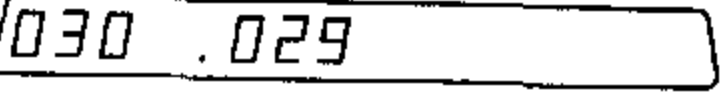

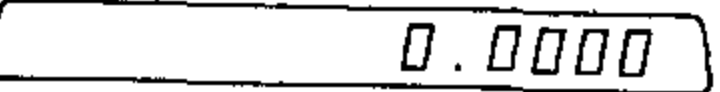
Figure 23

To load the main program



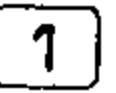

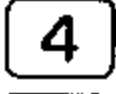


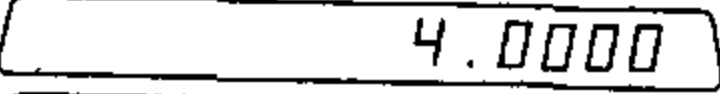
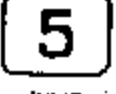


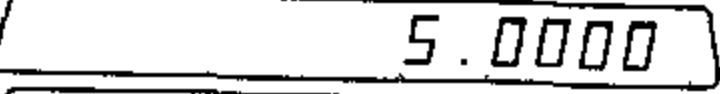




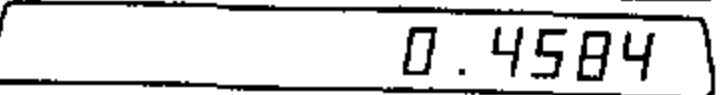
Do This	See This	Comments
	001 .001	Start loading the program at step 001.
	001 .003	Place A in the Entry register.
	012 .005	Branch to the subroutine at label <input type="checkbox"/> to calculate $\sqrt{1 + \ln A}$.
	026 .007	Place B in the Entry register.
	002 .009	Branch to the subroutine at label <input type="checkbox"/> to calculate $\sqrt{1 + \ln B}$.
	012 .011	
	021 .012	Place C in the Entry register.
	003 .014	Branch to the subroutine at label <input type="checkbox"/> to calculate $\sqrt{1 + \ln C}$.
	012 .016	
	027 .017	Calculate $\sqrt{1 + \ln B} + \sqrt{1 + \ln C}$.
	020 .018	Calculate Y
	033 .019	Display Y
	033 .021	Jump to step 001 to repeat the calculation using different values of A, B and C

Decisions and Conditional Testing

The subroutine may be loaded anywhere in program memory. For convenience we load it immediately following the main program.

Do This	See This	Comments
 LABEL 		Define label 
 		Calculate Ln X (X = A, B, or C).
  		Calculate 1 + Ln X.
		Calculate $\sqrt{1 + \text{Ln } X}$.
 		End of subroutine. Program returns to execute the step immediately after the last Branch that was executed.
		Place computer in Run mode to execute the program.

To run the program using the values A = 3, B = 4 and C = 5.

Do This	See This	Comments
  		Store the value of A in register 1.
  		Store the value of B in register 2.
  		Store the value of C in register 3.
 		Jump to step 1.
		Execute the program. The value of Y is displayed.

The computed value of Y = 0.4584

In programming, one often wishes to jump to another program or program segment or branch to a subroutine only if some condition is satisfied. The 326 Scientist is capable of testing the number in the Entry register to see if it satisfies a particular condition, and, if the condition is satisfied, performing the required jump or branch. You can test the number in the Entry register for the following conditions:

- Greater than zero (> 0)
- Less than zero (< 0)
- Equal to zero ($= 0$)
- Greater than or equal to zero (≥ 0)
- Less than or equal to zero (≤ 0)
- Not equal to zero ($\neq 0$)

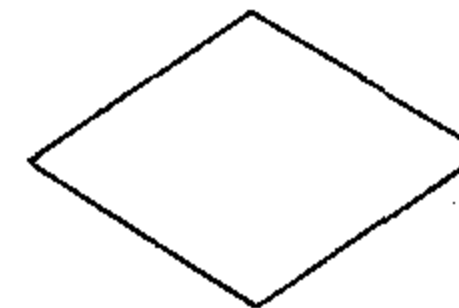
The 326 Scientist can also test whether an entry has been made from the keyboard.

A conditional test always asks a question. If the answer is yes, the program jumps or branches to the label that you specify. If the answer is no, the computer continues to execute the program steps that follow the test.

Flow Chart Symbol for Test

At this point, we will add another symbol to our "vocabulary" of flowchart symbols. A diamond-shaped box is used to indicate a test for some condition.

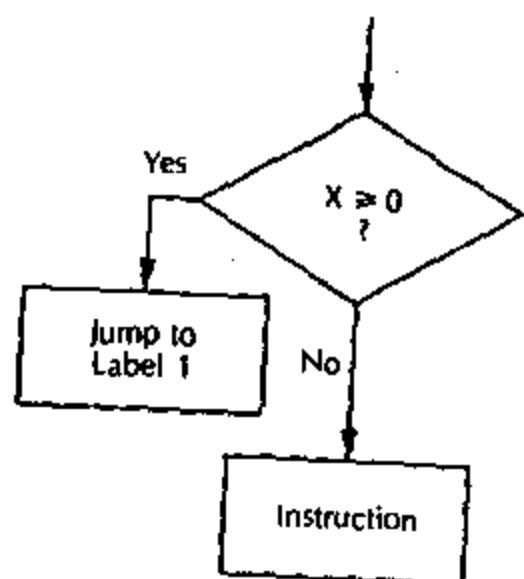
Symbol



Meaning

Test. A test is made for the condition indicated by the words in the diamond. One flowline enters the diamond. Two or three flow lines exit from the diamond-shaped box. The exit flow lines are usually labelled "yes" or "no" (if there are two exits).

Example:



In the portion of a flowchart shown, a number X is tested to determine if it is greater than or equal to zero. If the answer is "yes", a Jump is made to a program segment labelled 1; if the answer is "no", the program continues in sequence.

Conditional Jump

The key sequences, instruction codes and meanings for the *Conditional Jump* instructions are given in the following table. The label to which the Jump is to be made must be specified immediately after the Conditional Jump instruction.

Key Sequence	Instruction Code	Meaning
JUMP +	351	Jump, if the number in the E-register is positive, to the label specified in the following memory step.
JUMP -	352	Jump, if the number in the E-register is negative.
JUMP + -	353	Jump if the number in the E-register is positive or negative, but not zero.
JUMP =	354	Jump if the number in the E-register is zero.
JUMP + =	355	Jump if the number in the E-register is greater than or equal to zero.
JUMP - =	356	Jump if the number in the E-register is less than or equal to zero.
JUMP + - =	357	Jump if an entry has been made from the keyboard.

Example:

The key sequence **JUMP + = 2** means "Jump to label 2 if the number in the E-register is greater than or equal to zero."

Conditional Branch

The key sequences, instruction codes and meanings for the *Conditional Branch* instructions are given in the following table. The label to which the branch is to be made must be specified immediately following the Conditional Branch instruction.

Key Sequence	Instruction Code	Meaning
BRANCH +	361	Branch, if the number in the E-register is positive, to the label specified in the following memory step.
BRANCH -	362	Branch, if the number in the E-register is negative.
BRANCH + -	363	Branch, if the number in the E-register is positive or negative, but not zero.
BRANCH =	364	Branch, if the number in the E-register is equal to zero.
BRANCH + =	365	Branch, if the number in the E-register is greater than or equal to zero.
BRANCH - =	366	Branch, if the number in the E-register is less than or equal to zero.
BRANCH + - =	367	Branch, if an entry has been made from the keyboard.

Example:

The key sequence **BRANCH - 3** means "Branch to label 3 if the number in the E-register is negative."

Sample Program Using a Conditional Jump

A Conditional Jump instruction is used in the following program that calculates the sum of the first 17 integers (1 + 2 + ... + 16 + 17). As shown in the flow chart (Figure 24), the number in register 0 is tested to see if it equals 17. If it equal 17, a jump is made; if it doesn't equal 17, the program continues to the next memor step in sequence.

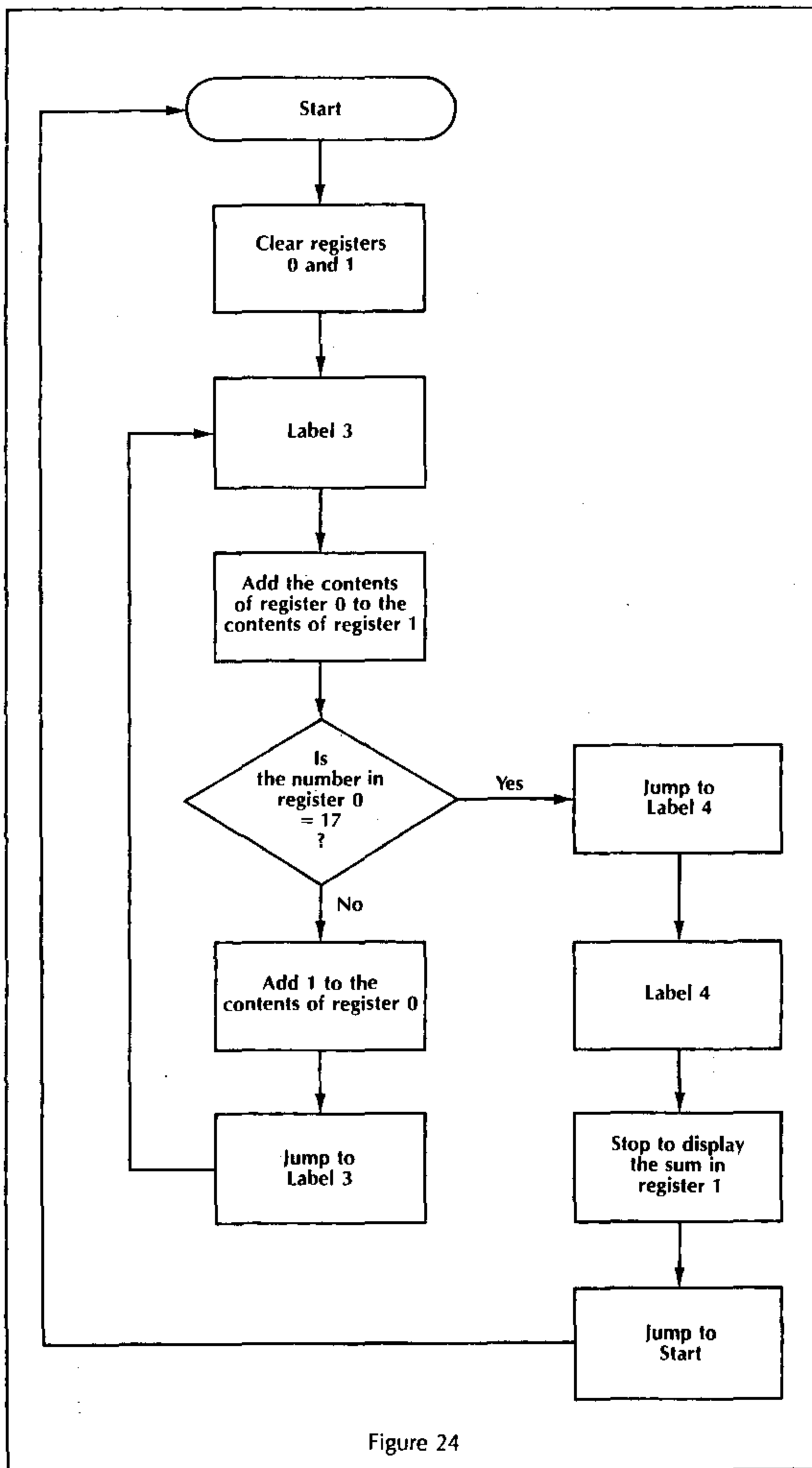


Figure 24

The program is loaded as follows:

Do This	See This	Comments
		Clear registers 0 and 1.
		Label 3 is at step 006.
		Add the number in register 0 to the number in register 1.
		Subtract 17 from the number in the E-register.
		If the number in the Entry register equals 0, jump to label 4. Otherwise continue.
		Increment Register 0.
		Jump to Label 3 at step 006.
		Label 4 is at step 023.
		Stop to display the sum in register 1.
		Return to the start of the program.
		Switch to Run mode to prepare for program execution.

To run the program:

Do This	See This

The sum of the first 17 integers is 153.

Sample Program that Tests whether an Entry Has Been Made

In this program we wish to calculate

$$y = a^{2.7}$$

for a number of values of a. The flowchart for this program is shown in Figure 25.

The Identifier 1 is used to indicate that a new value of a is to be entered. If an entry has been made, the program jumps to Label 2 and performs the calculation. If no entry has been made (indicating all calculations have been done) the Identifier 8 is displayed and the program is terminated.

To load the program

Do This	See This	Comments
1	001 .002	Enter 1.
9	111 .003	Identifier.
START STOP	033 .004	Stop to enter x.
JUMP + - =	357 .005	Jump if an entry has been made to Label 2.
2	002 .006	Enter 8.
8	010 .007	Identifier.
9	111 .008	End of program.
START STOP	033 .009	Move to step 020.
(Press FWD 11 times)	.020	Label 2 is arbitrarily placed at step 020.
LABEL 2	202 .021	a ^x function.
a ^x	025 .022	Enter exponent.
2 . 7	007 .025	Calculate y.
=	020 .026	Display y.
START STOP	033 .027	Switch to Run mode.
Run Step Load	0 .0000	

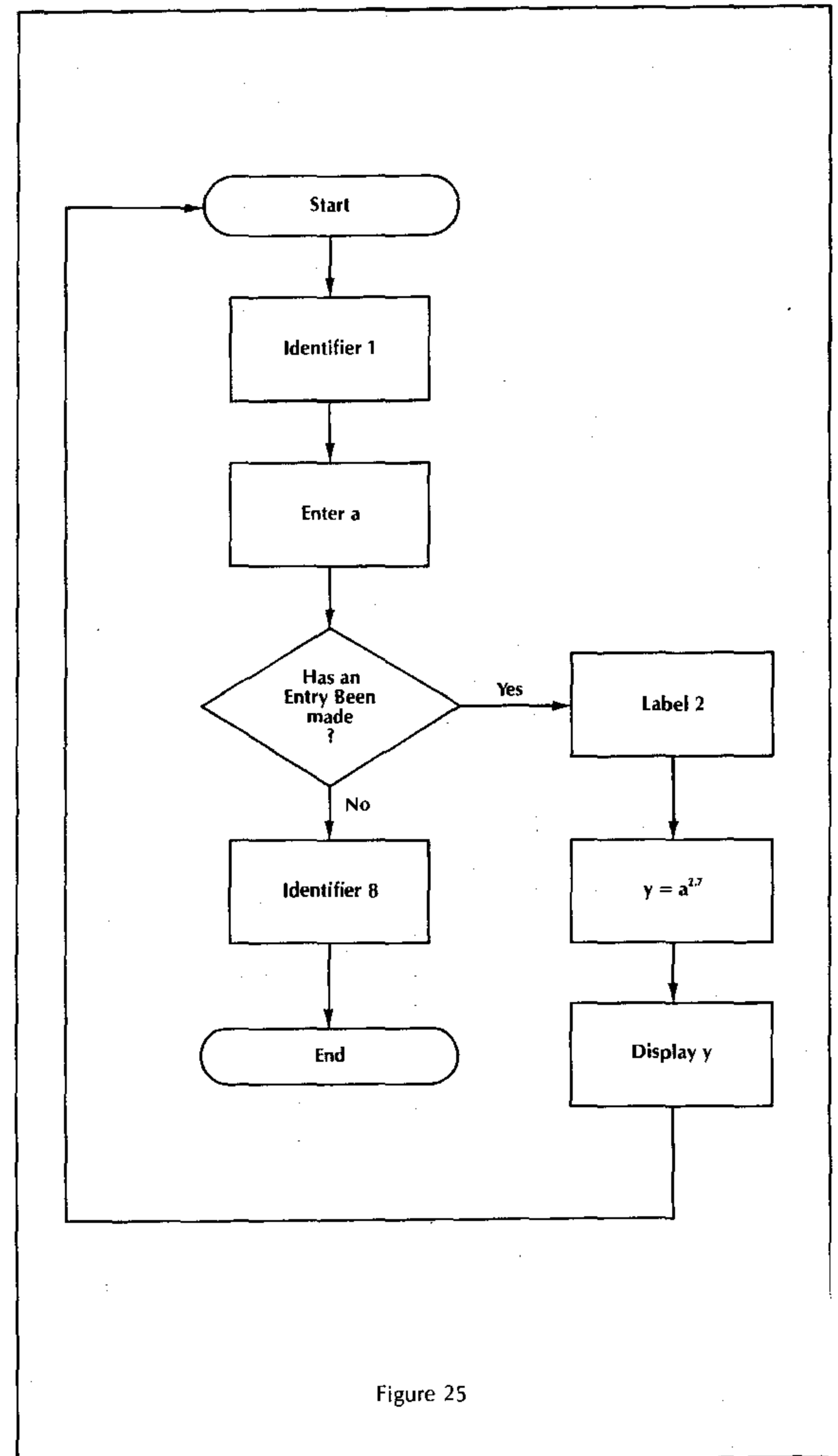


Figure 25

To run the program for values of $a = 2, 3$ and 4 :

Do This	See This	Comments
START STOP	1.	Start program execution. Identifier 1 is displayed.
2 START STOP	6.4980	Value of $y = 2^{2.7}$
START STOP	1.	Ready for a new value of a .
3 START STOP	19.4190	Value of $y = 3^{2.7}$
START STOP	1.	
4 START STOP	42.2242	Value of $y = 4^{2.7}$
START STOP	1.	Ready for a new value at a .
START STOP	8.	No entry was made. Identifier 8 is displayed indicating the end of the program.

Conditional Tests on Numbers other than Zero

Occasionally you may wish to test a number to see if it is less than, greater than or equal to a particular nonzero number. You may use the Conditional Jump or Conditional Branch instructions previously described for this purpose. First you perform a subtraction so that the test can be made in terms of a comparison with zero. For example, suppose you want to perform a Branch if a calculated number, Z , is greater than 100. Simply subtract 100 from Z and then perform the Branch if $Z - 100 > 0$.

In the following table, flowchart symbols of the various test conditions are shown along with the instruction sequences that will accomplish the tests. The label, l , may assume the values **0** - **9**, **.** or **CHG SIGN**.

Flowchart Symbol	Test Instructions
	Subtract N from the number in the Entry register JUMP + l or f BRANCH + l
	Subtract N from the number in the Entry register JUMP - l or f BRANCH - l
	Subtract N from the number in the Entry register JUMP = l or f BRANCH = l
	Subtract N from the number in the Entry register JUMP + = l or f BRANCH + = l
	Subtract N from the number in the Entry register JUMP - = l or f BRANCH - = l
	Subtract N from the number in the Entry register JUMP + - l or f BRANCH + - l

Sample Program: Testing Nonzero Numbers

Suppose that a function, F , is to be evaluated using one of three different expressions depending on the value of the independent variable, R . Specifically,

$$\begin{aligned}
 F &= 10^{(1.78 - \log R)} && \text{when } R \leq 2200 \\
 F &= 10^{(-3.7 + 0.65 \log R)} && \text{when } 2200 < R \leq 3600 \\
 F &= 10^{(-0.5 - 0.25 \log R)} && \text{when } 3600 < R
 \end{aligned}$$

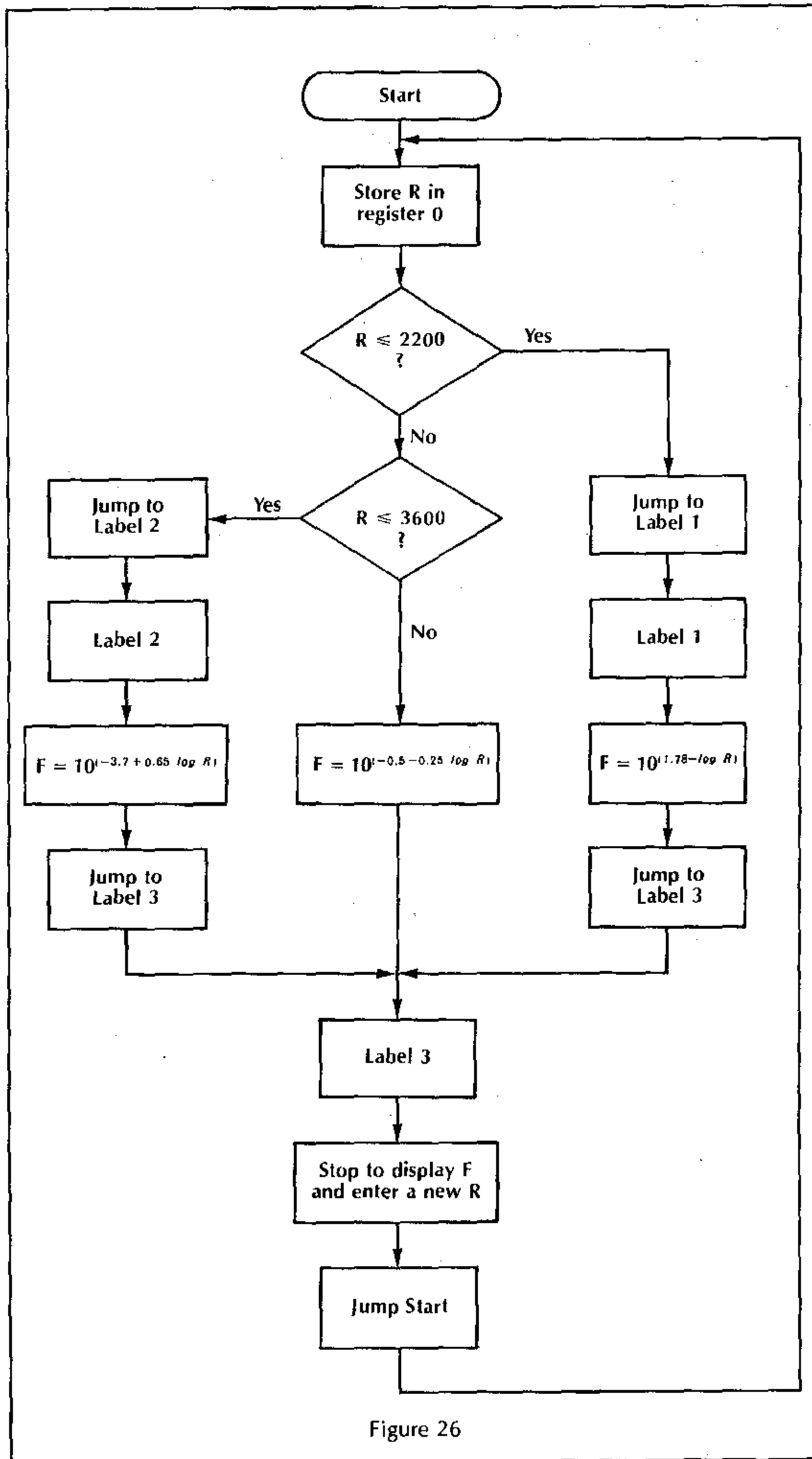


Figure 26

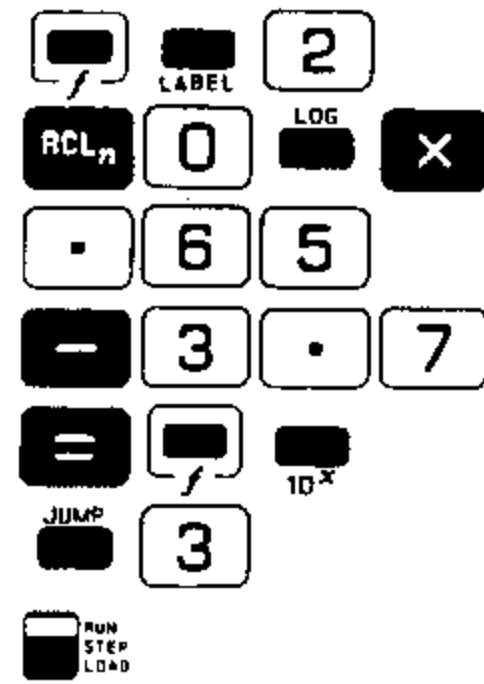
The flowchart for this program is shown in Figure 26 and includes two Conditional Jumps depending on whether $R \leq 2200$ and $R \leq 3600$. The value of R is assumed to be in the Entry register at the start of the program.

To load the program

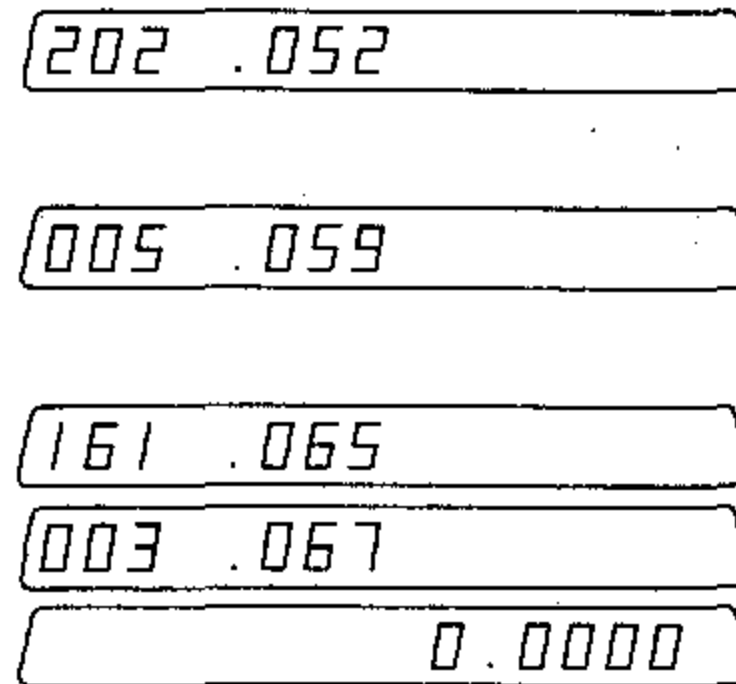
Do This	See This	Comments
JUMP START STOP RUN STEP LOAD	.001	Load program starting at step 001.
ST _n 0	000 .003	Store R in Register 0.
—		
2 2 0 0		
= JUMP —		
= 1	001 .011	Test for $R \leq 2200$. If true, jump to label 1.
CHG SIGN		
3 6 0 0		
RCL _n + 0	000 .018	Test for $R \leq 3600$. If true, jump to label 2.
JUMP — = 2	002 .020	
RCL _n 0 LOG X		
CHG SIGN . 2 5	005 .028	If R is not True for the above test, then $R > 3600$ and $F = 10^{(-0.5 - 0.25 \log R)}$
— . 5 =		
10 ^x	161 .033	
10 ^x 3	203 .034	Label 3 is at step 033.
START STOP JUMP START STOP	033 .037	Stop to display F and enter a new R. Then jump to step 001.
LABEL 1	201 .038	Label 1 is at step 037.
RCL _n 0 LOG		
CHG SIGN +		
1 . 7 8		
= 10 ^x	161 .049	If $R \leq 2200$, then $F = 10^{(1.78 - \log R)}$
JUMP 3	003 .051	Jump to label 3 to display value of F.

Stepping Through a Program

Do This



See This



Comments

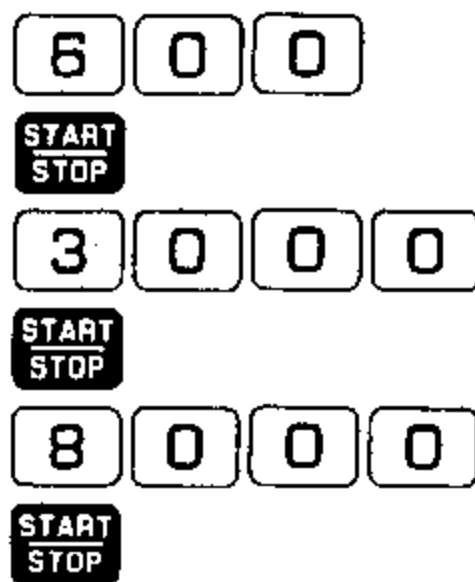
Label 2 is at step 051.

If $2200 < R \leq 3600$,
then
 $F = 10^{(-3.7 + 0.65 \log R)}$

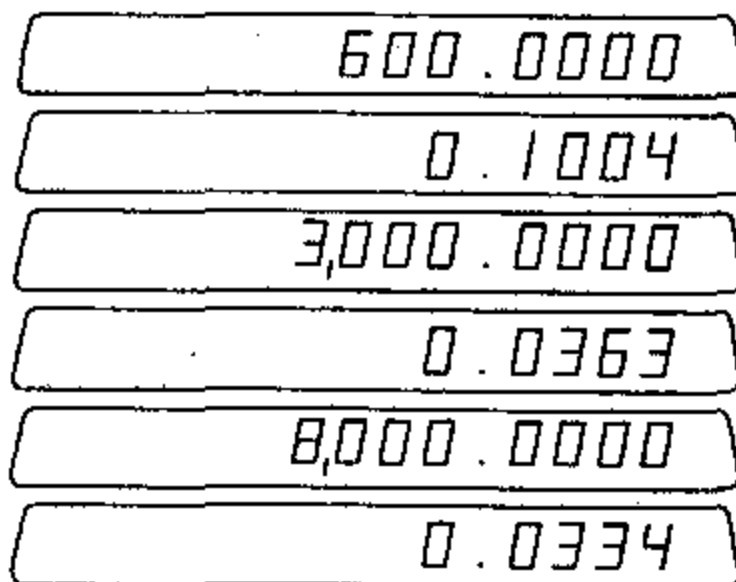
Jump to label 3 to display value of F.
Switch to Run mode in preparation for program execution.

To run the program for values of $R = 600, 3000$ and 8000

Do This



See This



Comments

Enter $R = 600$.


Value of F for $R = 600$.


Enter $R = 3000$.



Value of F for $R = 3000$.




Enter $R = 8000$.

Value of F for $R = 8000$.

The 326 Scientist permits you to execute a program one step at a time. Using this feature, you may observe program execution closely. This is particularly useful in determining what modifications a program may require. To step through a program, first set the RUN/STEP/LOAD switch to .

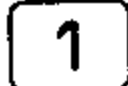

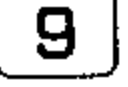










When in Step mode, each time you press  the following things happen:

- The display shows program information for about one second. The current memory step, its code and the previous and next code are also shown, in the Load format previously described.
- The instruction at the current location is executed and the result of the instruction is displayed. (Multi-step instructions like  are executed together as one instruction.)
- After the instruction is executed, the program stops with the current contents of the Entry register in the display. You must press  again to continue stepping through the program.







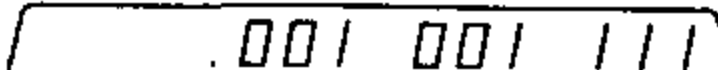
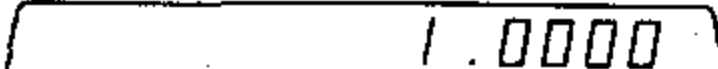

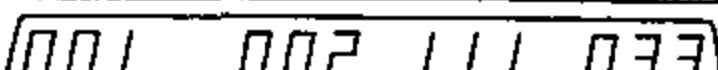


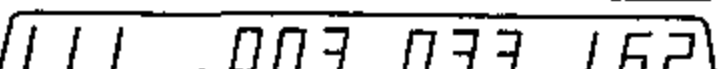
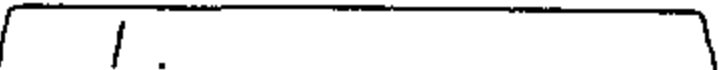


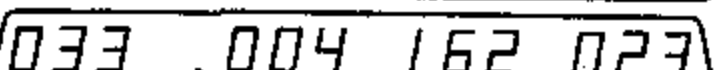
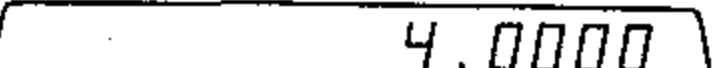


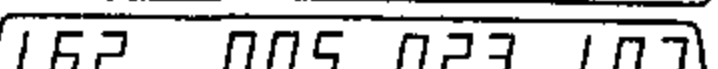
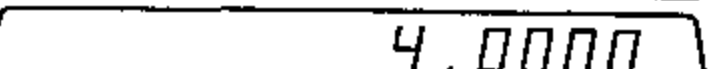

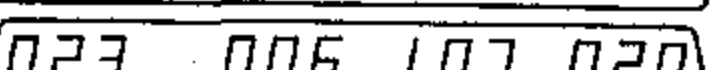


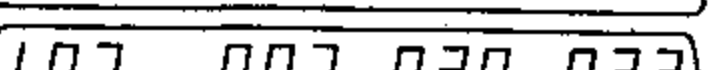
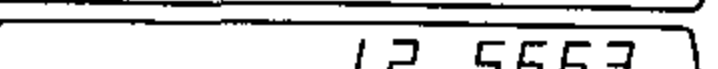


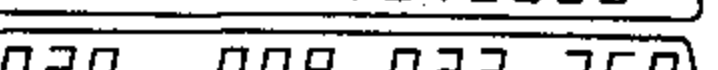
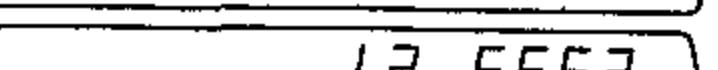

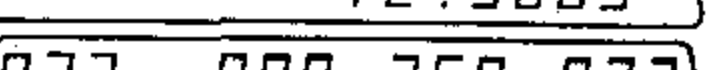
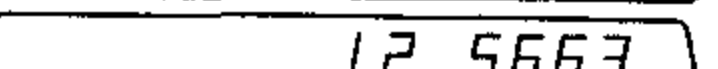





If you move the RUN/STEP/LOAD switch from  to , pressing  will cause program execution from the current step in memory.

Example:

To demonstrate stepping through a program, enter the following program that calculates the area of a circle ($A = \pi r^2$) into memory starting at step 001.



Memory Step	Instruction Code	Key	Comment
001	001		Enter 1.
002	111	 	Identifier.
003	033		Stop to enter r.
004	162	 	Calculate r^2 .
005	023		
006	107	 	
007	020		Calculate πr^2 .
008	033		Display A.
009	350		
010	033		Jump to step 001.

To step through this program with $r = 2$

Do This	See This	Comments
   		Jump to step 001 and switch to Step mode.
		Displays load format.
		Executes the instruction 001 (Entry of 1) in step 001.
		Display after execution. Displays load format.
		Executes the Identifier instruction in step 002.
		Display after execution.
		Executes the Stop instruction at step 003.
2		Display after execution.
		Enter $r = 2$.
		Executes the  instruction.
		Displays value of r^2 .
		Executes X instruction.
		Value of R^2 .
		Executes π instruction.
		Displays π .
		Executes  instruction.
		Calculates $A = \pi r^2$.
		Executes Stop instruction.
		Displays value of A.
		Executes   (a double instruction)
		The program is back to Step 001.
		Executes this instruction.



It is possible to make keyboard calculations when stepping through a program. Make sure that your keyboard arithmetic does not interfere with any program arithmetic in progress. To keep from interfering, you may need to isolate your keyboard arithmetic with parentheses. If your program uses parentheses, make sure that the sum of the program and keyboard parentheses doesn't exceed four.








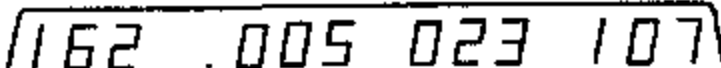



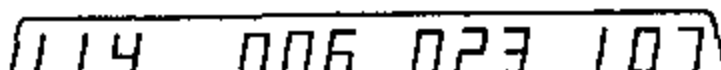


The Pause Instruction

The Pause instruction allows you to briefly display intermediate results during the execution of a program. The key sequence is  . This causes the program to pause about three seconds during execution to display the contents of the Entry register.

Multiple answers can be displayed by including a Pause instruction after each answer is calculated. Each answer can be identified by pausing to display an identifier just before the answer is displayed.

Example:

Suppose in the previous program calculating the area of a circle, we wish to pause to display the value of r^2 as well as displaying the final value of A. We will insert a Pause instruction   immediately after the calculation of r^2 at step 004.

Do This	See This	Comments
  		Jump to step 001. The number 1.0000 was previously in the Entry register.
		Switch to Load mode.
(Press  4 times)		Move forward to step 005, following the calculation of r^2 .
  		Insert a Pause instruction.
		Switch to run mode to execute the program.

To execute the program with $r = 2$

Do This	See This	Comments
		Identifier 1 displays.
		Enter $r = 2$.
		Start execution. The value of $r^2 = 4$ displays briefly, followed by the value of $A = 12.5663$.

Look Ahead Feature

When the 326 is stopped while executing a program, it may be desirable to look ahead to see what program step will be executed next without actually executing the next step.

To do this, press when a program is at a Stop. The display changes temporarily to Load format and the instruction code to be executed next is displayed just to the right of the next memory step number.

The Look Ahead feature is also useful when an error occurs in the execution of a program. If occurs during execution, press to see the number in the display just prior to . Then press to see the instruction that caused the Error condition to occur.

Display of Program Listing

With the RUN/STEP/LOAD switch set to , pressing briefly displays each program instruction in sequence. The display shows the program listing in Load format. No execution occurs.

Some Programming Hints and Techniques

This section describes some useful programming hints and techniques that may help you to write programs more efficiently.

Handling Variables and Constants

In general there are three types of data used within a program:

1. Data values that change every time the program is run
2. Data values that change infrequently
3. Data values that never change within a program

For most efficient programming and execution, the following rules should be followed:

1. A data value that changes each time the program is run should be entered at a Stop instruction within a program.
2. Values that are the same for many runs of the program should be stored in data storage registers. Thus, they need not be entered each time the program is run, but can still be changed whenever it is necessary.
3. Numbers that are constant within the program, and never change, can be entered as program steps. Once the program is written, they need never be entered again.


You may wish to store constants that have many digits in data storage registers rather than enter them in the program where one memory step must be used for each digit of the constant.

Stop Instructions in the Middle of Arithmetic Sequences

Using a instruction in the middle of an arithmetic sequence in a program is not recommended, since keyboard operations done while the Stop is in effect could change or interrupt the algebraic sequence in progress. Thus, program sequences such as

should be avoided. If a or other arithmetic operation is performed while at the Stop, the instructions are lost and the program will not calculate the intended answer.

Use of Identifiers

When several numbers must be entered within a program using different Stop instructions, it is very useful to identify each Stop with a unique number in the display. This may be accomplished by using the identifier instruction ( 9) with a numerical value.

Identifiers can also be used in displaying multiple answers with the Pause instruction.

Direct Register Arithmetic in Programs

When doing arithmetic operations that use the contents of registers, direct register arithmetic can often save program steps.



For example, to multiply the contents of register 5 by the number 2 and store the result in register 5, the sequence

uses seven program steps.

The same operation can be shortened to four steps using the sequence

Also, when these are loaded as program steps, only three instructions are loaded because   is loaded as one instruction.

Sample Program: Evaluation of a Polynomial

We have now covered all the basic techniques for programming the 326 Scientist. To become more familiar with the techniques discussed so far, we will write and execute a program that evaluates the polynomial function

$$y = a_0 + a_1x + a_2x^2$$

where a_0 , a_1 , and a_2 are constants and x is a variable. The flowchart for this program is shown in Figure 27. In the first segment of the program, before Label 1, values of the constants a_0 , a_1 , and a_2 are stored in registers 0, 1 and 2, respectively. The second part of the program is the calculation of the value of the polynomial and a final jump back to Label 1 to allow a calculation of y for a new value of x .

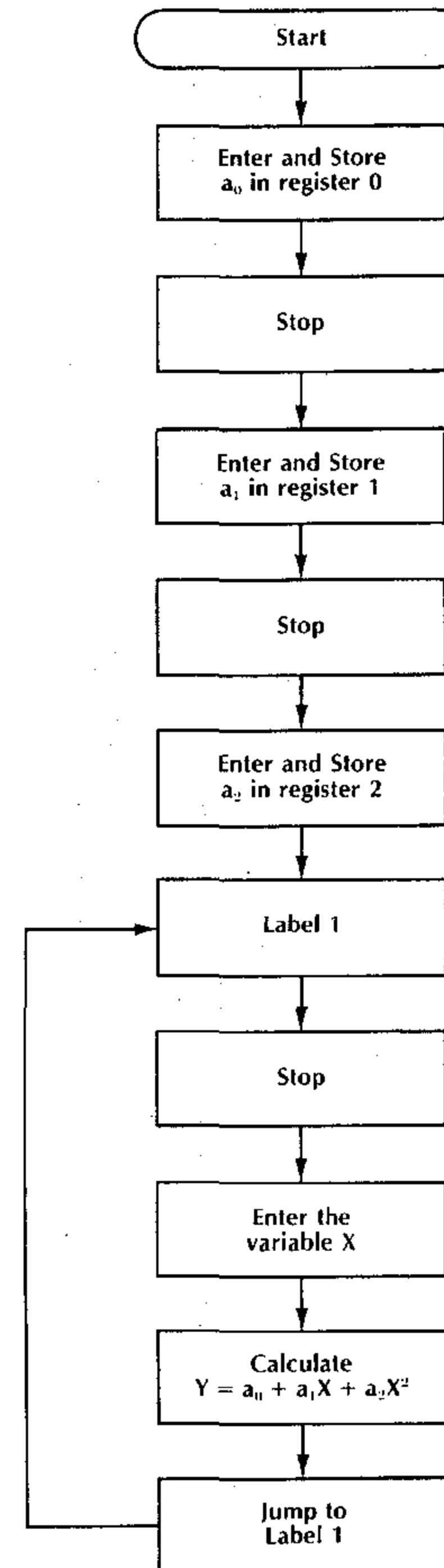




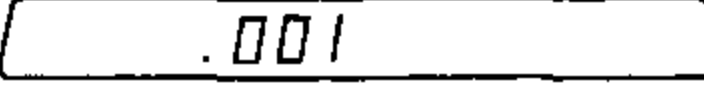

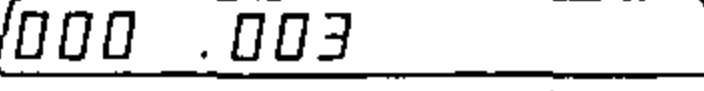

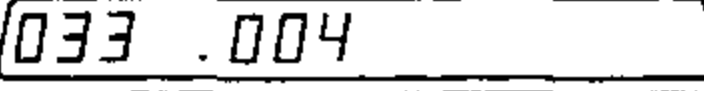

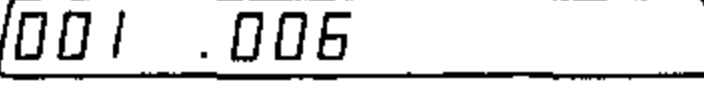

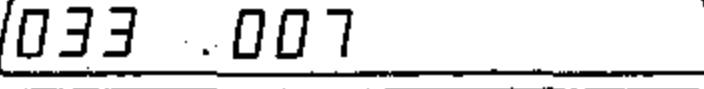

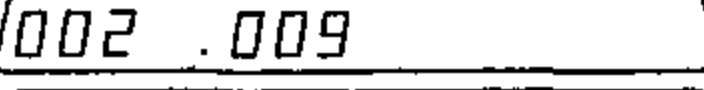

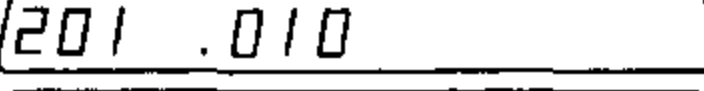

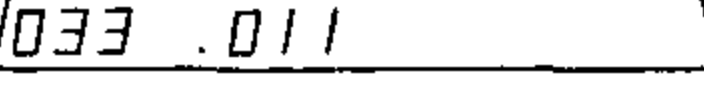

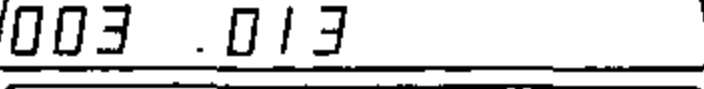

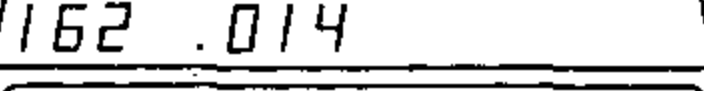


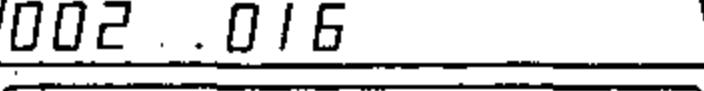


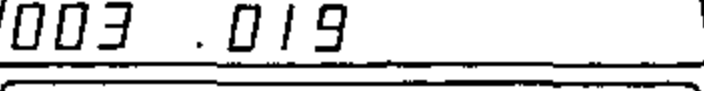



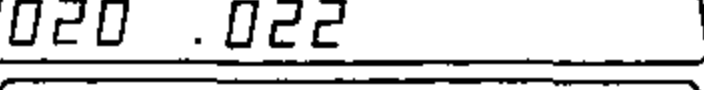


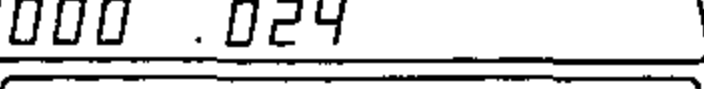

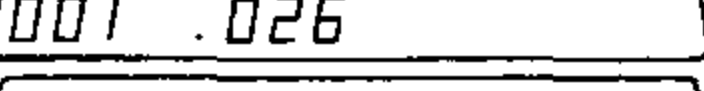

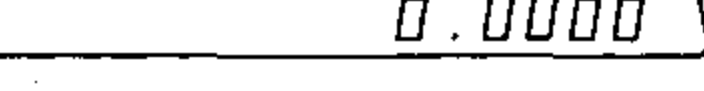


Figure 27

To Load the Polynomial Program

Do This	See This	Comments
   		Start loading the program at memory step 001.
 0		Store the number in the E-register in register 0.
		Stop to enter value of a_1 .
 1		Store a_1 in register 1.
		Stop to enter value of a_2 .
 2		Store a_2 in register 2.
 1		Assign the Label 1.
		Enter x and display the previously calculated value of y .
 3		Store the value of x into register 3.
		Calculate x^2 .
  2		Calculate a_2x^2 using direct register arithmetic.
  3		
  1 		Calculate $a_1x + a_2x^2$.
  0		Calculate $a_0 + a_1x + a_2x^2$.
 1		Jump to Label 1.
		Set the switch to Run mode in preparation for running the program. The display shows the same value that it held prior to the Load condition.

To run this program with the constants

$$a_0 = 4$$

$$a_1 = 2$$

$$a_2 = -5$$

and x values of

$$x = -3$$

$$x = 7$$

Do This

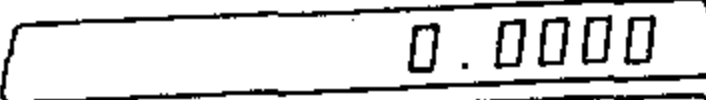
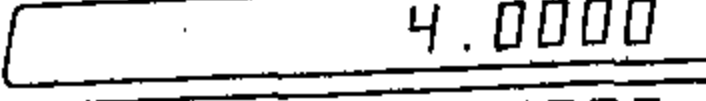
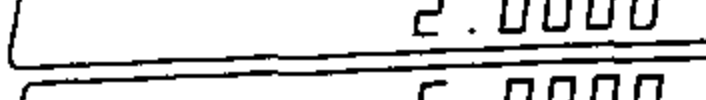
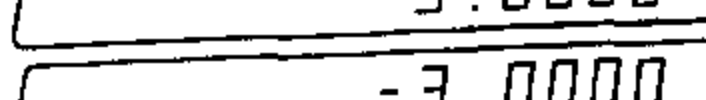
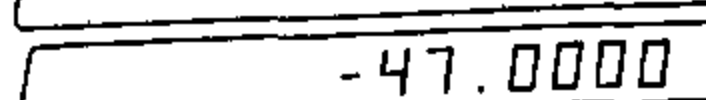

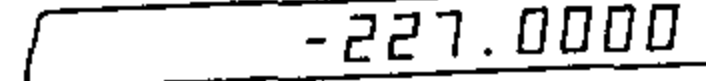











See This

Comments

Jump to step 001.
 Enter value of a_0 .
 Enter value of a_1 .
 Enter value of a_2 .
 Enter value of x .
 Calculate y for $x = 3$.
 Enter new value of x .
 Calculate y for $x = 7$.

You may, if you wish, execute a  to return to the beginning of the program to enter new values of a_0 , a_1 and a_2 , and then evaluate the new polynomial for different values of x .

You are encouraged to write other sample programs to solve problems in your special field of interest until you feel totally comfortable with the programming techniques presented so far.

The following section of this book describes how the programming capabilities of the 326 Scientist may be extended still further with the use of the Monroe 392 Tape Cassette Drive.

Section 5
Using The 392 Tape Cassette Drive

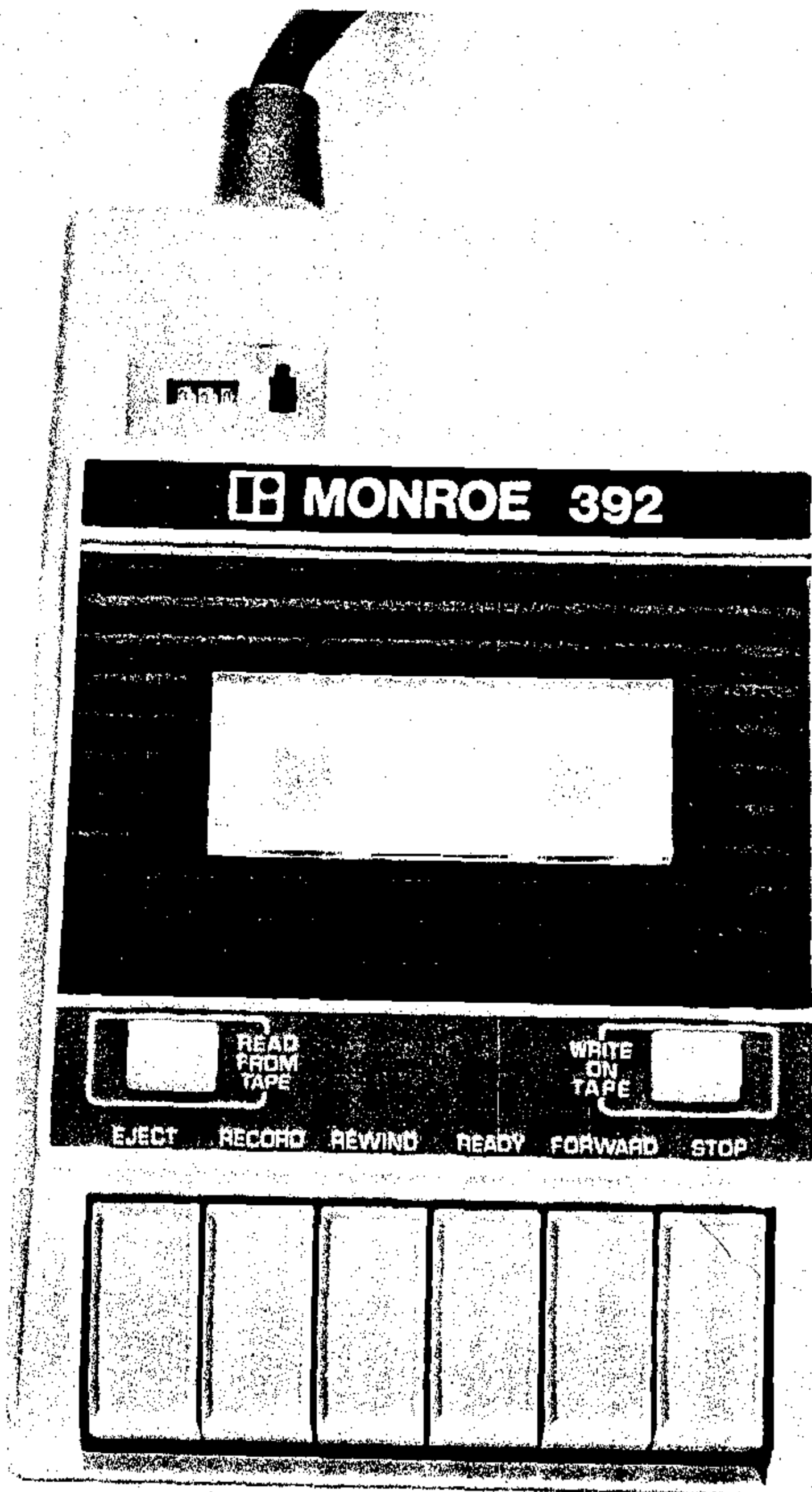


Figure 28

Why Use a Tape Cassette Drive?

The Monroe 392 Tape Cassette Drive (see Figure 28) greatly increases the programming capability of your 326 Scientist. First, you may use the Tape Cassette Drive to enter prepared programs into your computer. Monroe provides a comprehensive library of programs for a wide range of scientific, mathematical and engineering applications. This saves you the trouble of having to write your own programs to solve many types of problems.

You may store programs that you have written, permanently or temporarily, on magnetic tape, and enter them into the computer's program memory whenever you need them.

You may store data on tape, permanently or temporarily, and read it into the computer's data storage registers when required by your program.

The read and write operations of the magnetic tape drive can be placed under program control. Therefore writing programs and data onto magnetic tape and reading programs and data from tape can be done automatically by a program.

This ability to program tape drive operations greatly increases the *effective* size of the program memory and the data storage area.

For example, a large program may be divided into a number of segments, where each of the program segments is required to perform a specific part of the computation. The program segment currently in the computer's memory can include as its final instruction the command to read the next program segment from tape into memory. Thus a very large program could be executed in 160 program step segments. This is known as a *program overlay* technique, because the new program segment brought into the computer's memory overlays and replaces the program segment previously stored there.

Similarly, large amounts of data can be brought into the computer's storage registers from tape 12 data registers at a time. Input data could be brought into the computer from tape, computations could be done and the resulting data could be stored on tape. Then a new set of input data could be entered from tape and used for a new set of computations, and so forth.

Thus automatic program control of tape drive reading and writing operations permits you to write programs of considerably more sophistication than would be possible by programming only from the keyboard.

Connecting the Tape Drive

The 392 Tape Drive has a permanent signal cable with a connector at the end (see Figure 29). To connect the Tape Drive to the 326 Scientist, simply plug the connector into the socket which is recessed on the right side of the 326.

The cable is used to pass information to and from the Tape Drive. Power to run the Tape Drive is supplied through the 326 by means of this cable. The Tape Drive uses power only when the drive motor is running. The power consumption of the 392 Tape Cassette Drive is minimal and has little effect on the discharge time of the 326 Scientist's batteries. In the extreme case where the motor is kept running full time, the battery discharge time is reduced by approximately 15%.

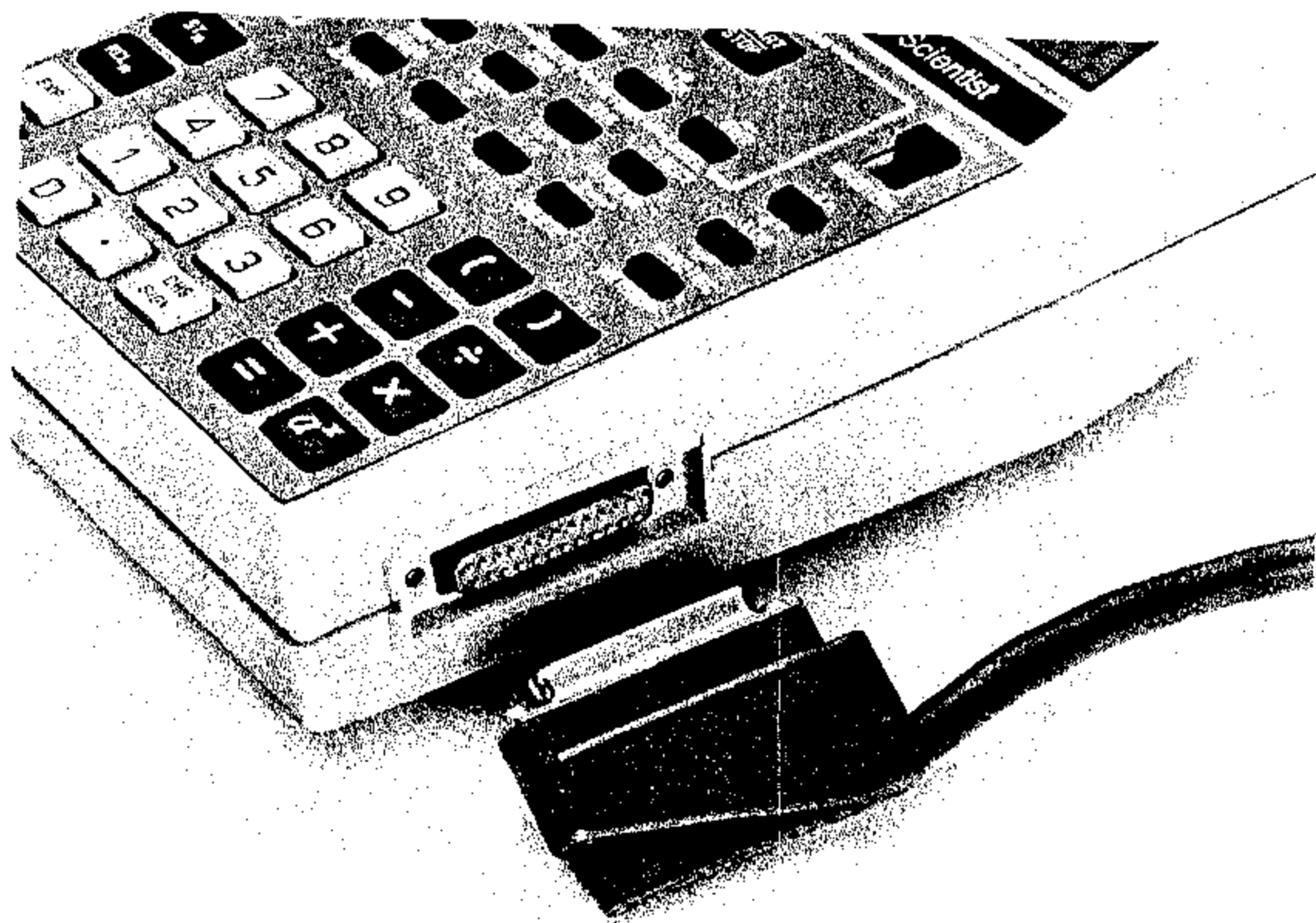


Figure 29

Tape Cassettes Used

Two kinds of tape cassettes may be used with the 392 Tape Cassette Drive. They are:

The Monroe Endless Tape Cassette

The Monroe Tape Cassette

Monroe Endless Tape Cassette

The endless tape cassette is a continuous loop of tape and is easily recognized by its red label. It can hold a maximum of fourteen *blocks* of information. Each *block* can hold the full capacity of program memory or the contents of a maximum of twelve data registers. Because only one side of the endless tape is used, it must be inserted in the tape drive so that the labeled side always faces up. The unlabeled side has a note of warning stating that the other side is to be inserted.

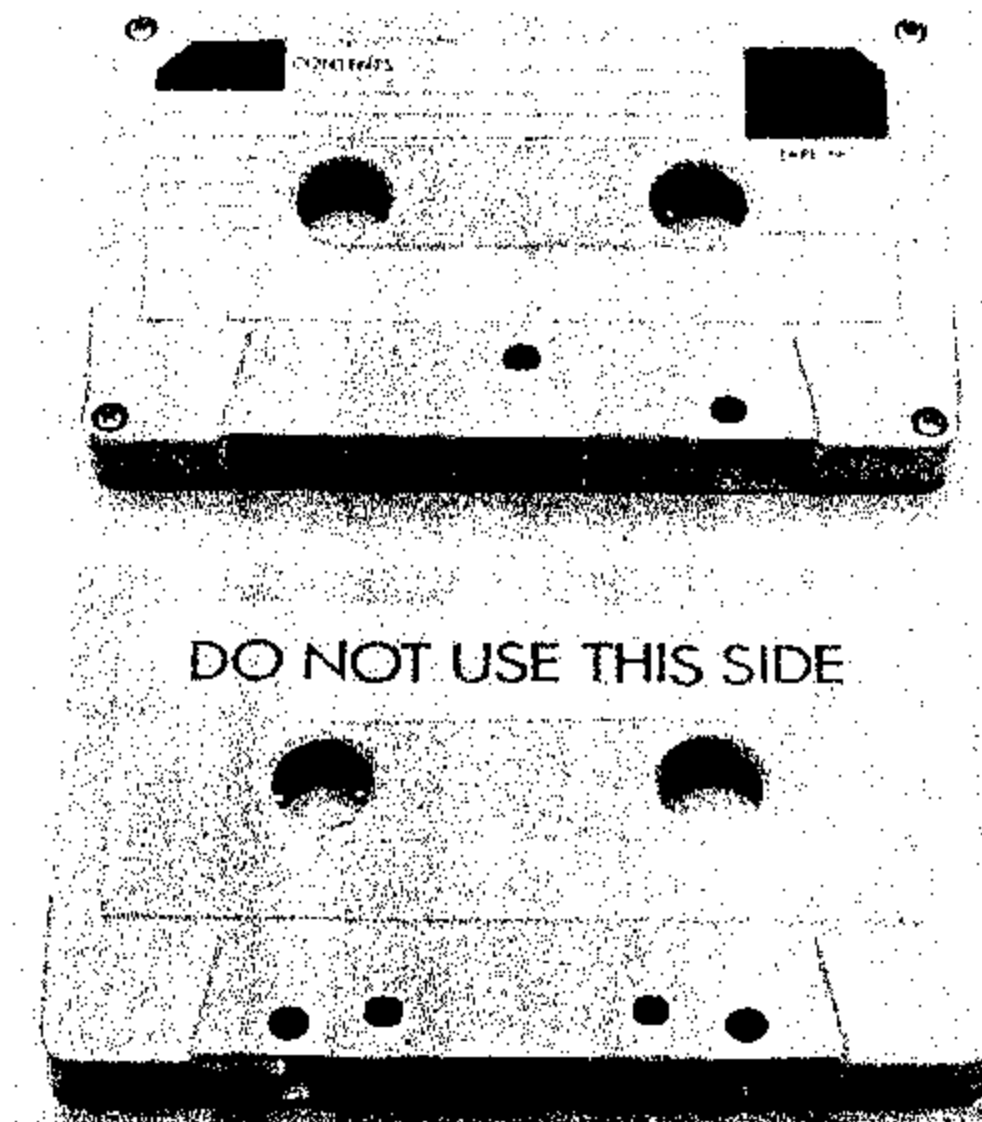


Figure 30

Tape Controls

Monroe Tape Cassette

The Monroe Tape Cassette is a conventional cassette that records on both sides, easily recognized by the blue label on the cassette. The two usable sides are designated Side A and Side B. Each side of the tape can hold up to 12 files of information. Each file contains fourteen blocks and each block can hold the full capacity of program memory or the contents of a maximum of twelve data registers.

Protection of Tape Against Accidental Erasure

After information has been recorded, cassettes may be protected from accidental erasure by removing a *Write Protect* tab. Of course, this prevents you from writing new information on that tape and implies that you are storing a program or data permanently rather than temporarily. Endless (loop) cassettes have only one write protect tab (see Figure 31). Conventional tape cassettes have two write protect tabs, one for each side of the cassette (see Figure 31).

Additional cassettes are available through your Monroe representative designated by the following part numbers:

CD8490443 for Monroe endless tape

CD8490435 for Monroe tape

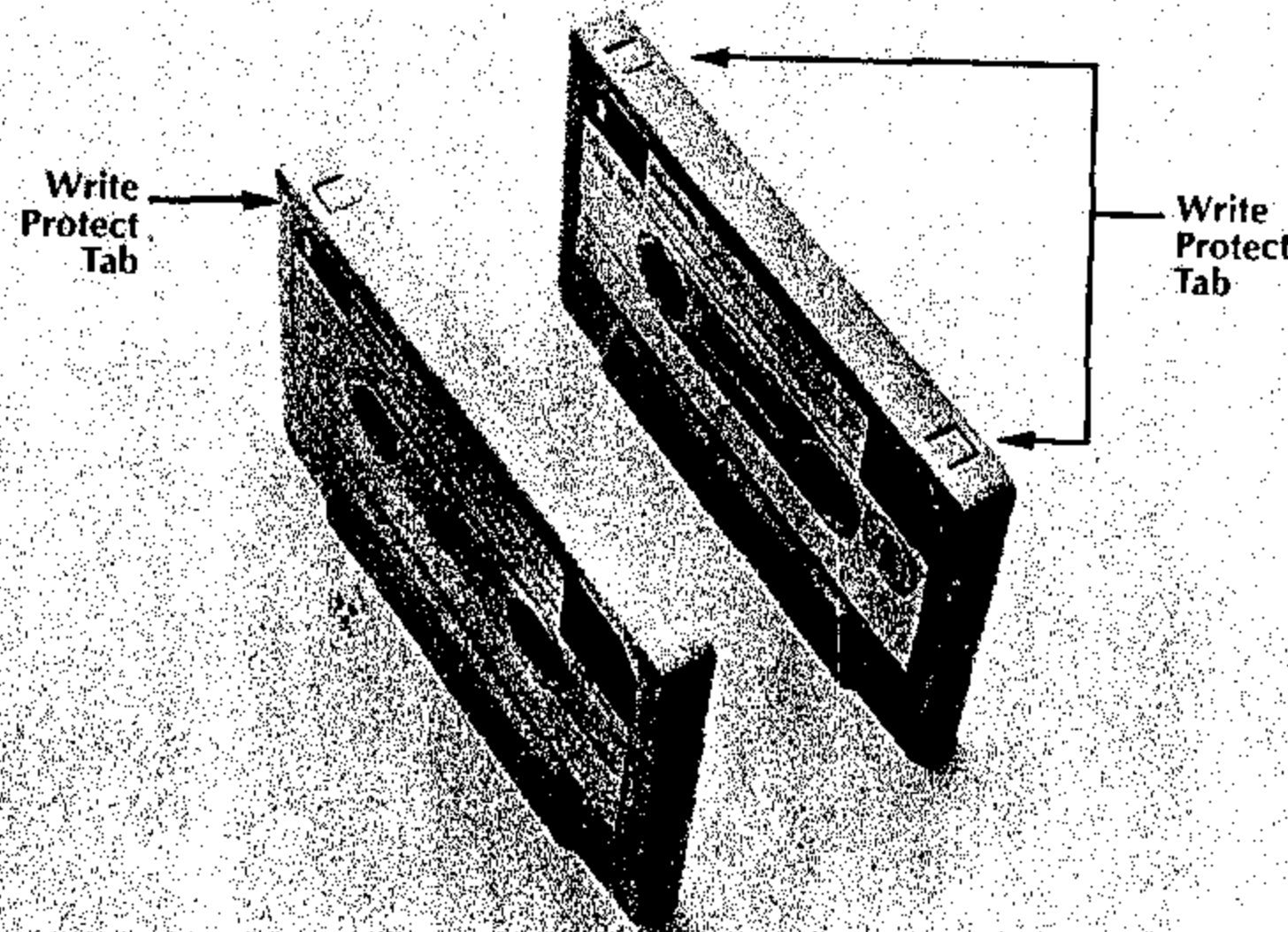


Figure 31

The controls used to operate the 392 Tape Cassette Drive are shown in Figure 28. All tape controls except the Program/Registers switch are on the cassette drive.

Control	Function
EJECT	Pressing this key lightly opens the cassette drive door to allow the insertion or removal of the cassette. Pressing more heavily, causes the tape cassette to be ejected.
RECORD	Allows data or programs to be recorded onto the tape from the data registers or program memory of the 326. Whether programs or data are recorded on tape depends on the setting of the Program/Registers switch. RECORD does not operate when the Write Protect tab is removed.
REWIND	This key causes two-sided tape (blue label) to move backwards at approximately 1 file every 2 seconds. The tape drive motor does not automatically turn off when the tape is fully rewound. This does not damage the tape cassette or the drive but will discharge the batteries after several hours. REWIND does not operate with the endless (red label) cassette.
READY	This key places the recording head against the tape. Ready must be pressed and latched to write onto tape or read from the tape. When writing onto tape, both READY and RECORD must be latched. When reading from the tape, only READY need be latched; RECORD may or may not be latched. However, protection against accidental writing is provided if RECORD is unlatched.
FORWARD	This key causes two-sided tape (blue label) to go forward at high speed. The tape motor does not automatically shut off when the tape is fully wound. This does not damage the tape cassette or the drive. FORWARD does not operate with endless (red label) tape.

STOP This key stops whatever tape motion is occurring and unlatches any other keys.

WRITE ON TAPE This key causes programs or registers to be written onto the tape from the 326. WRITE ON TAPE is also used to load a program instruction for automatic execution of WRITE ON TAPE.

READ FROM TAPE This key causes programs or registers to be read from the tape into the 326. Also used to load a program instruction for automatic execution of READ FROM TAPE.

PROGRAM/REGISTERS This switch is on the 326 Scientist. It selects whether the 326 is to write a program from memory or data from data storage registers onto tape. The PROGRAM/REGISTERS switch is only effective during the WRITE ON TAPE operation. When performing a READ FROM TAPE, the 392 automatically detects if the information read is a program or registers, and reads it into the 392 accordingly.

Tape Movement Counter

A digital counter that records tape movement is positioned at the upper left hand corner of the face of the 392 Tape Cassette Drive. This counter is extremely useful when you wish to move to a distant point on the tape since you may use the FORWARD key with the digital counter to move rapidly to the required part of the tape and then perform the remainder of the search at the normal Read/Write speed.



Tape Blocks and Preconditioning Tape

A *block* is a defined length of tape that can hold the full capacity of program memory or up to twelve data registers. All tape cassettes must be *preconditioned* prior to use on the 392. Preconditioning divides the tape into fourteen blocks. Each block is assigned a number in the sequence 01 through 14. Preconditioning erases all previous information on the tape. If you are using prepared program tapes exclusively and do not plan to write your own data or programs onto tape, you need not

be concerned with preconditioning tapes. All prepared program tapes that are provided to you by Monroe have already been preconditioned. Thus you may omit reading the following procedures and go to the discussion of Block Search (next topic).

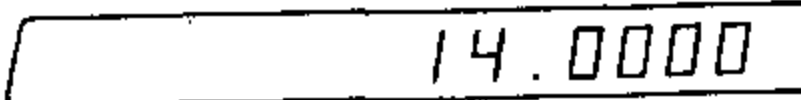
The procedures for preconditioning tapes are as follows:

Endless Tape (red label):

1. Connect and turn on the 326-392 system.
2. Insert the cassette to be preconditioned.
3. Press and latch READY and RECORD on the 392.
4. Press   on the 326 Scientist, then WRITE ON TAPE on the 392 Tape Drive.



The system automatically searches for the splice on the endless tape. This search may last as long as two minutes. The display remains blank during the search. After finding the tape splice, the block label numbers are written on the tape in sequence. This takes approximately one minute. After each block label is written, the display shows the two digit label for that block.

After block 14 is labeled the tape stops and the display shows:

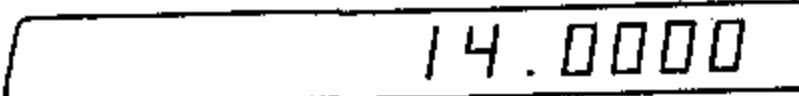


The Endless Tape cassette is now preconditioned and ready for use.

Two-Sided Tape (blue label):

1. Connect and turn on the 326-392 system.
2. Insert the cassette into the 392 cassette drive with the side that you wish to precondition facing up (Side A or Side B).
3. Press REWIND and fully rewind the tape.
4. Latch READY and RECORD on the 392 Tape Drive.
5. Press   on the 326, then WRITE ON TAPE on the 392.


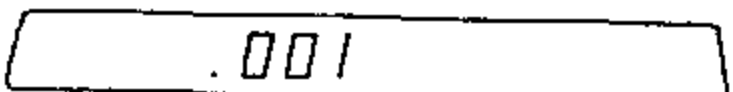
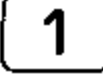

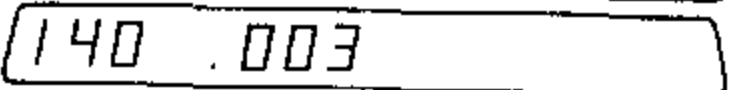
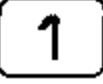


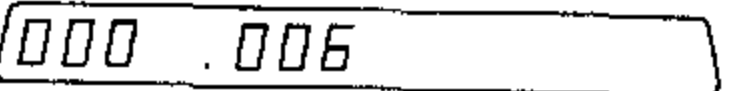

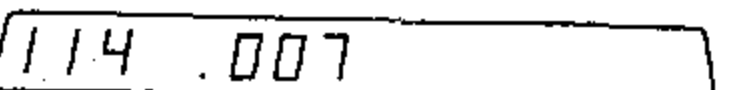

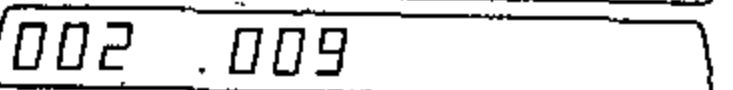
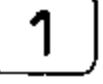





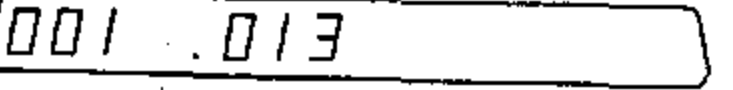



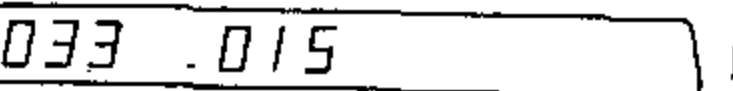

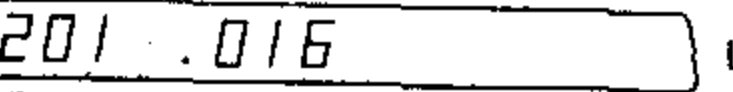



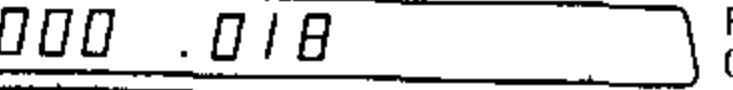


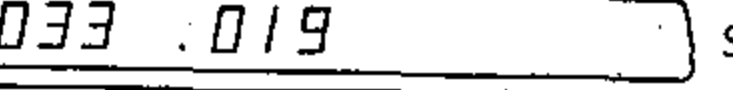



The tape moves forward for approximately 14 seconds before the first block is labeled. After each block is labeled, the block label number is displayed. After block 14 is labeled the tape stops and the display shows:


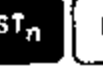





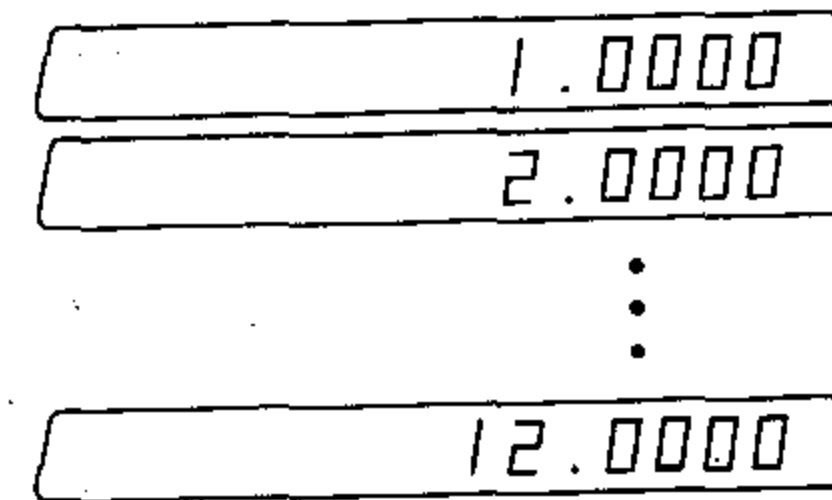
One segment or file of the tape has been preconditioned. Repeat the above process to precondition another file of the tape. The two-sided tape (blue label) can hold a maximum of 12 files on each side of the cassette. When the desired number of files have been preconditioned, rewind the cassette.

It is possible to automatically precondition all 12 files on one side of the two-sided tape cassette using the following procedure:

1. Connect and turn on the 326-392 system.
2. Insert the cassette into the 392 cassette drive with the side that you wish to precondition facing up (Side A or Side B).
3. Press REWIND to fully rewind the cassette.
4. Load the following program into the program memory of the 326.

Do This	See This	Comments
		Start loading at step 001.
 		Precondition tape (non-loop).
  		Increment register 0, leaving result in register 0 and E-register.
		Pause to display number in E-register.
		
 		Subtract number in register 0 from 12.
  		Jump to Label 1 if number in the Entry register equals 0.
  		Jump to step 001.
		Label 1.
  		Recall number in register 0.
 		Stop execution.
		Switch to Run mode.
		

5. Store zero in register 0. (Press   .)
6. Press and latch RECORD and READY.
7. Press .
8. Press . Preconditioning starts. After each segment of 14 blocks is preconditioned, the program pauses to display the segment number;



The program stops after the last segment has been preconditioned with 12.000 in the display as shown.


Block Search

To write a program or data registers onto a specific tape block; or to read a program or data registers from a tape block, you may either search for a specific block to be used or simply use the next block.

- To search for a specific block, enter the block number.
- The entered block number must be an integer (1 through 14), to search for a specific block; or zero to search for the next succeeding block.

Endless Tape (red label):

- The tape moves forward and the system searches until the block is found. As each block is passed, its block number displays. Block 1 follows Block 14.
- If the entered block number is zero, the tape moves forward to the next block.
- If the entered block number is greater than 14, E-----3 displays.

Press  or  to recover from this error condition.

Two-Sided Tape (blue label):

- Usually the cassette should be fully rewound before any Read or Write operations are performed.
- The tape moves forward to search for the entered block number (1 through 14). It may be necessary for the search to go to the next file of 14 blocks.
- If the indicated block does not exist on whatever tape remains, the search goes to the end of the tape. The Write or Read does not occur.
- If the entered block number is greater than 14, a tape error condition E-----3 occurs. Press **RESET** or **CLEAR** for error recovery.*

Using Prepared Programs with the 326

For your convenience, Monroe has an extensive library of prepared program packages. Each package is designed for a particular applications area and programming skills are not required to use the prepared programs. Your Monroe representative can supply you with a complete listing of available program packages.

A typical package has between 10 and 20 programs with detailed instructions and examples for using each program. The actual programs are stored on one or more tape cassettes, which are provided as a part of the package.

To use the packaged programs, the 392 Tape Cassette Drive must be used with the 326 Scientist. The following is the general procedure for using a prepared program:

1. Connect the 392 Tape Cassette Drive to the 326 Scientist.
2. Insert the cassette containing the program to be used.
3. Rewind the cassette by pressing and latching the REWIND key.

Programs are stored on tape cassettes in files and blocks. There are always 14 blocks in a file. The number of files on one side of a tape cassette depends on the applications package being used. Each block holds one program.

*Five different error conditions can occur that are related to the use of the 392 Tape Cassette Drive. (See Appendix F.)

4. Enter the block number (from the keyboard of the 326) of the program to be used (1 through 14).
5. Press and latch the READY key on the 392 Tape Cassette Drive.
6. Press READ FROM TAPE on the tape drive.

The 326-392 system searches for the block indicated. The tape moves forward. As each block is passed, the block number is displayed as a two digit number: 01, 02, ..., etc.

When the block search is complete the program is read from the tape into the memory of the 326. The block number in the display changes from a two digit number to an integer followed by a decimal point and four zeros; for example, 05 becomes 5.0000.


7. Press START to begin program execution.

Refer to the program package operating instructions for the complete description and procedures for using specific prepared programs.

See the following sections for a more complete description and procedure for writing programs on tapes and reading programs from tape.

Writing a Program onto a Block of Tape

To write a program onto a block of tape, follow this general procedure. It is assumed the Tape Cassette Drive is connected and a preconditioned tape cassette has been inserted.



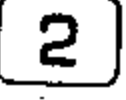




1. Place the PROGRAM/REGISTERS switch in the  position.
2. Press and latch the RECORD and READY keys.
3. Enter the specific block number (or zero to write on the next block in the sequence of blocks). For example, to write a program onto Block 3, enter 3 from the keyboard of the 326.
4. Press WRITE ON TAPE.

The system searches for the block indicated. If the block is found, the full capacity of program memory is written onto the block of tape. Program memory remains unchanged.

Reading a Program from the Tape into Program Memory

Example: Writing a Program onto Tape

Let us write a simple program on a block of tape. Suppose there is a program in the computer's memory that computes the square root of an entered number, X, plus two. The instructions for this program are as follows:

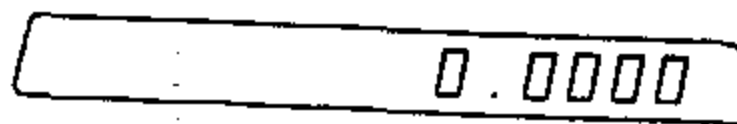
Memory Step	Instruction Code	Key	Comments
001	033		Stop to enter a number, X.
002	021		
003	002		Add 2 to the number X.
004	020		Calculate sum.
005	062		Take square root.
006	350		
007	033		Jump to step 001.

To write the program on tape in block number 3

Do This



See This



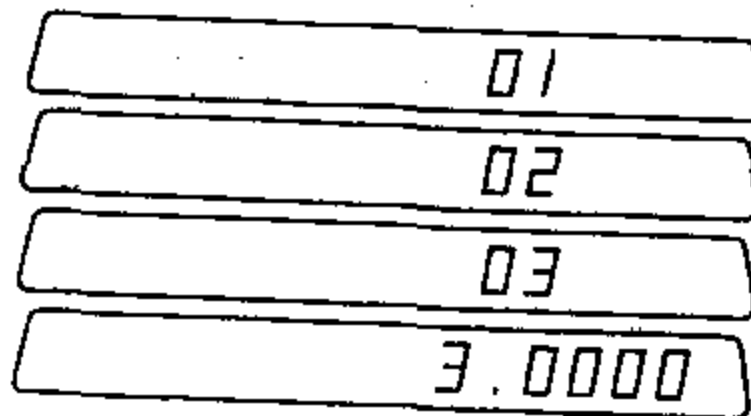
Comments

Specify that program memory is to be written onto tape.

(Press and latch READY and RECORD)

3

(Press WRITE ON TAPE)



Block numbers appear briefly as tape moves forward. Tape stops after having written the program onto block 3.

To read a previously recorded program from a block of tape into the program memory, follow this general procedure. It is assumed that the Tape Cassette Drive is connected and a preconditioned cassette has been inserted.

1. The PROGRAM/REGISTERS switch can be in either position because the kind of information (program or register data) is recorded on the tape during the writing operation.
2. Press and latch the READY key.
3. Enter the block number to be read or zero to indicate the next block. For example, to read the program from block 3 of the tape into program memory, enter 3 from the keyboard of the 326.
4. Press READ FROM TAPE.

The system makes a block search. When the block is found, the program information on the full block of tape is read into the 326. Program memory is completely filled. The information on the block of tape is unchanged.

Example: Reading a Program from Tape

In this example we will read the program previously stored onto block 3 of the tape back into the program memory. First, turn the 326 Scientist off using the ON/OFF switch located behind the right side of the computer. This causes all information in the computer's program memory to be lost. Turn the computer on again. Program memory is now blank. (Turning the 326 off and on is not necessary, of course. It merely serves to make this example more dramatic.)

To read the program from Block 3 on the tape into program memory.

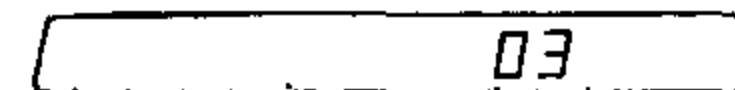
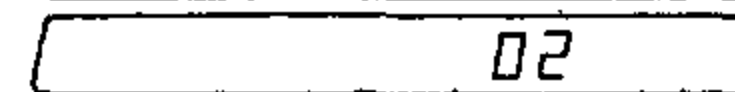
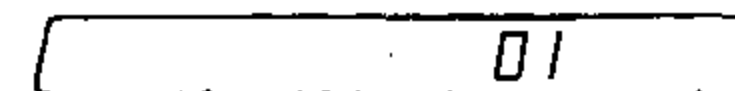
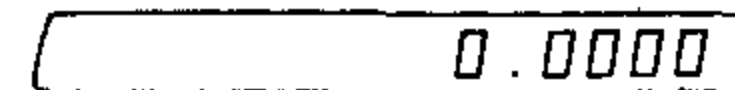
Do This

(Press and latch READY)

3

(Press READ FROM TAPE)

See This



Comments

Enter tape block number where desired program is stored.

A search is made for tape block 03.

The tape cassette stops after block 03 has been located. The desired program is now in memory.

To run the program that calculates $\sqrt{X+2}$, which has been read into program memory.

Do This	See This	Comments
		Jump to step 001 in memory.
		Start execution of program. Program is stopped for entry of X.
		Enter X = 5.
		Calculate value of $\sqrt{5+2} = \sqrt{7}$.
		Enter X = 14.
		Calculate value of $\sqrt{14+2} = \sqrt{16}$.

Writing Data Registers onto Tape

To write from data storage registers onto a tape block, the procedure is as follows. (It is assumed that the Tape Cassette Drive is connected and a preconditioned cassette has been inserted.)

1. Place the PROGRAM/REGISTERS switch in the position.
2. Press and latch the RECORD and READY keys.
3. Enter the block number to be used or zero to indicate the next block (for example, press for Block 4).
4. Press WRITE ON TAPE.
5. Press a numeral or symbol key n (where n is - , , or) to indicate that the contents of registers 0 through n are to be written onto the block (e.g., press to write registers 0 through 6 onto the specified tape block). The system makes a block search. If the block is found, the contents of registers 0 through n are written onto the tape block. Pressing will cause the contents of registers - and to be written on tape. Pressing will cause the contents of all twelve data storage registers to be written on tape.

The contents of the data registers remain unchanged.

Example: Writing from Data Registers onto Tape

Suppose we wish to write the contents of data storage registers 0-3 onto tape Block 1. Assume that the data registers contain the following numbers:

Register 0	5.0000
Register 1	10.0000
Register 2	20.0000
Register 3	35.0000

Rewind the tape and proceed as follows:

Do This	See This	Comments
		Specify that data from storage registers will be written onto tape.
(Press and latch READY and RECORD)		
		Specify tape Block 1.
(Press WRITE ON TAPE)		
		Indicate that data registers 0-3 are to be written on tape Block 1.
		Writing begins.
		The data has been written on tape block 1.

Reading from Tape into the Registers of the 326

To read the previously recorded contents of data registers from a block of tape into the data storage registers of the 326, follow this general procedure. It is assumed that the Tape Cassette Drive is connected and a preconditioned cassette has been inserted.

1. The PROGRAM/REGISTERS switch can be in either position because the kind of information (program or register data) is recorded on the tape block during the writing operation.
2. Press and latch the READY key.

Programmed Tape Control

3. Enter the block number to be read or zero to indicate the next block (for example, press **4** to read from Block 4).

4. Press READ FROM TAPE.

The system makes a block search. When the block is found, the same number of registers that were recorded on the tape block during the Write operation will be read back into the computer's data registers. The information on the block of tape is unchanged. The registers not filled by the READ FROM TAPE are unchanged.

Example: Reading from a Tape Block into Data Registers

To read the information stored in tape Block 01 in the previous example back into data registers 0-3, proceed as follows. (Rewind the tape before beginning this procedure.)

Do This

(Press and latch READY)

1

(Press READ FROM TAPE)

See This

1.0000

Specify tape Block 1.

01

1.0000

The information that was previously written onto tape block 1 from data registers 0-3 has now been read back into data registers 0-3 of the Alpha 325.

5.0000

10.0000

20.0000

35.0000

Recall the numbers stored in registers 0-3.

RCL_n 0

RCL_n 1

RCL_n 2

RCL_n 3

The READ FROM TAPE and WRITE ON TAPE operations can be loaded as program steps. This enables you to program the 392 Tape Cassette Drive for automatic operation. This feature greatly expands the capability of the 326-392 system.

For example, a program may be too large to fit into program memory. With programmed tape control, it can be divided into two or more sections or overlays. The 326 can use one section and then at the appropriate time automatically read the next section into program memory.

When a program block is entered from the tape under program control, execution automatically continues from step 001 after the program Read is complete. It is not necessary to press START.

A variation of the overlay technique uses the logical decision capability of the 326 to choose which particular program segment is to be used next.

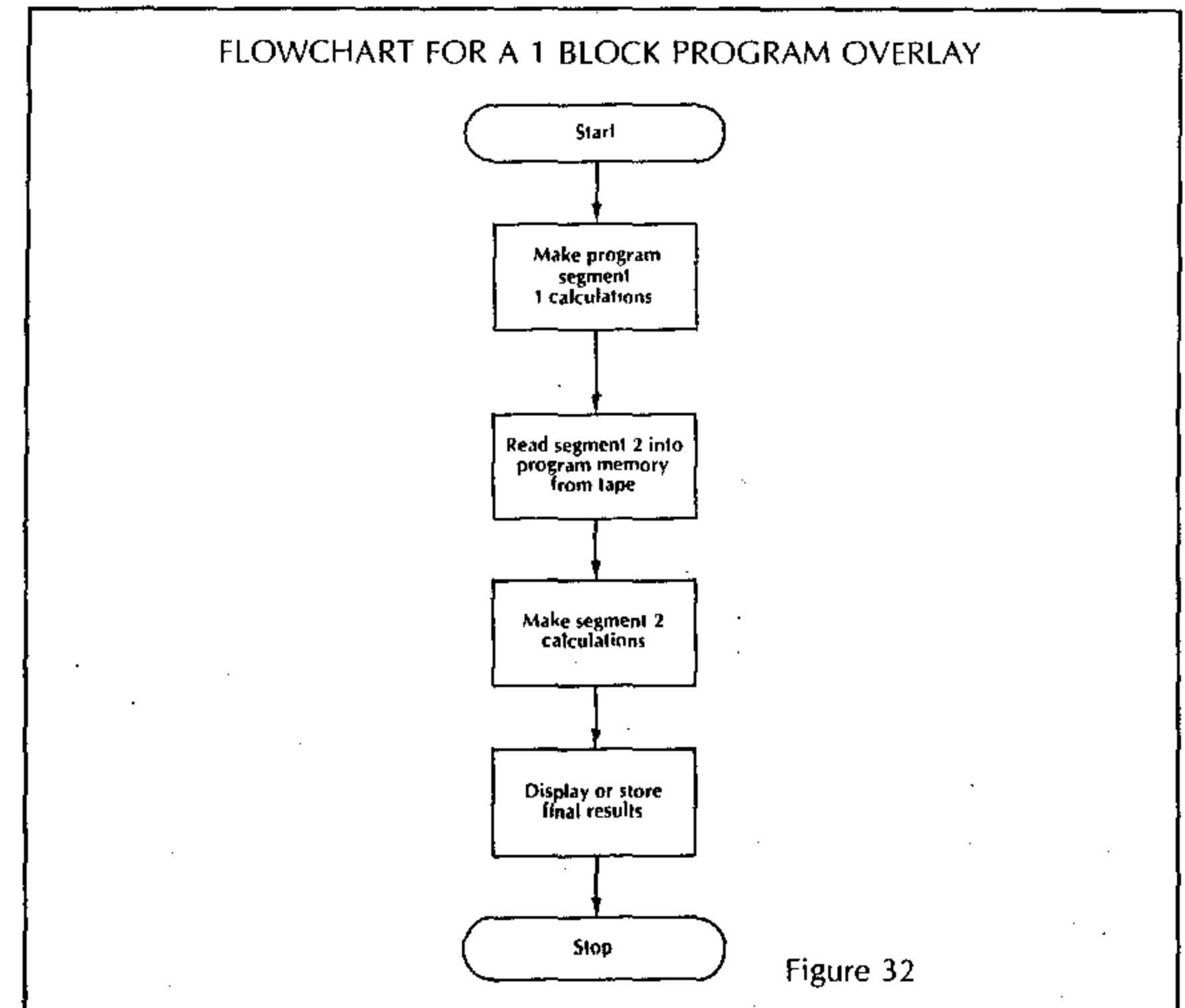


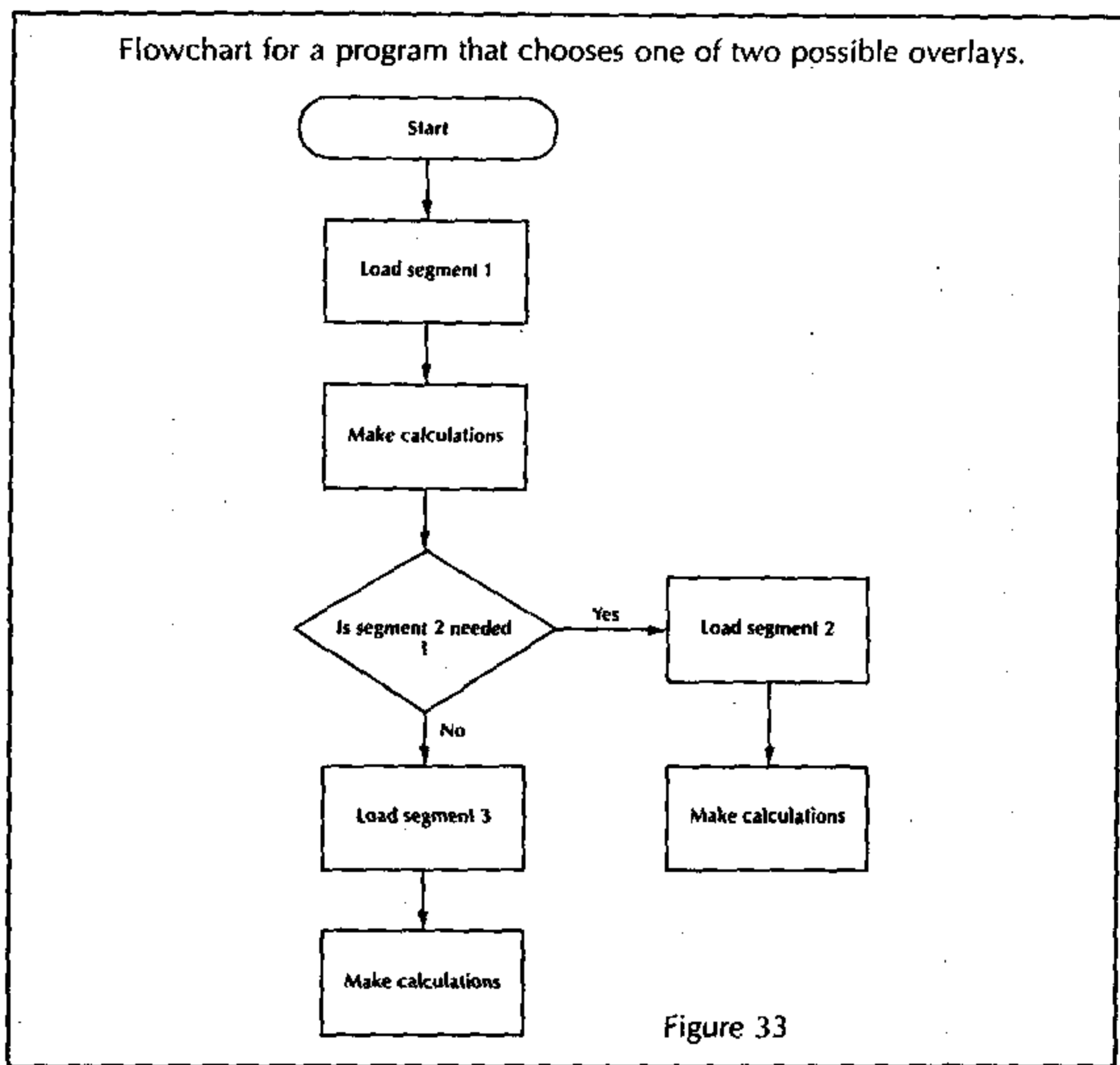
Figure 32

Registers are not altered as new sections of a program are read from the tape. Registers can be used to store the results needed to execute the next section of the program.

Programs requiring more than twelve (12) data registers can automatically store or recall up to twelve (12) registers from a block of tape.

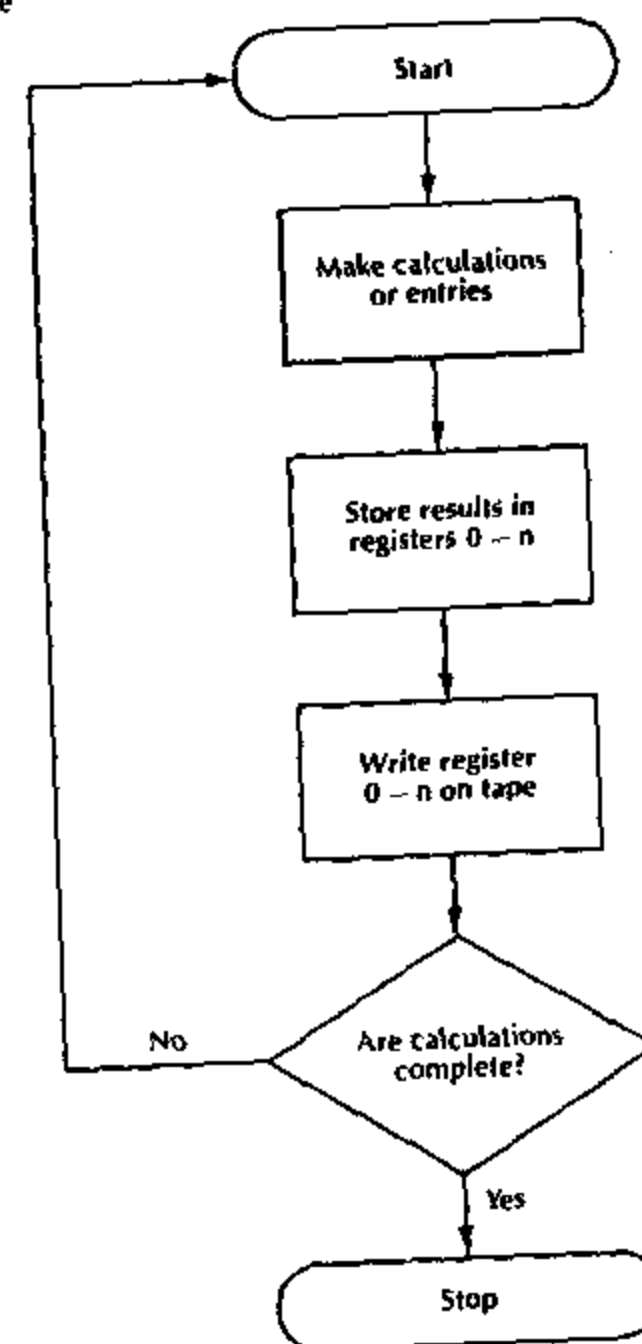
The flowchart in Figure 32 shows the general form of a program that uses a one-block overlay. The flowchart in Figure 33 shows the general form of a program that selects one of two possible overlay blocks, depending on the result of calculations performed.

Figure 34 shows the general flowchart of a program that requires more than 12 data storage registers. Part A shows a repeated writing of the contents of data registers onto different blocks of tape. Part B of the figure shows a successive reading of different sets of data values into the computer's data registers from different tape blocks as needed.



Flowchart for a program that requires more than 12 registers.

A Writing on tape



B Reading from tape

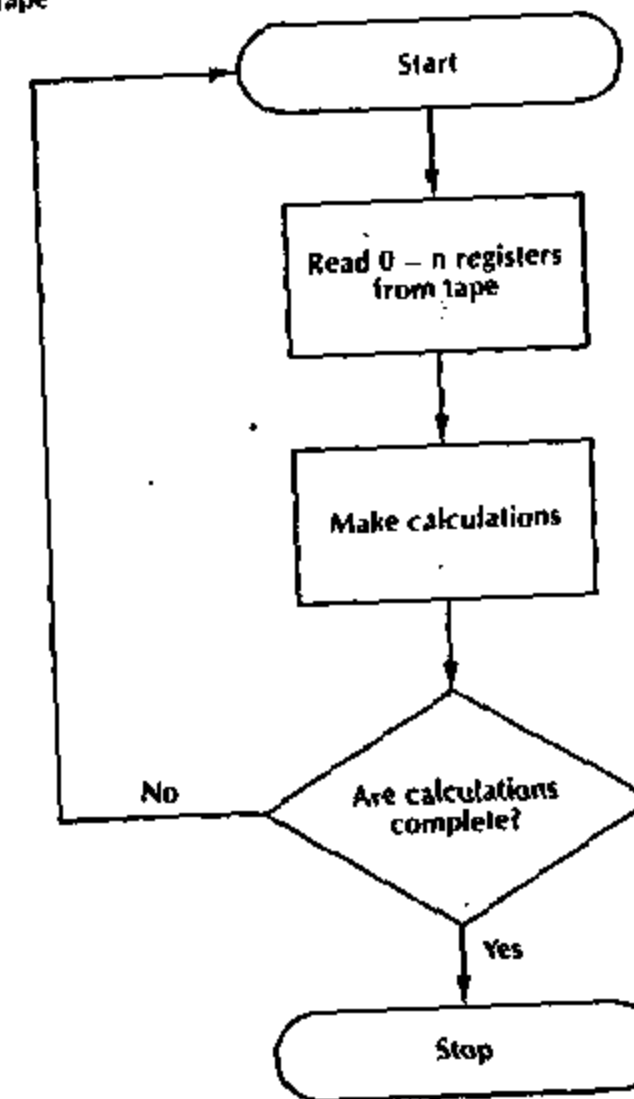




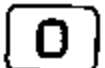


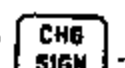


Figure 34

Programmed Tape Control Instructions

Instruction codes that control Tape Cassette Drive writing and reading are loaded into program memory by pressing WRITE ON TAPE and READ FROM TAPE when the RUN/STEP/LOAD switch is set to .

There are three different tape control instructions:

PROGRAM/REGISTERS Switch Setting	Key	Instruction Code
(Switch position has no effect)	READ FROM TAPE	041
	WRITE ON TAPE (writes a program on tape)	040
	WRITE ON TAPE (writes data registers on tape)	340
	The 340 instruction must be followed by a register number	000
	 -  ,  , or 	
	This tells what registers are to be written on tape.	013

Example: Programmed Tape Control for Overlays

This example demonstrates techniques for writing a program that uses overlays. The program shown consists of 3 segments; the main program segment and two overlay segments. Each segment is programmed and written on a block of an Endless Tape Cassette.

The main program (segment 1) tests a keyboard entry. If the entry is greater than or equal to zero, segment 2 is read from the tape and executed. If the entry is less than zero, segment 3 is read from the tape and executed.

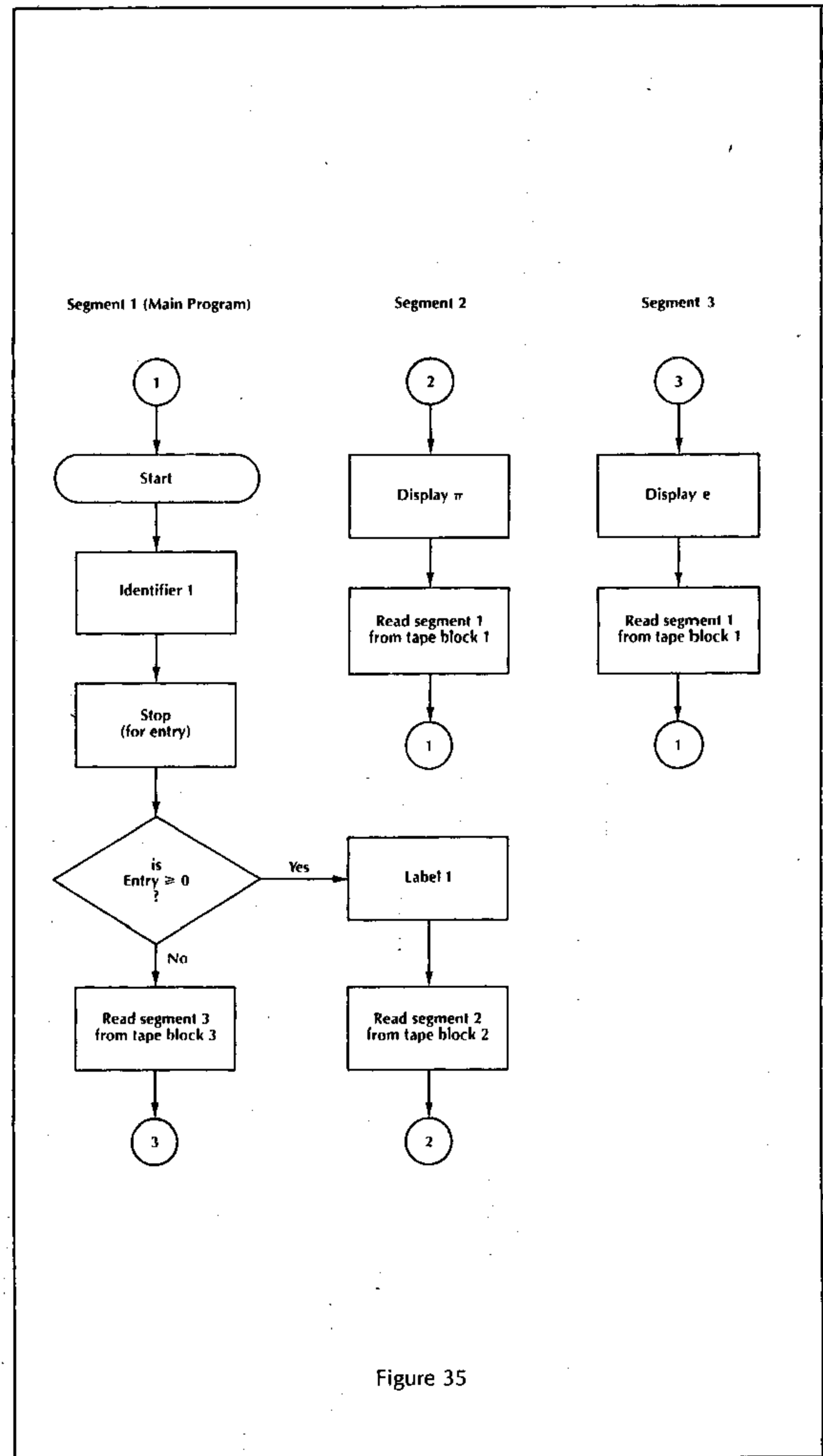


Figure 35

The execution of segment 2 (when the entry ≥ 0) causes the number $\pi (=3.1415)$ to be displayed and then reads segment 1 back into program memory from the Endless Tape.

The execution of segment 3 (when the entry < 0) causes the number $e (=2.7182)$ to be displayed and then reads segment 1 back into memory.

Each segment is programmed and written on a block of endless tape.

- Segment 1 is written on Block 1
- Segment 2 is written on Block 2
- Segment 3 is written on Block 3

To load segment 1 into program memory

Do This	See This	Comments
JUMP START STOP		
RUN STEP LOAD	.001	Segment 1 starts at program memory location 001.
1 / 9	111 .003	Identifier 1.
START STOP	033 .004	Stop for entry.
JUMP + =	355 .005	Jump to label 1 if the entry is ≥ 0 .
1	001 .006	
CLEAR	037 .007	This CLEAR separates the number previously in the E-register from the 3.
3	002 .008	
READ FROM TAPE	041 .009	Read Block 3 from the tape (if the entry is < 0).
/ LABEL 1	201 .010	Label 1.
CLEAR	037 .011	This CLEAR separates the number previously in the E-register from the 2.
2	002 .012	
READ FROM TAPE	041 .013	Read Block 2 from tape (if the entry is ≥ 0).
RUN STEP LOAD	0 .0000	The number in the Entry register prior to LOAD is displayed.



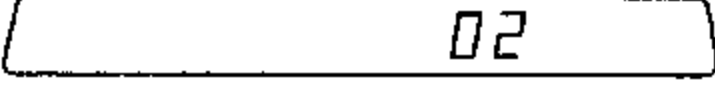
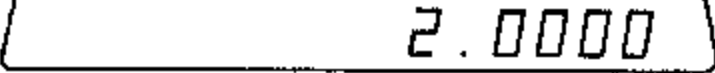
To write segment 1 onto block 1

Do This	See This	Comments
PROGRAM REGISTERS		Specify writing from program memory onto tape.
(Press and latch RECORD and READY)		
1	1.0000	Block 1.
WRITE ON TAPE		Write contents of program memory onto tape Block 1.
	01	
	1.0000	Then system searches for Block 1 and writes the full capacity of program memory onto tape.




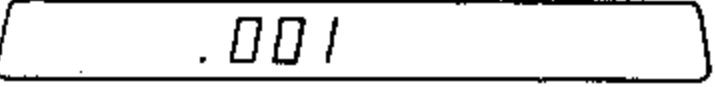
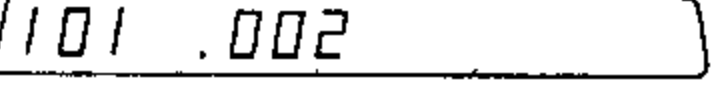

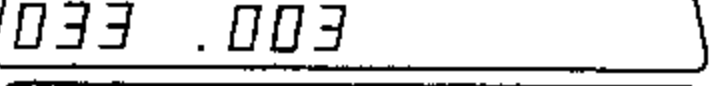
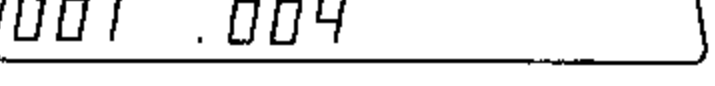
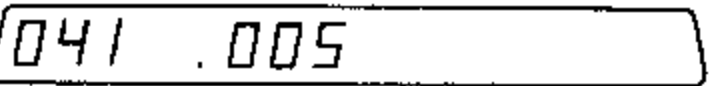

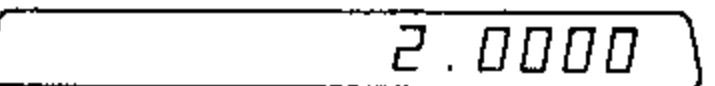
To load segment 2 into program memory

Do This	See This	Comments
JUMP START STOP		
RUN STEP LOAD	.001	Jump to step 001.
/ 7	107 .002	Start loading at step 001.
START STOP	033 .003	Display π .
1	001 .004	
READ FROM TAPE	041 .005	Read Block 1 from the tape.
RUN STEP LOAD	1.0000	The number in the Entry register prior to LOAD is displayed.





To write segment 2 onto block 2

Do This	See This	Comments
 (Press and latch RECORD and READY)		Specify writing from program memory onto tape.
2		Block 2.
WRITE ON TAPE		Write contents of program memory onto tape Block 2.
		

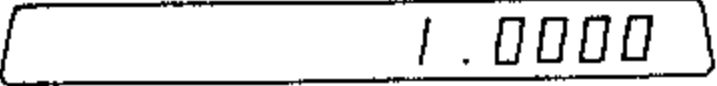


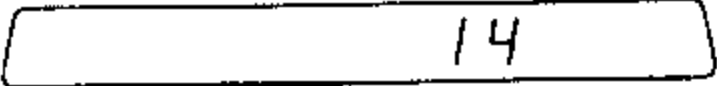
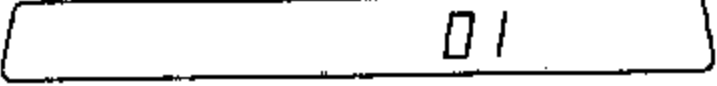



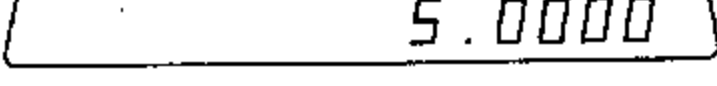




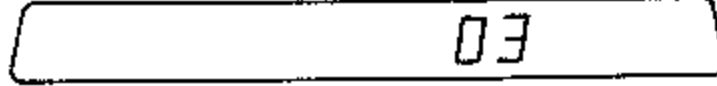


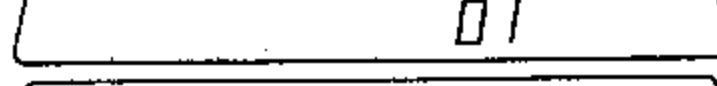
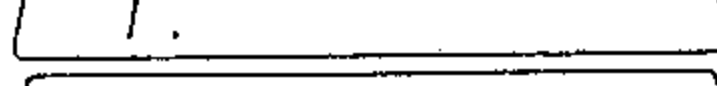


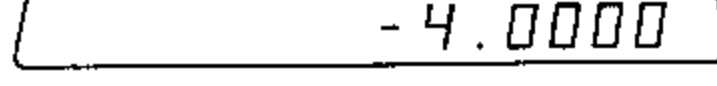


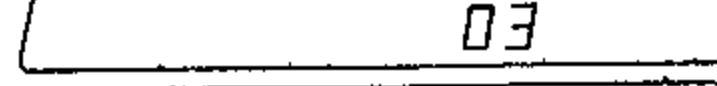

To load segment 3 into program memory

Do This	See This	Comments
 		Jump to step 001.
		Start loading at step 001.
1		Display e
		
1		Read Block 1 from the tape.
READ FROM TAPE		
		The number in the Entry register prior to LOAD is displayed.

To write segment 3 onto block 3

Do This	See This	Comments
 (Press and latch RECORD and READY)		Specify writing from program memory onto tape.
3		Block 3.
WRITE ON TAPE		Write contents of program memory onto tape Block 3.
		

To run the program with values of 5 and -4 entered from the keyboard.

Do This	See This	Comments
(Press and latch RECORD and READY)		
1		Specify block 1 on tape.
READ FROM TAPE		
		
		Read Block 1 from tape into program memory.
		
		
		Start execution.
		Identifier 1 displays.
5		Enter 5 as the number to be tested.
		Start execution.
		Since the number entered is greater than zero, block 2 is automatically read into program memory causing the value of π to be displayed.
		
		
		
		
		Tape Block 1 is read into program memory again.
		
		Identifier displays.
 		Enter -4 as the number to be tested.
		
		Since the number entered is less than zero, tape Block 3 is read into program memory causing the value of e to be displayed.
		
		

Note that when these program segments are read into program memory under program control, execution begins automatically.

Example: Programmed Tape Control for Writing from Data Registers onto Tape

The following example shows how data registers are automatically stored onto tape. It also provides an opportunity to use many of the programming techniques already described in this book.

Suppose that you want to solve the same equation or problem many times and then store the solutions so that they can be studied at your convenience. It is possible to write a program that automatically changes the value of the variable, solves the equation and stores the result into a data register. After a certain number of registers have been used, the program automatically writes the contents of the data registers onto a block of tape and continues the program, solving the equation for other values of the variable and storing additional results onto a tape block.

Applications in which such storage of data registers may be used include the following:

- Programs that have long execution times and many final results. This can include the solution of problems by numerical integration or iterative techniques.
- Programs that require the manipulation of large amounts of data, such as the solution of statistical problems.

For example, let us solve the equation for the area of a circle, $A = \pi r^2$, for values of the radius, r , over the range $1 \leq r \leq 9$ where r is increased by 0.5 each time a new A is to be calculated.

Store the calculated values of A in registers 0 through 9. When 10 registers have been used, write from the registers onto a block of tape. Then clear the data storage registers and continue executing the program. When the final calculation for A has been made, store all ten registers onto another block of tape and stop.

The program flow on the next page shows a convenient way of solving the problem. The main program stores the calculations for A into the correct registers and writes 10 registers onto a block of tape as soon as they are filled. A subroutine calculates A , increments the value of r , and tests to see when the last calculation has been made.

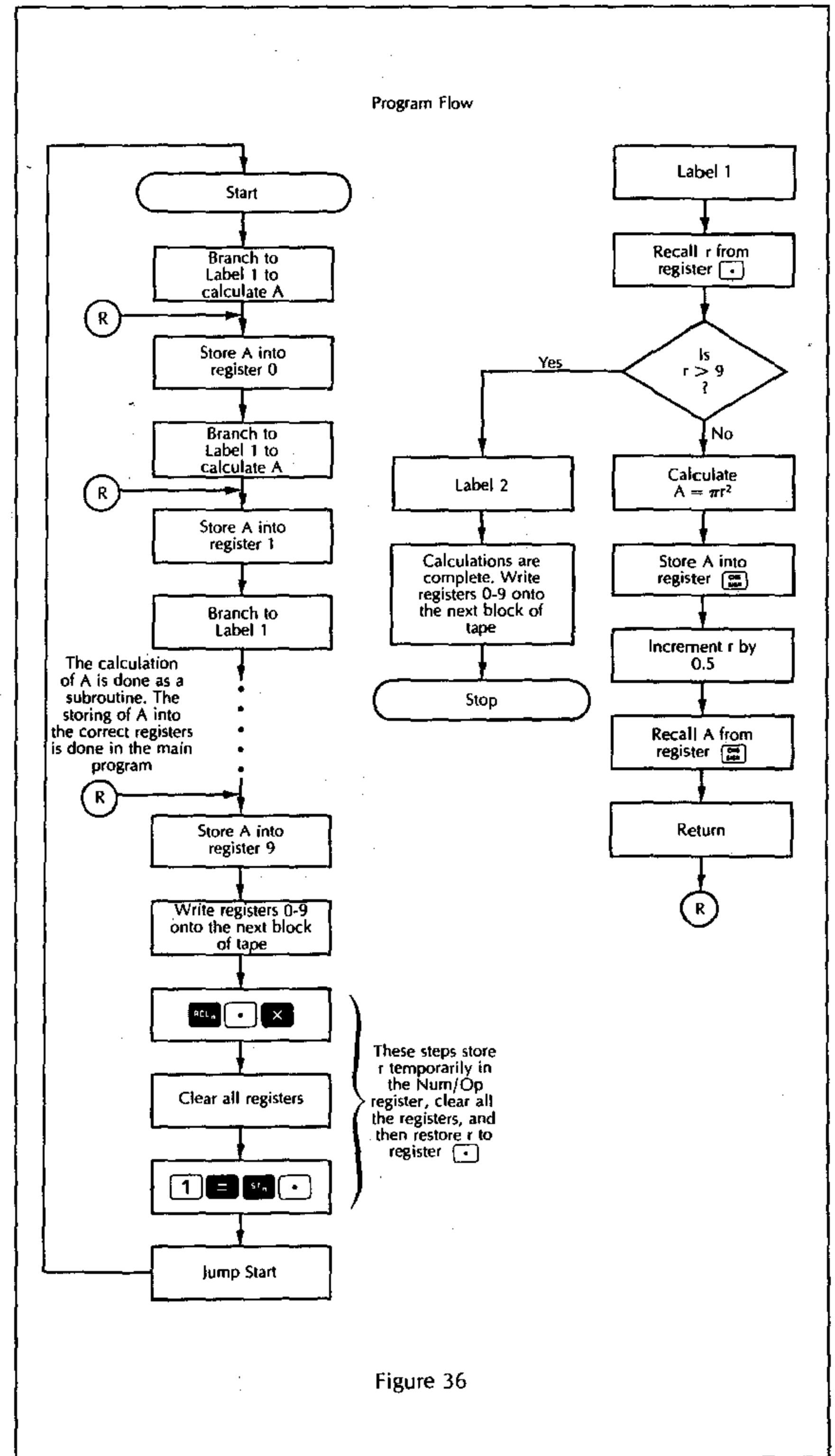


Figure 36

To load the main program

Do This	See This	Comments
		Start loading at step 001.
		Branch to subroutine labeled 1.
		Store calculated value of A in register 0.
		Branch to subroutine.
		Store A in register 1.
		Branch to subroutine.
		Store A in register 2.
		Branch to subroutine.
		Store A in register 3.
		Branch to subroutine.
		Store A in register 4.
		Branch to subroutine.
		Store A in register 5.
		Branch to subroutine.
		Store A in register 6.
		Branch to subroutine.
		Store A in register 7.
		Branch to subroutine.
		Store A in register 8.
		Branch to subroutine.
		Store A in register 9.
		Clear Entry places a zero in the E-register.
		Write registers 0-9 onto the next block of tape. (The Clear is used to designate the next tape block instead of zero. If zero were used, a zero digit would be added to the value of A in the display.)

Do This	See This	Comments
	 	This places the current value for r into the Number/Operator register so that it won't be affected by the following Clear All Registers instruction.
		This clears all the data registers. This is done so that when data registers are later read from the tape, you will know which register was the last one used. The last A value in a register will be followed by a register containing a zero.
	 	This brings r back to the Entry register from the Num/Op register. R is then stored again in register .
		Jump to step 001 to continue calculations.

To load the subroutine

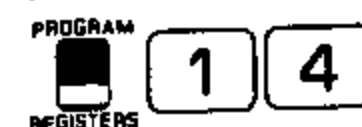
Do This	See This	Comments
LABEL 1	201 .055	Label 1 (subroutine).
RCL _n .	012 .057	Recall r from register .
- 9	011 .059	Subtract 9 from r.
JUMP + 2	002 .062	Jump to label 2 if r is greater than 9 (last A has been calculated).
RCL _n . RETURN	162 .065	Program continues in sequence if $r \leq 9$. Calculate r^2 .
X LABEL 7 ST _n =	020 .068	Calculate $A = \pi r^2$.
ST _n CHG SIGN	013 .070	Store A into register CHG SIGN.
. 5	005 .072	Increment r (in register .) by 0.5.
ST _n + .	012 .074	
RCL _n CHG SIGN	013 .076	Recall A.
RETURN	030 .077	Return to the main program.
LABEL 2	202 .078	Label 2.
WRITE ON TAPE	000 .079	Write registers 0-9 on the next block of tape.
LABEL 0	340 .080	
LABEL 9	011 .081	
START STOP	033 .082	Stop executing. The last calculation of A has been made.
RUN STEP LOAD	0.0000	Switch to Run mode in preparation for running program.

To run the program

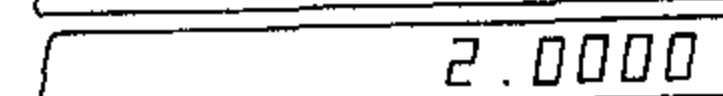
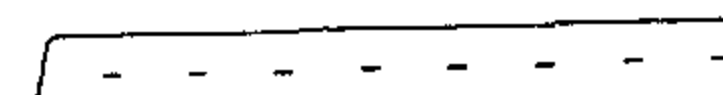
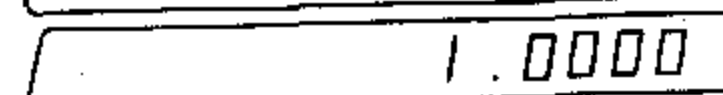
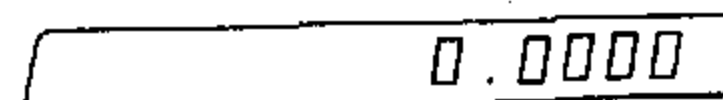
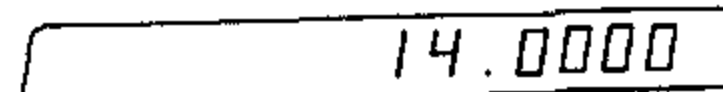
Assume that the system is connected and a preconditioned tape is inserted. Endless or two-sided tape may be used. If endless tape is used, the number of calculations that can be recorded is limited to 12 X (the number of registers used in each block). In this case 10 registers are written onto each block of tape.

Do This

(Press and latch RECORD and READY)



See This



Comments

This positions the tape so that the data registers will be written on Block 1 first. Later when data is read from the tape you will start reading from Block 1. To move the tape to block 14 we have specified writing register 0 onto block 14.

Clear all registers.

Store the initial r value into register (R = 1.0000).

Start the program.

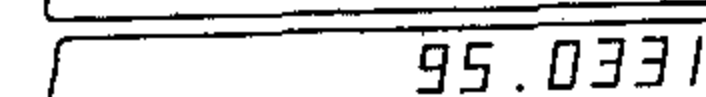
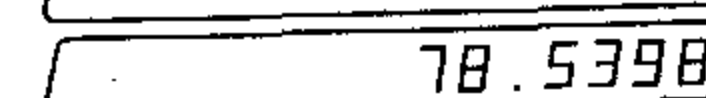
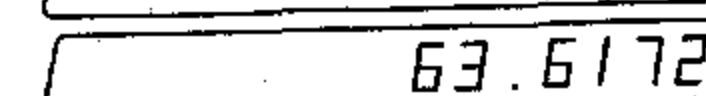
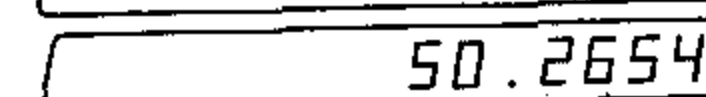
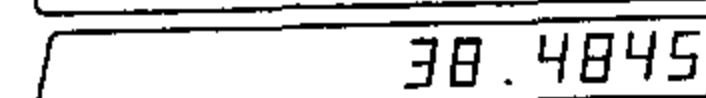
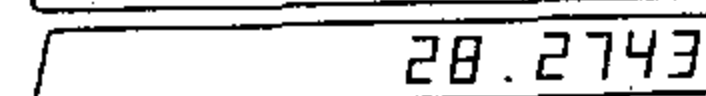
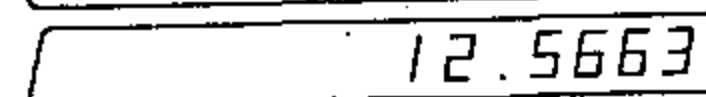
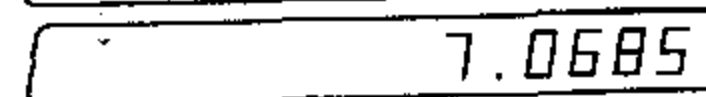
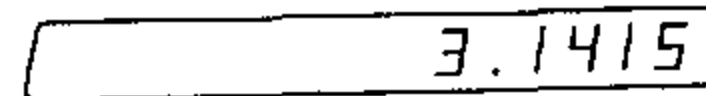
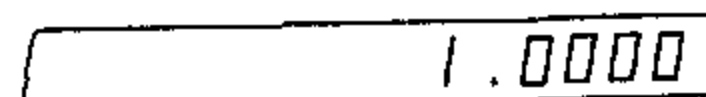
The program executes until 2.0000 displays. (Registers are stored on block 02 last.)

To see the results of the calculations

Do This



See This



Comments

The system searches for block 1 and then reads the data into the registers of the 326.

Data registers on Block 1 have been read into the data registers of the 326.

Values of A for $r = 1$

$r = 1.5$

$r = 2$

$r = 2.5$

$r = 3$

$r = 3.5$


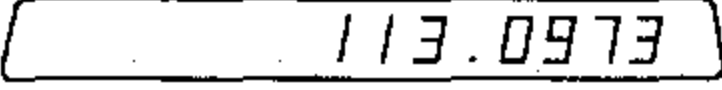
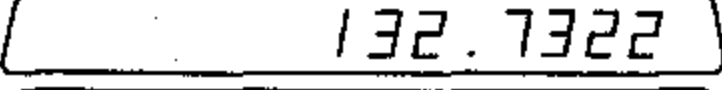
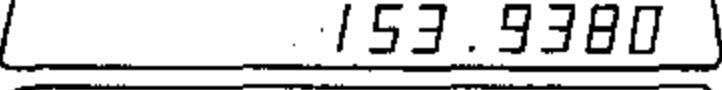
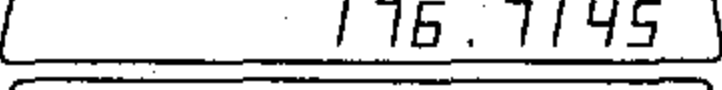

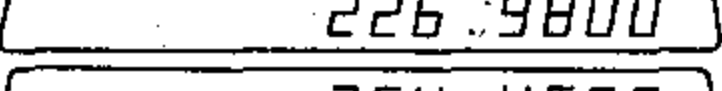
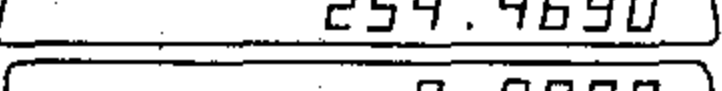
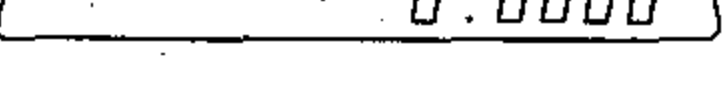
$r = 4$

$r = 4.5$

$r = 5$

$r = 5.5$



Do This	See This	Comments
2 READ FROM TAPE		The contents of the data registers that were written on Block 2 are read into the data registers of the 326.
RCL_n 0		Values of A for $r = 6$
RCL_n 1		$r = 6.5$
RCL_n 2		$r = 7$
RCL_n 3		$r = 7.5$
RCL_n 4		$r = 8$
RCL_n 5		$r = 8.5$
RCL_n 6		$r = 9$
RCL_n 7		Last A calculation is in register 6.

The procedure for displaying the results of the program can also be made into a program that is completely automatic. You might try this as an exercise.

Hints:

1. Start reading from block 1 and then use the "read from next block" instruction sequence.
2. Recall successive registers.
3. A display of r can precede the display of the A calculation. Use the Pause instruction (**PAUSE**) for displays.
4. Increment and test the value of r . Stop the program when $r > 9$, or you can test for the first register that contains zero.

Appendix A: Accuracy and Number Sizes

All numbers used in the 326 Scientist have 13 significant digits and a two digit exponent. This allows the range of numbers to go from 10^{-98} to $9.99999999999 \times 10^{99}$.

Add and subtract are accurate to the full 13 digits. Multiply and divide are rounded in the 13th digit to gain greater accuracy (e.g., $2 \div 3 = 0.6666666666667$). The functions a^x , to Rectangular, to Polar, Sin, Sin^{-1} , Cos, Cos^{-1} , Tan, Tan^{-1} , Ln, e^x , log, 10^x are almost always accurate to one part in the 12th digit and never less than one part in the 11th digit.

Square root is accurate to 2 parts in the 13th digit. Radians to Degrees and Degrees to Radians are both accurate to 2 parts in the 13th digit.

$1/x$ is simply a divide and has the same accuracy as divide.

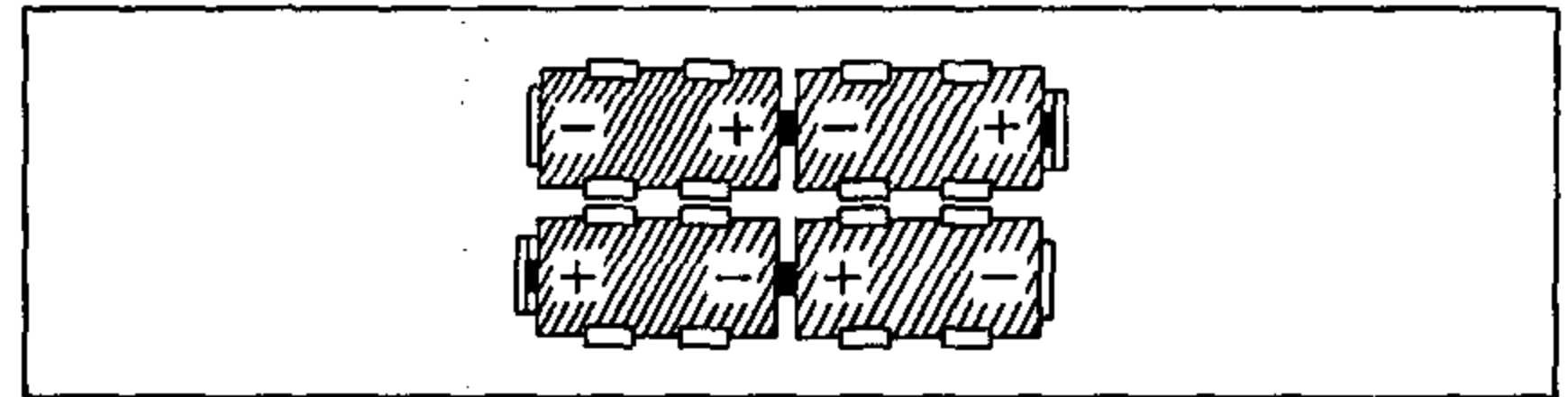
The accuracy of the Scientist most probably exceeds the accuracy you need in most cases but it is there when you require it.

Appendix B: Batteries and Recharging

The 326 Scientist may operate from the AC voltage adapter supplied with the 326 or from the batteries when operation from the AC line is inconvenient.

The batteries supplied with the 326 may be discharged and recharged approximately 1000 times during their useful life. When fully discharged, the batteries may be recharged in 16 to 20 hours by connecting the computer to the AC adapter and the adapter to the AC line. Battery recharging occurs regardless of the position of the ON/OFF switch and does not interfere with calculations.

The AC adapter supplies 300 to 350 milliamperes of charging current to the batteries. For this reason only Monroe part number CD3400017 or similar nickel-cadmium batteries rated for at least three ampere-hours should be used in the 326. If you do need to use other types in an emergency, DO NOT connect the AC adapter. Attempting to recharge a rechargeable battery rated at less than 3 ampere-hours or a dry cell may burst the batteries and damage the computer. Such damage is NOT covered by warranty or equipment service agreement.





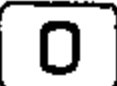
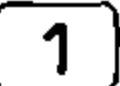
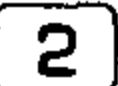
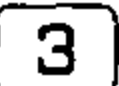
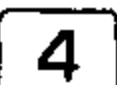

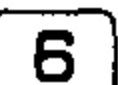
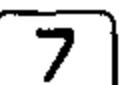
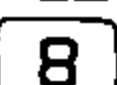
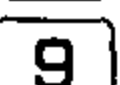






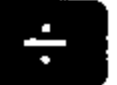










To replace the batteries, turn the 326 over, squeeze together the two round plungers at the top of the case, and pull the door open. Remove the batteries and replace with fresh ones, observing the polarity shown in the illustration.



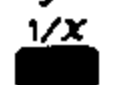


















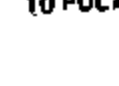

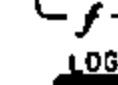





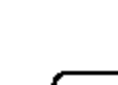
All nickel-cadmium batteries exhibit an apparent reduction of capacity after a number of cycles of partial discharging and recharging. You may easily correct this by running the computer on batteries until the display goes completely blank, and then recharging the batteries for 16 to 20 hours. Care must be taken not to continue the discharge period beyond the point when the displays goes blank. You may use the computer during the recharge period.

The 326 concentrates many thousands of logic circuits into a small volume, so the metal label on the bottom may feel noticeably warm to the touch. This temperature is normal and should cause no alarm provided it does not exceed 49°C (120°F).

Appendix C: Summary of 326 and 392 Keys and Switches

326 Scientist Keys and Switches

Key (or Switch Setting)	Description	Reference Page Number
ON	Turns the 326 Scientist on	9
OFF	Turns the 326 Scientist off	9
	Clear the Entry register	10
	Reset the Number/Operator register and clear the Entry register	17
   	Numeral entry	9
   		
 		
		
	Change sign	10
	Exponent	12
	Add	18
	Subtract	19
	Multiply	19
	Divide	20
	Equals	17
 	Parentheses	22
	Store into a data storage register	36
	Recall from a data storage register	37
	Function key	51
	Degrees, minutes, seconds	30
 	Convert from a D/M/S angle to an angle in degrees with decimal fraction or from degrees with decimal fraction to D/M/S	57
	Square root	52

Key (or Switch Setting)	Description	Reference Page Number
 	Square	52
	Reciprocal	51
 	Factorial	51
	Grads	56
	Degrees	56
	Sine	58
 	Arc sine	58
	Cosine	59
 	Arc cosine	59
	Tangent	59
 	Arc tangent	60
	Polar to rectangular conversion	61
 	Rectangular to polar conversion	62
	Logarithm to the base e	53
 	Antilogarithm to the base e (exponential function)	53
	Logarithm to the base 10	52
 	Antilogarithm to the base 10	53
	Statistical summation; add 1, X and X ² into registers 1, 2 and 3, respectively	63
 	Delete from statistical summation; subtract 1, X and X ² from registers 1, 2 and 3, respectively	65
	English to Metric Conversions	68
	Metric to English Conversions	68

Key or Switch Setting	Description	Reference Page Number
0	Clear registers 1, 2, 3	63
1	e	70
2	Absolute value	70
3	Mean	66
4	Standard deviation	67
5	Fraction	71
6	Integer	71
7	π	71
8	Round	72
9	Identifier	72
CLEAR	Clear all registers	73
EXP	Pause	72
JUMP	Jump to Label	87
BRANCH	Branch to a subroutine	105
INSERT	Insert a program instruction code	100
REMOVE	Remove a program instruction code	100
FWD	Go forward to the next program step	100
BACK	Go back to the previous program step	100
LABEL	Names a particular program step	93
RETURN	Return to the main program from a subroutine	106
START STOP	Start or Stop program execution	86
RUN STEP LOAD	Make keyboard calculations or execute program	85, 87
RUN STEP LOAD	Execute a program one step at a time	85, 123

Key (or Switch Setting)	Description	Reference Page Number
RUN STEP LOAD PROGRAM	Load program instructions into memory	85, 87
REGISTERS PROGRAM	A program from the computer's memory is to be written onto a tape block	140
REGISTERS	Data from the data storage registers is to be written onto a tape block	140
a^x	Raise a to the power x	24, 28
SET D.P.	Set decimal point	11

392 Tape Cassette Drive Controls

Key or Switch Setting	Description	Reference Page Number
EJECT	Opens the door of the Tape Cassette Drive and ejects the cassette	139
RECORD	Record (i.e., Write) programs or data on tape	139
REWIND	Rewind a non-loop tape	139
READY	Place the Read/Write head against the tape	139
FORWARD	Move non-loop tape rapidly forward (fast forward)	139
STOP	Stops all tape operations	140
WRITE ON TAPE	Write a program or data onto tape	140
READ FROM TAPE	Read a program or data from tape	140
WRITE ON TAPE	Precondition an unblocked tape cassette	141

Appendix D: Instruction Codes for the 326 Scientist

000	0	023	x	071	COSINE	173	TO POLAR	304	ST ÷
001	1	024	÷	072	TANGENT	200	↓	305	ST a^x
002	2	025	a^x	073	TO RECT.	213	↓	310	RCL
003	3	026	(100	↓	220	↓	311	RCL +
004	4	027)	114	↓	234	↓	312	RCL -
005	5	030	RETURN	115	↓	240	↓	313	RCL x
006	6	033	START/STOP	140	↓	253	↓	314	RCL ÷
007	7	036	RESET	137	↓	260	↓	315	RCL a^x
010	8	037	CLEAR	140	↓	273	↓	320	EXCH
011	9	040	WRITE PRG.	150	↓	300	↓	321	EXCH +
012	*	041	READ TAPE	160	↓	301	↓	322	EXCH -
013	CHG SIGN	050	$\Sigma n \cdot x^2$	161	↓	302	↓	323	EXCH x
014	EXP	060	ln	162	↓	303	↓	324	EXCH ÷
015	D/M/S	061	LOG	163	↓	300	↓	325	EXCH a^x
020	=	062	$\sqrt{\quad}$	170	ARC SINE	301	ST +	340	WRITE DATA
021	+	063	1/X	171	ARC COSINE	302	ST -	350	JUMP
022	-	070	SINE	172	ARC TANGENT	303	ST x	351	JUMP +

← METRIC AND →		CODE		FUNCTION		CODE
		TO METRIC	FROM METRIC			
0	OF → °C	240	280	0	CLEAR REGS 1,2,3	100
1	INCH → CM	241	281	1	DISPLAY e	101
2	FOOT → METRE	242	282	2	ABSOLUTE VALUE	102
3	MILE → KILOMETRE	243	283	3	MEAN	103
4	IN ² → CM ²	244	284	4	STD. DEV.	104
5	US GAL → LITRE	245	285	5	FRACTION	105
6	UK GAL → LITRE	246	286	6	INTEGER	106
7	POUND → KILOGRAM	247	287	7	DISPLAY π	107
8	OUNCE → GRAM	250	278	8	ROUND DISPLAY	110
9	LB/FT ³ → GM/CM ³	251	271	9	IDENTIFIER	111
*	LB/IN ² → KG/CM ²	252	272	CHG SIGN	AUTO TEST	113
				EXP	PAUSE	114
				CLEAR	CLEAR ALL REGS	137

326 SCIENTIST


Appendix E: 326 Scientist Error Conditions



Operations which cause ERROR are:

- Divide by zero.
- 1/X of zero.
- $\sqrt{\quad}$ of a negative number.
- Log or Ln of zero or negative number.
- Arc sine or arc cosine of a number greater than 1 or less than -1.
- Calculating 0^{-x} with the a^x key.
- Calculating 0^0 with the a^x key.
- Factorial of numbers less than zero.
- Factorial of numbers greater than 69.
- Factorial of non-integer.
- Attempting to generate a number outside the range of the computer (10^{-98} to $9.999999999999 \times 10^{99}$).
- Exceeding the parentheses level limit.
- Close without corresponding open when using parentheses.
- Mean with $n = 0$.
- Standard deviation with $n \leq 1$.

Press **REKEY** or **CLEAR** to restore operation after 326 error conditions.





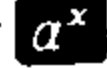
Appendix F: 392 Tape Cassette Drive Error Conditions

Display	Error
E-----1	READY not latched at the time of the execution of READ FROM TAPE, or WRITE ON TAPE.
E-----2	RECORD not latched at the time of manual or programmed execution of WRITE ON TAPE.
E-----3	Invalid preconditioning sequence. No entry or an entry other than 0 or 1 before the sequence  WRITE ON TAPE. or: Block requested not 0 or outside the range 1-14.
E-----4	An attempted read of a block that has neither program nor data registers written on it.
E-----5	The information cannot be read. Either the tape contains a flaw or the stored program or data has been altered.

Press  or  to restore operation after 392 Tape Cassette Drive error conditions.

Appendix G: Parentheses and the Number/Operator Register


In addition to the Number/Operator register (Num/Op register) there are four other internal registers. These registers are used when working with the parentheses keys. For convenience, these registers are referred to as Paren 1, Paren 2, Paren 3 and Paren 4.

The contents of the Num/Op register can only be changed by pressing an arithmetic operator key (    or ). The designated operator and the current contents of the E-register are placed in the Num/Op register for use in calculation.

The parenthesis registers operate in conjunction with the Num/Op register, but are only affected when parentheses are used; and then only according to a fixed hierarchy.

The first time a left parenthesis is used, the contents of the Num/Op register is duplicated in Paren 1. Paren 1 does not change again until either another left parenthesis or a right parenthesis is used. A second left parenthesis duplicates the contents of Paren 1 into Paren 2 and duplicates the contents of Num/Op in Paren 1. A similar duplication into Paren 3 and Paren 4 takes place when a 3rd and 4th left parenthesis is used.

When right parentheses are used the order of duplication is reversed. A right parenthesis duplicates the contents of Paren 1 in Num/Op; Paren 2 is duplicated in Paren 1; Paren 3 is duplicated in Paren 2 and Paren 4 is duplicated in Paren 3.

Arithmetic operations that change the contents of the Num/Op register have no effect on the Paren registers. Pressing  clears the Num/Op and the Paren registers.

Example:

Using two levels of parentheses

The following table shows what happens to the internal registers for the calculation $5 \times [(6/2) + (7/2)] =$

Calculation	Key	Paren Counter	Display	Num/Op	Paren 1	Paren 2	Paren 3	Paren 4
	5		5.0000	Blank	Blank	Blank	Blank	Blank
	x		5.0000	5x	-	-	-	-
	(1	1	5x	5x	-	-	-
	(2	2	5x	5x	5x	-	-
	6	2	2	5x	5x	5x	-	-
	÷	2	2	6÷	5x	5x	-	-
	2	2	2	6÷	5x	5x	-	-
6 ÷ 2 =)	1	1	5x	5x	-	-	-
	+	1	1	3+	5x	-	-	-
	(2	2	3+	3+	5x	-	-
	7	2	2	3+	3+	5x	-	-
	÷	2	2	7÷	3+	5x	-	-
	2	2	2	7÷	3+	5x	-	-
7 ÷ 2 =)	1	1	3+	5x	-	-	-
3 + 3.5 =)		6.5000	5x	-	-	-	-
5 × 6.5 =	=		32.5000	5x	-	-	-	-

Note: A dash (-) signifies that the register is blank.

Example:

Using four levels of parentheses

This example shows the changes in the internal registers for the calculation:

$$3 \times [0.8^{(0.7 \times (5 + (2 \times 6)))}] =$$

Calculation	Key	Paren Counter	Display	Num/Op	Paren 1	Paren 2	Paren 3	Paren 4
	3		3.0000	Blank	Blank	Blank	Blank	Blank
	x		3.0000	3x	-	-	-	-
	(1	1	3x	3x	-	-	-
	· 8	1	1	3x	3x	-	-	-
	a ^x	1	1	0.8a ^x	3x	-	-	-
	(2	2	0.8a ^x	0.8a ^x	3x	-	-
	· 7	2	2	0.8a ^x	0.8a ^x	3x	-	-
	x	2	2	0.7x	0.8a ^x	3x	-	-
	(3	3	0.7x	0.7x	0.8a ^x	3x	-
	5	3	3	0.7x	0.7x	0.8a ^x	3x	-
	+	3	3	5+	0.7x	0.8a ^x	3x	-
	(4	4	5+	5+	0.7x	0.8a ^x	3x
	2	4	4	5+	5+	0.7x	0.8a ^x	3x
	x	4	4	2x	5+	0.7x	0.8a ^x	3x
2 × 6 =)	3	3	5+	0.7x	0.8a ^x	3x	-
5 + 12 =)	2	2	0.7x	0.8a ^x	3x	-	-
0.7 × 17 =)	1	1	0.8a ^x	3x	-	-	-
0.8 ^(11.9) =)		0.0702	3x	-	-	-	-
3 × 0.0702	=		0.2108	3x	-	-	-	-

Note: A dash (-) signifies that the register is blank.

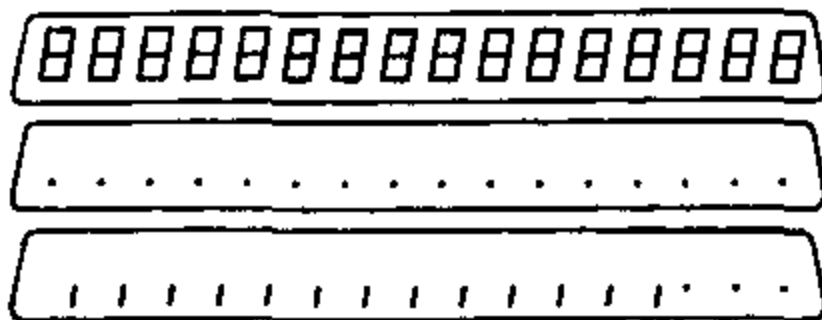
Appendix H: Auto Test Routine

The 326 has a built-in self testing routine. This routine tests the operation of the display, the 40,000 bits of Read Only Memory and the 4,000 bits of Random Access Memory. If you think the 326 is not operating correctly, use the self test routine. If the self test does not operate as described, the 326 may not operate correctly.

To use the self test routine, press **7** **VX** **CMO** **SIGN**. The auto test procedure destroys any programs or register data that are in the 326. The self test routine causes the display to go through a sequence of changes as described below.

Display Test

See This



Explanation

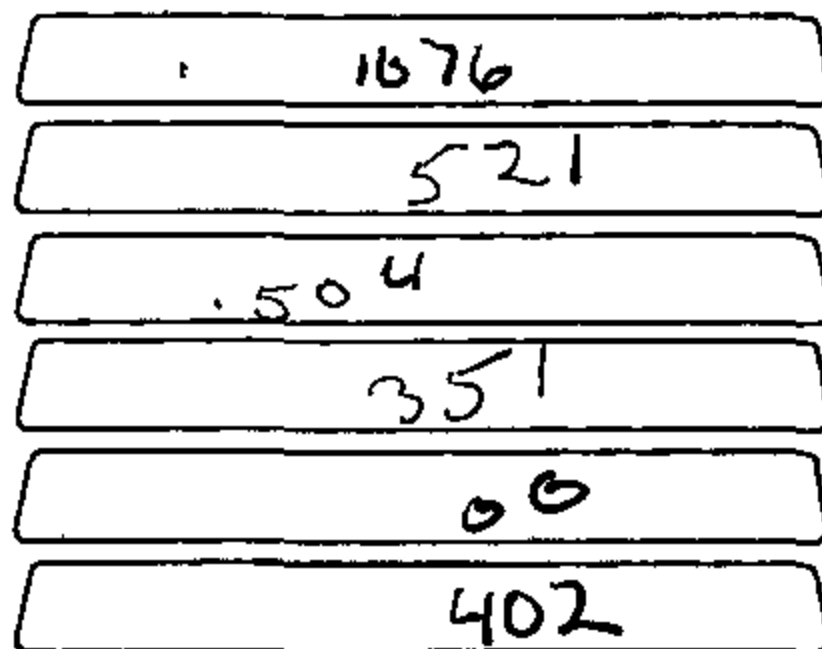
This tests all the numeral segments of the display (16 "8's").
 This tests the 16 decimal points in the display.
 This tests the 13 commas in the display. The last three positions are points only.

The self test continues to the ROM chips test.

ROM Chips Test

The self test continues with the ROM chips test showing the check sum for each of the six ROMs. The check sums will always be the same. Record them in the blank displays for future reference.

See This



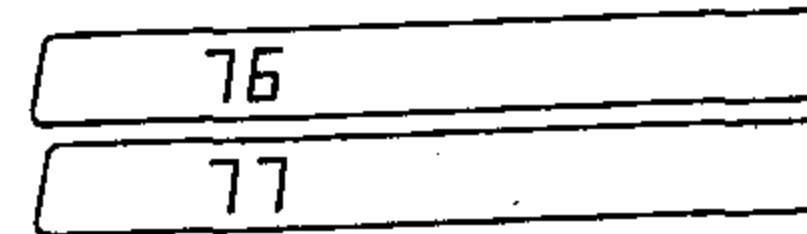
Explanation

Check sum for ROM chip 1
 Check sum for ROM chip 2
 Check sum for ROM chip 3
 Check sum for ROM chip 4
 Check sum for ROM chip 5
 Check sum for ROM chip 6

The self test continues to the RAM chips test.

RAM Chips Test

See This



Explanation

Check sum for RAM chip 1
 Check sum for RAM chip 2

After the RAM chips test, the routine loops back to the display test and continues. The self test will continue to repeat itself.

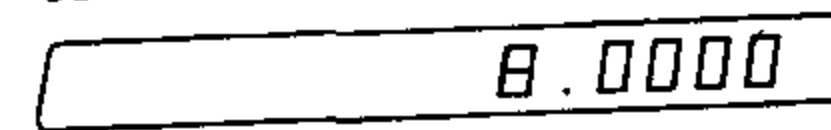
To restore normal operation you must turn off the 326 and then turn it on again.

As a final test:

Do This



See This



The self test is completed.

Appendix I: Environmental Conditions

Temperature and Relative Humidity

The ranges of ambient temperature and relative humidity which the 326 can accept without possible malfunction are as follows:

<i>Operating</i>	<i>Minimum</i>	<i>Maximum</i>
Temperature	-6.7°C (20°F)	40.6°C (105°F)
Relative Humidity	5%	95%
<i>Storage</i>	<i>Minimum</i>	<i>Maximum</i>
Temperature	-40°C (-40°F)	50°C (140°F)
Relative Humidity	0%	95%

Voltage

When using the 326 Scientist with the AC adapter, be sure that the switch on the adapter is at the proper setting (115 Volts or 230 Volts) for your area.

Appendix J: Peripheral Devices

The use of a Model 395 Interface allows the 326 Scientist to be connected to a Model 33 Teletype or any other device compatible with the Electronic Industry Association's RS 232 specifications.

For further information about the interface or compatible peripheral devices, contact your Monroe representative.

Appendix K: Notes on the Tape Cassette Drive

The following points should be noted with respect to tape cassettes and the 392 Tape Cassette Drive:


1. If tape cassettes from other suppliers than Monroe are used with the 392 Tape Cassette Drive, the tape must be computer grade tape that is free of flaws.
2. It is important not to subject the tape Cassette Drive to external shocks while it is reading or writing tape. Sudden movements or shocks may result in a failure to read or write some portion of the tape correctly, which can cause an error condition.
3. The Tape Cassette Drive's recording head must be kept clean to assure that information is correctly written onto and read from tape. After approximately 150 tape reading or writing operations, or if there seems to be difficulty in reading or writing tape correctly, clean the recording head using the following procedure:
 - (a) Open the door of the Tape Cassette Drive and remove the tape cassette.
 - (b) Rub the recording head gently using a cotton swab moistened with denatured alcohol.
 - (c) Dry the recording head with a clean dry cotton swab.

Index

A

Absolute value	70,
Accuracy and Number Sizes	168
Addition	18, 27
Angles	30
Arithmetic with	33
Converting	34, 56, 57
Degrees/Minutes/Seconds	30, 33
Degrees with Decimal Fraction	30, 34, 57
Entry of	30, 34, 57
Grads	30
Radians	30, 32, 56, 57
Antilogarithm	30, 33, 34, 56, 57
Base 10	53
Base e	53
Arc cosine	53
Arc sine	59
Arc tangent	58
Arithmetic	60
Chain	15, 18
Constant	20
Direct Register	26, 27, 28
Arithmetic (Continued)	39, 40, 42, 44
Operations	15, 18, 19, 20, 24
Auto Test Routine	180
a^x function	24, 28

B

Backspace 	100
Batteries and Recharging	169
Branch	106

C

Change Sign	10
Clear	10, 17
Clear Registers 1, 2, and 3	63
Clear All Registers	73
Conditional Tests on Number Other than Zero	118
Cosine	59


D

Data Storage Registers	34, 39, 148, 149
Decimal Point Setting	11, 12
Decisions and Conditional Testing	111, 112, 113, 118
Digit Keys	9
Direct Register Arithmetic	39, 40, 42, 44
Display Format in Programming	85
Division	20, 27, 28
D/M/S	30, 33

E

e (Base of natural logarithms)	53, 70
English/Metric Conversion	68, 70
Entering Numbers, Rules	14
Entry-Register (E-Register)	9
Equal Sign	17
Error Conditions	10, 17, 175, 176
Exchange	37
Exponent	12, 24

F

Function Keys, Summary of	73, 170
Factorial	51
Flowcharts	88, 111
Forward 	100
Fraction	71
Functions, Applications of	74, 75, 77, 78, 80

G	
GRAD/DEG Switch.....	56
H	
Hyperbolic Functions.....	54
I	
Identifier.....	72
Insert.....	100, 102
Integer.....	71
J	
Jump.....	87, 112
K	
Keys and Switches (summary).....	170
L	
Labels (symbolic addressing).....	93
Load.....	84, 85
Loading a Program.....	87
Logarithm.....	52, 53
Base 10.....	52
Base e.....	53
M	
Memory.....	84
Metric/English Conversions.....	68
Multiplication.....	19, 27
N	
Number/Operator (Num/Op) Register.....	15, 17, 18
Use with Parentheses.....	177
O	
On/Off Switch.....	9
Overlays.....	135, 151, 154

P	
Parentheses.....	22, 23, 177
Pause.....	72, 125
Pi (π).....	71
Polar to Rectangular Coordinates.....	61
Prepared Programs on Tape.....	4, 144
Program.....	82
Programming.....	82, 84
Back.....	100
Branch.....	105, 106
Forward.....	100
Identifier.....	72, 128
Insert.....	100, 102
Jump.....	89, 112
Modifying Programs.....	100, 101, 102, 103
Pause.....	72, 125
Remove.....	100
Return.....	106, 107
RUN/STEP/LOAD.....	84, 85
Running a Program.....	85, 86, 87
START/STOP.....	86
Step Feature.....	85, 123
Subroutines and Branching.....	105, 106, 107
Tape Cassette Drive Under Program Control.....	151

R

Reading a Program from Tape.....	147
Reading Data from Tape.....	149
Recall.....	37
Reciprocal.....	51
Rectangular to Polar Coordinates.....	62
Registers, Data Storage.....	34
Arithmetic with	38, 39
Exchange	37
Recall.....	37
Store	36
Reset.....	17
Return.....	106, 107
Round.....	72
Run	85, 87

S

Scientific Notation	12
Set Decimal Point.....	11
Sine.....	58
Square Root.....	52
Statistical Functions.....	63
Summation.....	63, 64
Grouped Data.....	64
Ungrouped Data.....	63
Deleting Data	65
Mean	66
Standard Deviation.....	67
Summary of Rules.....	68
Step.....	85, 123
Store	36
Subtraction.....	19, 27

T

Tangent	59
Tape Blocks.....	140
Tape Block Search	143
Tape Cassette.....	137, 138
Preconditioning.....	140
Drive	135
Connecting.....	136
Controls.....	139
Programmed Control of.....	151
Reading a Program from Tape.....	147
Writing a Program onto Tape.....	145
Reading Data from Tape.....	149
Writing Data onto Tape.....	148
Tape Movement Counter	140
Trigonometric Functions Summary.....	60

V

Variables and Constants	127
Vectors	46

W

Writing a Program onto Tape.....	145
Writing Data onto Tape.....	148



Monroe, The Calculator Company • Orange, New Jersey • Toronto, Canada • Zurich, Switzerland
Sales and Service Throughout the World

1788-5 Printed in U.S.A.

